HİTİT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

Veri Tabanı Yönetim Sistemleri

İÇİNDEKİLER

1. Veri Tabanı Temel Kavramları 4
Doç. Dr. ABDULKADİR ÖZDEMİR

2. Veri Tabanı ve Normalizasyon 24
Doç. Dr. ABDULKADİR ÖZDEMİR

3. Veri Tabanı Araçlarının Kurulumunu Yapmak 43
Dr. Öğr. Üyesi SERDAR AYDIN

4. Tabloları Oluşturmak ve Özelliklerini Belirlemek 67
Dr. Öğr. Üyesi SERDAR AYDIN

5. TSQL ile Veri Tabanı ve Tabloları Oluşturmak, Özelliklerini Belirlemek 91
Dr. Öğr. Üyesi SERDAR AYDIN

6. Sorgu Oluşturmak ve Çeşitlerini Kullanmak-1 112
Dr. Öğr. Üyesi SİNAN KUL

7. Sorgu Oluşturmak ve Çeşitlerini Kullanmak-2 136
Dr. Öğr. Üyesi SİNAN KUL

8. Sorgu Oluşturmak ve Çeşitlerini Kullanmak - 3 157
Dr. Öğr. Üyesi SİNAN KUL

9. Sorgu Oluşturmak ve Çeşitlerini Kullanmak - 4 180
Arş. Gör. YAKUP BAYOĞLU

10. İlişkili Tablolar ile Sorgu Hazırlamak 203
Arş. Gör. YAKUP BAYOĞLU

11. Görüntü (VIEW), Store Prosedür ve Fonksiyonlar 223
Arş. Gör. YAKUP BAYOĞLU

12. Veri Tabanı Yönetimi Yapmak 248
Dr. Öğr. Üyesi AHMET KAMİL KABAKUŞ

13. Veri Tabanı Güvenliğini Sağlamak 271
Dr. Öğr. Üyesi AHMET KAMİL KABAKUŞ

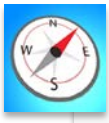
14. Veri Tabanında Yedekleme ve Geri Yükleme 291
Dr. Öğr. Üyesi AHMET KAMİL KABAKUŞ

VERİ TABANI TEMEL KAVRAMLARI

İÇİNDEKİLER

- HEDEFLER
  - Veri Tabanı ile Dosyaların Karşılaştırılması
  - Veri Tabanı Tanımı
  - Veri Modelleri
  - Veri Tabanı Veri Türleri
  - Veri Tabanı Tabloları
  - Veri Tabanı Kullanıcıları

HEDEFLER

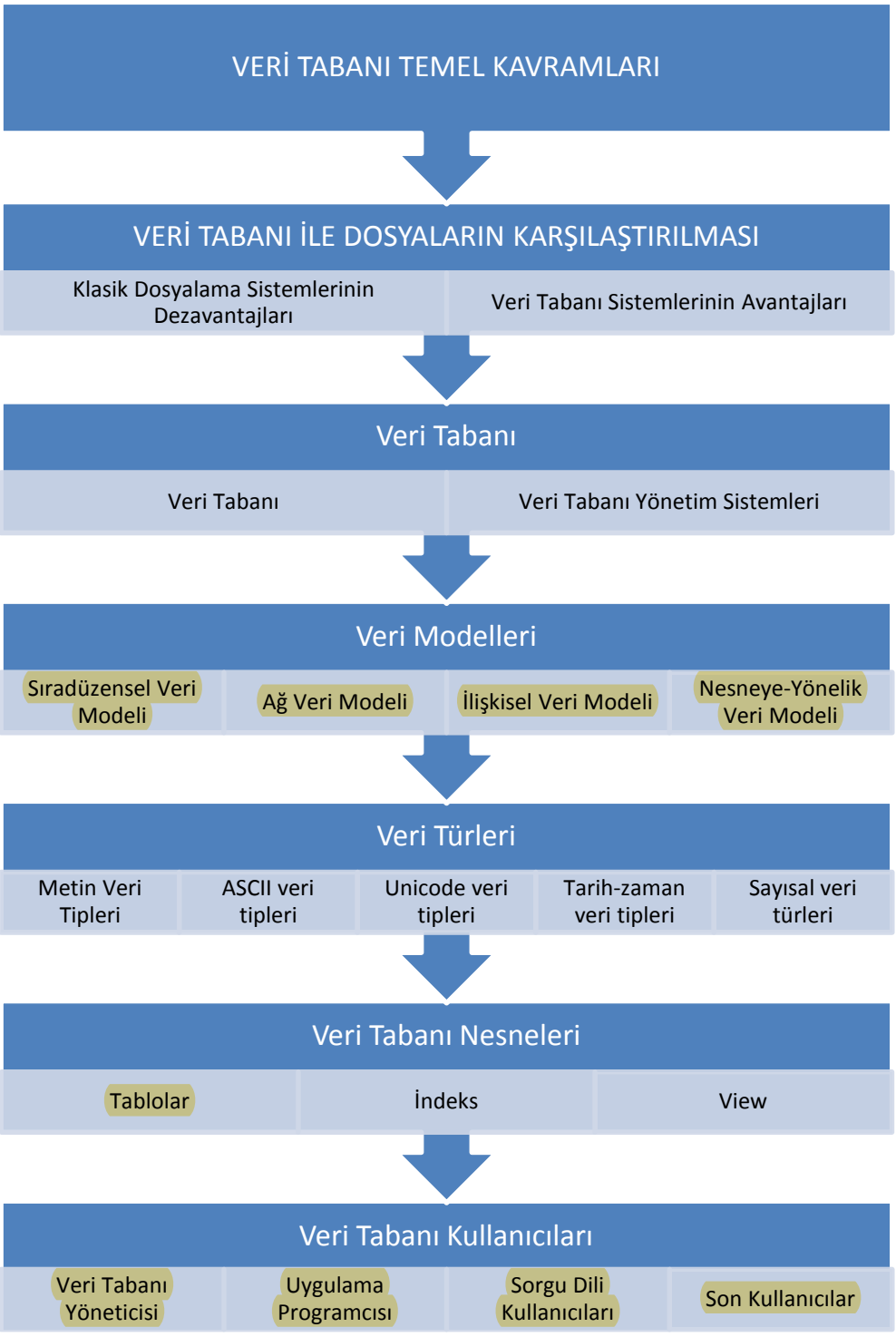


- Bu üniteyi çalıştıktan sonra;
- Neden Veri Tabanı kullanmanız gerektiği hakkında fikir sahibi olabilecek,
- Veri Tabanı türleri ile veri tabanlarında kullanılan veri türlerini kavrayabilecek,
- Veri Tabanında kullanılan tablolar hakkında bilgi sahibi olabilecek,
- Veri Tabanı kullanıcılarının yaptığı görevlerin farkını anlayabileceksiniz.

VERİ TABANI YÖNETİM SİSTEMLERİ

ÜNİTE

1


VERİ TABANI TEMEL KAVRAMLARI
VERİ TABANI İLE DOSYALARIN KARŞILAŞTIRILMASI
Klasik Dosyalama Sistemlerinin Dezavantajları
Veri Tabanı Sistemlerinin Avantajları
Veri Tabanı
Veri Tabanı
Veri Tabanı Yönetim Sistemleri
Veri Modelleri
Sıradüzensel Veri Modeli
Ağ Veri Modeli
İlişkisel Veri Modeli
Nesneye-Yönelik Veri Modeli
Veri Türleri
Metin Veri Tipleri
ASCII veri tipleri
Unicode veri tipleri
Tarih-zaman veri tipleri
Sayısal veri türleri
Veri Tabanı Nesneleri
Tablolar
İndeks
View
Veri Tabanı Kullanıcıları
Veri Tabanı Yöneticisi
Uygulama Programcısı
Sorgu Dili Kullanıcıları
Son Kullanıcılar

GİRİŞ

Bilgisayar teknolojilerindeki en büyük gelişmelerden biri de saklama ortamlarındaki kapasite artışıdır. Bu kapasite artışları aslında verilerdeki artış ve bu artan verilerin saklanması ihtiyacından kaynaklanmıştır. 1980'lerde 100 MB seviyelerinde bulunan sunucu Hard Disk kapasiteleri günümüzde 100 TB mertebelerini geçmiştir. Bu artış bir milyon kat artışı ifade etmektedir. Bu kapasite artışı verilerdeki artışın da başka bir göstergesidir.

Veri artışı aslında verilerin saklandığı klasik dosyalama yapısının sınırlarını zorlamakta ve bazen de verilerin yönetilmez hâlini almasını sağlamaktaydı. Bu zorlukların üstesinden gelmek için Veri Tabanı kavramı ortaya atılmış ve ilk olarak IBM tarafından kullanılmıştır.

Sonraki gelişmelerle hep yapısal hem de içerik olarak oldukça gelişen veri tabanları aynı zamanda yazılım dünyasının her alanında kullanılmaya başlandı. Aslında bu kullanımda veri tabanlarının standart bir yapısal dile sahip olmaları da önemli bir etkiye sahiptir.

Veri tabanlarının standart sorgulama dili veriye erişimde büyük kolaylık, esneklik ve güvenliği de beraberinde getirdiğinden her programlama dili tarafından desteklenen bir özellik olmuştur. Programlama dillerinin nerdeyse tümünde standart veri tabanı sorgulama dili olan SQL için destek sağlanmaktadır.

Bilginin işletmelerin sermayesi olarak kabul edildiği çağımızda, bu bilgileri korumak, gerektiğinde kullanmak, kullanıcıları yetkilendirmek ve içindeki anlamlı bilgileri alacak veri depolama sistemleri işletmelerde olduğu gibi artık her alanda kullanılmaktadır. İşletme ve kuruluşlar için veri tabanlarında tutulan verilerde yeni yaklaşımlara göre sermaye olarak kabul edilmektedir. Bazen bu tür verilerin alınması için çok yüklü miktarda ücretler ödenebilmektedir.

VERİ TABANI İLE DOSYALARIN KARŞILAŞTIRILMASI

Veri tabanları bu kadar yaygın kullanılmadan önce dosyalama sistemleri kullanılırdı ve saklanacak ve işlenecek her tür veri de bu dosyalar üzerinde saklanırdı. Geleneksel dosyalama sistemleri çoğunlukla ücretsizdi ve her programcı bu sistemleri tasarlayarak kullanabilirdi. Bu yönleri ile avantajlı gibi görünen geleneksel dosyalama sistemleri verileri artması ve internetin yaygınlaşması ile sorunların da ortaya çıkmasına neden olmuştur. Veri tabanı (Database) ile geleneksel dosyalama sistemlerini karşılaştırmak için iki veri saklama sisteminin iyi ve kötü yönlerini saymak yeterli olacaktır (Özseven, 2012).

Dosyalama sistemlerinde verilerin yönetilmesi çok zor ve hatalara neden olabilmektedir.

- Uygulama Bağımsızlığı: Dosyalama sistemleri her bir uygulama için ayrı ayrı tasarlandığından, aynı verilerin her bir uygulamada yeniden saklanması gerekebilmektedir. Oysaki, uygulama bağımsız veri tabanlarını farklı uygulamalar kullanabildiğinden gereksiz veri tekrarları ortadan kalkmaktadır.
- Veri Yapısı: Geleneksel dosyalama sistemlerinde verinin yapısını bilmek için uygulamayı geliştiren programcı olmak gerekirken veri tabanlarında

sadece kullanılacak verileri barındıran veri tabanına erişim yetkiniz olması yeterlidir.

- Veri Güvenliği: Dosyalama sistemlerinde verilerin güvenliğini sağlamak programı geliştiren programcının yeteneği ile sınırlı kalmaktayken veri tabanı sistemlerinde daha gelişkin bir güvenlik sistemi kullanılmaktadır.
- Programlama Dili Bağımsızlığı: Dosyalama sistemlerinde her veriye erişim için farklı diller kullanılması gerekmekte iken veri tabanlarında standart bir sorgulama dili kullanılarak her programcının kullanabileceği bir yapı oluşturulmuştur.
- Erişim Hızı: Dosyalama sistemlerinde verilere erişim hızı veri tabanlarına göre çok yavaştır.
- Kullanıcı Yetkilendirme: Dosyalama sistemlerinde birden çok kullanıcının verilere erişim hak ve yetkilerini düzenlemek çok büyük çaba gerektirirken veri tabanlarında bu işlemler çok kolaylıkla yapılabilmektedir.

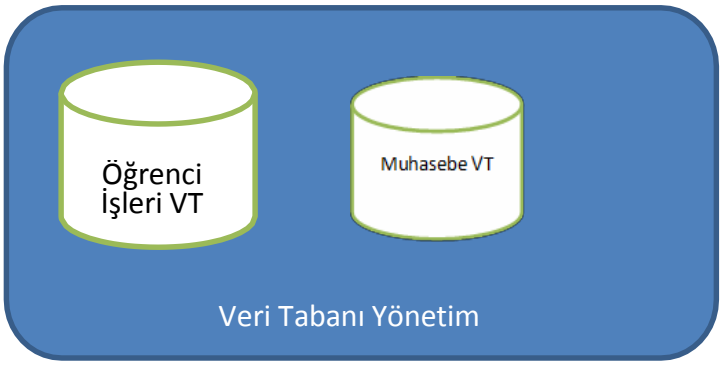
VERİ TABANI TANIMI

Veri tabanını (*Database*) tanımlamak için birbiri ile ilgili verilerin bir arada tutulduğu sistematik yapılar ifadesi kullanılmaktadır. Aynı zamanda belli bir konuya yönelik oluşturulmuş ve bu alandaki verilerin saklandığı, güncellendiği ve erişim için imkân sağlandığı organizasyonel yapılar şeklinde de tanımlanabilir.

Birbiri ile ilişkili verilerin bir arada tutulduğu yapılara veri tabanı denir.

Veri tabanları (VT) günümüzde çok kullanılmalarına karşın eğer yapılandırılması ve planlaması iyi yapılmazsa performans sorunları nedeniyle sistemlerin kilitlenmesi sebep olabilmektedir. Performans sorunlarının en aza indirgenmesi için çeşitli normalizasyon teknikleri geliştirilmiştir. Bu teknikler klasik dosyalama sistemlerinden veri tabanlarına geçişte oldukça yarar sağlamıştır.

Günümüzde doğrudan veri tabanı ile veri yönetme yazılımları geliştirenler açısından bu teknikler bilindiğinden performans sorunları çok fazla yaşanmaktadır.



Şekil 1.1. Örnek Veri Tabanı Gösterimleri

Veri tabanları genel amaçlı olarak oluşturulabileceği gibi sadece konu bazlı da oluşturulabilir. Örneğin bir işletmeye ait genel bir veri tabanı olabileceği gibi insan kaynakları, muhasebe ve öğrenci işleri gibi özel konulu veri tabanları tasarlanabilir.

Veri Tabanı Yönetim Sistemleri

Veri tabanı kavramı Veri Tabanı Yönetim Sistemi (*Database Management System*) ile karıştırılmasına karşın aslında farklı kavramlardır. İki kavramı en iyi anlamak için şu tanım daha açıklayıcı olacaktır. Farklı türde uygulamalar ve amaçlar için oluşturulmuş birden çok veri tabanının kullanılmasına ilişkin hak ve yetkiler ile verilerin bütünlüğünü ve güvenliğini sağlamaya yönelik yazılımlara Veri Tabanı Yönetim Sistemleri denir. Kullanılan bilgisayar ve hard disk kapasitesine göre çok sayıda veri tabanını yönetebilir.

Veri Tabanı Yönetim Sistemlerinin (VTYS) günümüzdeki örnekleri Oracle, Microsoft SQL, MySQL ve Sybase sayılabilir. Bu yazılımlardan ilk ikisi çok büyük veri tabanlarını yönetmek için tercih edilen ve ücretli yazılımlar iken MySQL açık kaynak kodlu ve amatör kullanımlar için ücretsiz olarak kullanılabilen bir VTYS'dir. Veri tabanına örnek olarak ise Microsoft Office paketlerinde yer alan Access verilebilir. Bu uygulama tek başına bir veri tabanıdır. Aynı zamanda bir VTYS içinde yer alan muhtelif uygulamalara ait (Personel, Öğrenci, Banka gibi) farklı verilerin saklandığı veri yapılarının her biri bir veri tabanıdır. (Kaya, 2007)

VERİ MODELLERİ

Verileri mantıksal düzeyde düzenlemek için kullanılan yapılar, kavramlar ve işlemler topluluğuna veri modeli (data model) denir. Her VTYS belirli bir veri modelini kullanır. Bir VTYS'yi kullanarak oluşturulacak her veri tabanında yer alacak veriler ve veriler arası ilişkiler, mantıksal düzeyde ilgili veri modeline göre düzenlenir; bu veri modeli kullanılarak veri tabanının kavramsal ve dış şemaları oluşturulur.

Bugüne kadar geliştirilmiş olan çok sayıda veri modeli vardır. Ancak geçmişte ve günümüzde yaygın kullanılan veri modellerini 4 grupta toplamak mümkündür: (Özseven, 2012)

- Sıradüzensel Veri Modeli (Hierarchical Data Model)
- Ağ Veri Modeli (Network Data Model)
- İlişkisel Veri Modeli (Relational Data Model)
- Nesneye-Yönelik Veri Modeli (Object-Oriented Data Model)

Bu sıralama aynı zamanda kronolojik bir sıralamadır.

Günümüzde yaygın olarak kullanılan veri tabanı veri modeli, İlişkisel Veri Modelidir.

Sıradüzensel (Hiyerarşik) Veri Modeli en eski model olup 1960 ve 1970'li yıllarda çok kullanılmıştır. 1969'da ortaya çıkan Ağ Veri Modeli 1970'li yıllarda ve 1980'li yılların ilk yarısında kullanılmıştır. İlişkisel veri modeli de ilk kez 1969 yılında ortaya atılmış, 1970'li yılların sonunda kullanılmaya başlanmış ve 1985 yılından sonra yaygınlaşmış bir yaklaşımdır. 1990'lı yıllarda yaygın kullanılan VTYS'lerin hemen hemen tümünün ilişkisel tabanlı olduğu söylenebilir.

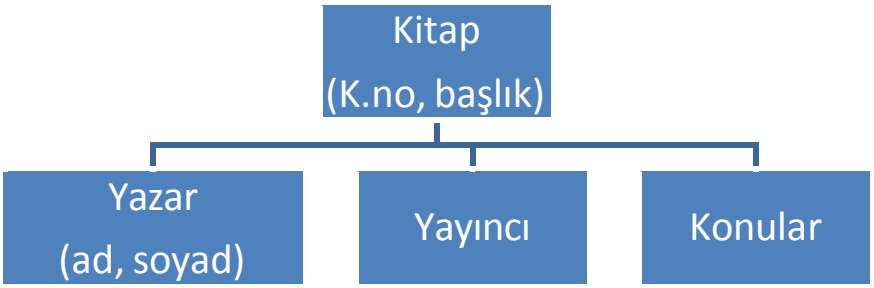
Nesneye-yönelik veri modeli yaklaşımı ise on yılı aşkın süredir gündemde olan, günümüzde çok yaygın kullanılmasa bile, kullanımı giderek yaygınlaşan bir yaklaşımdır. Geçmişe baktığımızda, ilişkisel yaklaşımın kullanılmaya başlanması ile sıradüzensel ve ağ yaklaşımlarının terk edildiği görülmektedir. Buna karşılık

nesneye-yönelik yaklaşımın kullanılmaya başlanması ile ilişkisel yaklaşım terk edilmemiştir. Günümüzde hem ilişkisel hem de nesneye-yönelik yaklaşımı birlikte kullanan VTYS'lerinin yaygınlaştığı görülmektedir.

Sıradüzensel Veri Modeli (Hierarchical Data Model)

Ağaç veri yapısına benzer. Her kaydın bir ebeveyn kaydı, birçok çocuk kaydı var (IBM IMS: Information Management System).

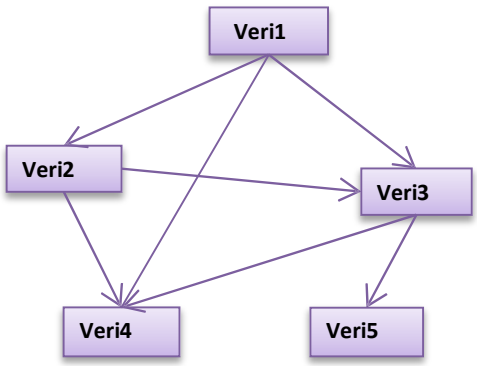
1960'lar ve 1970'lerde veri tabanları için kullanılan ilk modeldir. Bu veri modeli, ana bilgisayar ortamında çalışan yazılımlar tarafından kullanılmaktadır. IBM tarafından çıkarılan IMS (Information Management System) bu türde en çok kullanılan yazılımdır. Sıradüzensel model, bir ağaç yapısına benzer. Model dâhilindeki herhangi bir düğüm, altındaki n sayıda düğüme bağlanırken, kendisinin üstünde ancak bir düğüme bağlanabilir. Sıradüzensel yapının en tepesindeki düğüm noktasına kök denir ve bu düğümün sadece bağımlı düğümleri bulunur. Bu veri yapısını gösteren grafiğe de sıradüzensel tanım ağacı denir. Sıradüzensel bir veri yapısı ve tanım ağacı aşağıda verildiği gibi bir veri modeli oluşturur. Aşağıdaki yapı bir şirketin yönetim kademesinin hiyerarşik yapısını içermektedir.



Şekil 1.2. Sıradüzensel Veri Modeli

Ağ Veri Modeli (Network Data Model)

Ağ veri modeli 1970'li yıların başında geliştirilmiştir. Bir verinin doğası gereği birden çok veri ile ilişkisinin olmasından dolayı hızlıca kabul görmüştür. Bu modelde verilerin birbirine ağ seklinde bağlandığı varsayılır ve her kaydın bir ebeveyn ve çocukları olduğu varsayılır.



Şekil 1.3. Ağ Veri Modeli

İlişkisel Veri Modeli (Relational Data Model)

Günümüz veri tabanı üreticilerinin tümü İlişkisel Veri Tabanını desteklemektedir.

1970'lerin başında E.F.Codd tarafından geliştirilmiş bir veri modeli şeklini esas alır. Bu sistemde veriler tablolar şeklinde saklanır. Bu veri tabanı yönetim sisteminde; veri alışverişi için özel işlemler kullanılır. Bu işlemlerde tablolar operandlar olarak kullanılır. Tablolar arasındaki matematiksel bağlantılarla(ilişkilerle) temsil edilen ilişkiler belirtilir. Günümüzde hemen hemen tüm veri tabanı yönetim sistemleri ilişkisel veri modelini kullanır. Bu model, matematikteki ilişki teorisine dayanır. İlişkisel veri modelinde (relational data model) veriler basit tablolar hâlinde tutulur. Kayıtlı bu veriler ise anahtar sahalar ile birbiri ile ilişkilendirilerek kullanılır.

Nesneye-Yönelik Veri Modeli (Object-Oriented Data Model)

Günümüzde nesne kavramı her yerde kullanılmaktadır. Pek çok kelime işlemci ve hesap tablosu programlarının alıştığımız görünümlerine artık bir de nesneler eklenmiştir. Ancak bu gerçek anlamda bir nesneye yönelik yazılım demek değildir. Yüzde yüz nesneye yönelik bir yazılımın tamamen nesneye temelli çalışması gerekir. Yazılımın mutlaka nesneye yönelik bir dilde yazılmış olması beklenir. Fakat Windows gibi işletim sistemi üzerinde çalışan yazılımlar bu özelliklere tümüyle sahip değildir. Sadece nesne kavramını kullanarak bazı ek özellikler sunar. Nesneye yönelik veri tabanı da, C++ gibi nesneye dayalı bir dille yazılmış olan ve yine C++ gibi nesneye dayalı bir dille kullanılan veri tabanı anlamına gelir. Günümüz teknolojisinde yüzde yüz nesneye yönelik bir veri tabanı yaygın olarak kullanıma sunulmuş değildir. Ancak nesneye yönelik veri tabanlarının bazı üstünlükleri olacağından söz edilmektedir.

Bireysel Etkinlik

- Veri modellerindeki gelişmeler ihtiyaçlardan doğmuştur.
- İlişkisel Veri Modelinden sonra Nesneye Dayalı Veri Modeli neden ortaya çıkmıştır? İnceleyiniz.

VERİ TABANI VERİ TÜRLERİ

Farklı veri tabanı yazılımlarında birbirine benzer veri türleri kullanılsa da bazılarında farklılıklar bulunabilmektedir. Bu farklılıklar veri türüne göre olabileceği gibi bazen de kullanım şekline göre farklılık gösterebilmektedir. Bu kitapta Microsoft Firması tarafından üretilmiş olan Microsoft SQL Sever (MS SQL Server) den bahsedilecektir. (Kaya, 2007)

Bu bölümde anlatılacak veri türlerinin birçoğunun programlama dillerinde de karşılıkları bulunmaktadır. Veri türlerini program içerisinde kullanırken verilerin türlerinin yanı sıra boyutlarının da aynı olmasına özen gösterilmesi veri kaybı ve program hatalarının önüne geçilmesi açısından önemlidir.

MS SQL Server'da veri türleri sütunlar için tanımlanmaktadır ve bu tanımlama verilerimizin saklandığı tablolarda meta veri (meta data) olarak tutulmaktadır. Bir sütun için veri tipi seçilirken bu sütunda ne tür verilerin saklanacağının belirlenmiş olması gerekir. Veriler sadece harflerden, sadece sayılardan veya bunların karışımından oluşabileceği gibi tarih ve daha başka birkaç çeşit veri tipinden oluşabilir.

Metinsel veri tipleri "var" ile başlıyorsa değişken boyutta olduğunu gösterir.

Metin Veri Tipleri

MS SQL Server metinleri kaydedebilmek için 2 farklı temel veri saklama formatına destek verir: ASCII ve Unicode. Ayrıca veri tipinde "var" ifadesi kullanılmış ise bu alanın en fazla olabileceği veri saklama alanının tanımlandığını, az veri girilmesi hâlinde boş alanın veri tabanında açılmayacağını gösterir. Eğer belirtilmemiş ise bu durumda veri alanının belirtilen büyüklükte açılacağını ve değer girilmemişse boş olarak tutulacağını belirtmiş oluruz.

ASCII veri tipleri

ASCII türünden bir sütun tanımlandığında her karakter için 1 Byte veri alanı ayrılır. Parantez içerisinde verilen n değeri bu sütunun alan genişliğini gösterir.

Tablo 1.1. ASCII Türü Metin Veri Tipleri

Tür	Açıklama
char (n)	En fazla 8000 Byte'lık ASCII formatında ve sabit boyutta veri saklamak için kullanılır.
varchar (n)	En fazla 8000 Byte'lık ASCII formatında ve değişken uzunlukta veri saklamak için kullanılır. Standart değerlerden gelmeyen sütunlar için kullanılabilecek en etkin veri tipidir.

ASCII Amerikan Standartlar Enstitüsü tarafından kabul edilmiş 0-255 arasında toplam 256 adet standart karakterden ibarettir. Bu kadar sembol de her dildeki harf ve işaretleri ifade etmek için yetersizdir.

Unicode veri tipleri

ASCII kodlama standardı, bütün dillerde yer alan harfleri ve sembolleri ifade etmek için yeterli değildir. Unicode sembolleri 16 Bitlik (2 Bayt) veriler olarak ele alınır. Bu ise 2^{16}= 65536 adet sembol ifade edebilme kapasitesine ulaşmaktadır. Bu tür veri saklayan veri türlerinin adları 'n' ile başlar (nvarchar ve nchar gibi).

Tablo 1.2. UNICODE Metin Veri tipleri

Tür	**Açıklama**
nchar (n)	En fazla 8000 Byte'lık UNICODE formatında ve sabit boyutta veri saklamak için kullanılır. Bu ise 4000 karaktere karşı gelmektedir.
nvarchar (n)	En fazla 8000 Byte'lık UNICODE formatında ve değişken uzunlukta veri saklamak için kullanılır. Bu ise 4000 karaktere karşı gelmektedir. Standart değerlerden gelmeyen sütunlar için kullanılabilecek en etkin veri tipidir.

Tarih-zaman veri tipleri

Tarih formatı genellikle hassas tarih tutmak ve zamansal sorgulamalar yapmak için tercih edilir. Genellikle bir işin hangi tarihte cereyan ettiğini anlamak

için bu türden sütunlar, GETDATE () fonksiyonu default değer olmak üzere tanımlanır.

Tarih veri tipleri sayısal değerler olsa da tarihlerin sıralanmasında farklı özellikler olduğundan tarih verilerinde kullanılmalıdır.

SQL Server 2008 tarih ve zaman ile ilgili veri tiplerine, SQL Server adına yeni bir bakış açısı getiriyor. Sağlanan dört yeni veri tipi ve kullanım şekillerini teker teker ele alalım:

Tablo 1.3. Tarih Veri Tipleri

Tür	Açıklama
cfetetime	1 Ocak 1753'ten 31 Aralık 9999 arası tarihleri gösterebilen bir veri tipidir. (SQZ. Server 2005 ve eski sürümlerle aynı tip-Eski sistem uyumu için)
smalldatetime	1 Ocak 1900 ile 6 Haziran 2079 arası tarihleri gösterebilen bir veri tipidir. Dakikanın önemli olmadığı tarihleri kaydetmek için kullanılabilir. (SQL Server 2005 ve eski sürümlerle aynı tip -Eski sistem uyumu için)
date	Gün ay ve yıldan ibaret bir günü belirtebilir. Zaman belirtemez.
time	Tarih hakkında bir bilgi içermez. Sadece günün hangi anı olduğuna dair bilgi tutabilmektedir.
datetime2	İşlev olarak eski klasik DateTime (SQL Server 2005 ve aşağısı) veri tipine karşılık gelmekte olup hem tarih hem de zamanı bir arada tutabilir. Eski tipe göre daha hassas (nanosaniye türünde) tarih zaman tutabilir.
datetimeoffset	DATETIME2 tipi ile aynı özellikleri taşımanın yanı sıra tarih ve zamanın dünya üzerindeki hangi coğrafi zaman dilimi esas alınarak verildiğini de tutabilmektedir.

Sayısal veri türleri

Sayısal verilerin saklandığı veri türleridir. Sayılarda çok çeşitli veriler olduğu gibi veri türlerinde de farklılıklar vardır. Her sayının veri tabanında saklanması farklı boyutta alan tuttuğundan sayısal değerleri saklamak için kullanılacak veri türlerinin doğru bir şekilde seçilmesi hem veri tabanının performansını hem de büyüklüğünü etkilemektedir.

Sayısal değerler ikili sayı karşılıkları ile saklandığından metinlere göre daha az yer kaplar.

Tablo 1.4. Sayısal Veri Türleri

Veri tipi	Açıklama
Int	Tam sayı girişi yapılabilen veri tipi Minimum Değeri: -2,147,483,647 Maksimum Değeri: +2,147,483,647 Kapladığı Yer 4 byte
Tinyint	Int veri tipinin daha dar bir nümerik aralığa sahip hâli. Sadece pozitif sayılar alabilir. Minimum Değeri: 0 Maksimum Değeri: 255 Kapladığı Yer 1 Byte
Smallint	Int veri tipinin daha dar bir numerik aralığa sahip hâli. Minimum Değeri: -32,768 Maksimum Değeri: 32,767 Kapladığı Yer 2 byte
Decimal	Ondalıklı rakam girişi yapılabilen veri tipi. Numeric'le aynı görevi görür. Decimal (Precision, Scale) olarak tanımlanır. Precision: Rakamın ondalık dahil kaç haneden oluşacağını belirler. Scale ise ondalık

	kısmının kaç haneden oluşacağını belirler. Decimal(3,1) dediğimiz zaman 22,1, -22,1 gibi veri girişleri yapabiliriz. Eğer 22, 12 yazarsak SQL Server bunu 22,1'e dönüştürecektir.
Float	Ondalıklı rakam girişi yapılabilen veri tipidir. Minimum Değeri: -1.79E + 308 Maksimum Değeri: 1.79E + 308 Kapladığı Yer 8 Byte
Real	Real sayı girilebilen veri tipidir. Minimum Değeri: -3.40E + 38 Maksimum Değeri: 3.40E + 38 Kapladığı Yer 4 byte

Bireysel Etkinlik

- Buraya kadar anlatılan veri türlerinden farklı olarak kullanılan diğer veri türlerini araştırınız.
- Bu verilerin hangi amaçla kullanılabileceklerini tartışınız.

VERİ TABANI TABLOLARI

Veri tabanlarında veriler, tablo (table) isimli yapılarda saklanmaktadır.

Tablolar aynı tür verilerin bir arada tutulduğu mantıksal yapılar olarak tanımlanabilir. Tablolar verileri saklamanın yanında verilere ait birçok özelliği de barındırdıklarından veri tabanlarının kullanım alanları çok geniştir.

Tablolar satır ve sütunlardan oluşan veri yapıları olarak gösterildiklerinden satır (row) ve sütun (column) kavramlarının iyice kavranması gerekmektedir.

Tablolar da veri girişi çoğunlukla kayıt olarak yapılır.

Tablolarda satır olarak isimlendirilen yapı verinin birbiriyle ilişkili olan farklı türdeki değerlerinin saklandığı yapıdır. Satırlar aynı zamanda kayıt (record) olarak da isimlendirilir. Tablolarda kayıtlar eklenecek her bir veri ile arttığından tablo büyüklüğünü doğrudan belirler.

Tablolardaki diğer önemli bir kavram ise sütunlardır. Sütunlar aynı tipte verilerin kaydedildiği yapılardır. Sütunlar ise alan (field) olarak da isimlendirilmektedir. Alanlar tanımlanırken veri türü dâhil birçok özelliği de tanımlanmaktadır.

Tablolarda kayıtlar programlar ile girilirken alanlar daha çok veri tabanı yönetim sistemi ile yapılmaktadır. Tabloların alanlarına yapılacak ilave ve düzenlemeler verileri doğrudan etkilediğinden her değişiklik yapılamayabilir.

Tablo 1.5. Tablo Örneği

Sicil	Adı	Soyadı	Birimi
125	Ahmet	Narin	Satış
250	Mehmet	Okur	Pazarlama
300	Ayşe	Konak	Muhasebe
…	…	…	…

Satır ve Sütun Kavramı

Tablolara yazılan kayıtlar satır olarak da isimlendirilmektedir.

Veri tabanı tabloları mantıksal yapılar olmalarına karşın bazı kavramlarla anlam kazanabilmektedir. Bu kavramlardan en önemlileri satır ve sütundur.

Satırlar (Row) için *kayıt (Record)* kavramı da kullanılabilmektedir. *Sütunlar (Column)* için de *alan (Field)* kavramı kullanılabilmektedir.

Bir varlığa ait verilerin bir arada tutulduğu mantıksal yapıya satır denir. Satırlardaki verilerin her biri farklı bir türde olabileceğinden farklı verilerin bir arada tutulduğu veriler bütünü olarak da tanımlanmaktadır. Yukarıda örnek olarak verilen Tablo-5'te yer alan *Ahmet Narin*'e ait verilerin tümü tablodaki sütun tanımı karşılığıdır.

Sütunlara kaydedilebilecek veriler tablo tanımlanırken belirlenen veri türlerinden olabilir.

Sütun tanımında ise bir tabloda yer alan verilerin aynı tür olanların tutulduğu mantıksal yapıya denir. Yukarıda örnek olarak verilen Tablo- 5'te yer alan Sicili, Adı, Soyadı, Birimi olarak isimlendirilen alanlar sütun olarak tanımlanmaktadır. Sütunlarda saklanacak veri türleri tablo tanımlanırken belirlendiğinde her bir alana ancak belirlenen türde veriler saklanabilir. Sütunların satırlardan farkı her bir sütunun bir isme ve başka özelliklere sahip olabilmektedir. Sütunların özelliklerine ait bazı kavramlar ise aşağıda açıklanacaktır.

Veri Tipleri

Her bir sütun için bir veri türü tanımlanması gerekmektedir. Veri türleri daha önce anlatılan veri türlerinden biri olabileceği gibi veri tabanı yazılımları günün ihtiyaçlarına göre yeni veri türleri de duyurmaktadır. Bu veri türlerinin her biri saklanacak veriye uygun olarak veri tabanı tasarımcısı tarafından belirlenerek veri tabanı mantıksal yapısına dâhil edilmektedir.

Veri tipleri seçilirken bu alana saklanacak veriye uygun alan seçilmesi çok önemlidir.

Örnek

- Örneğin bir alanda sayısal veri saklanacaksa bu alanın sayısal tipte tanımlanması veri tabanının bilgisayarda daha az yer kaplamasını sağlayacağı gibi aynı zamanda bu veriler üzerinden matematiksel işlemler yapılmasına da imkân verecektir.
- Bu alanda saklanacak 12550 sayısını sayısal alanda saklamamız hâlinde 2 Byte'lık bir alana ihtiyaç duyacakken aynı veriyi metin olarak saklayacak olsak her bir haneyi 2 Byte'ta saklamak gerektiğinden 10 Byte'lık bir alana ihtiyaç duyacaktır.
- Ayrıca metin olarak veri tabanına saklanan değerlerin matematiksel işlemlere tabi tutulabilmesi için öncelikle sayısal veriye dönüştürülmeleri gerekir, aksi hâlde matematiksel işlemler yapılamaz.

Anahtar Saha

Tablolardaki kayıtlara doğrudan erişmek için veya birçok tabloda yer alan aynı varlığa ait kayıtlar arasında ilişkiler kurmak üzere tanımlanan özel alanlar kullanılır. Bu alanlar bir sütun olabileceği gibi birden fazla sütunun birleşiminde oluşturulabilir. Bu anahtarlar veri tabanı tasarımcısının veya programcının oluşturduğu yapıya göre kendilerinin belirleyebileceği tanımlardır. Veri tabanlarında genel olarak iki tür anahtardan bahsedebiliriz. Ayrıca anahtar saha olmayıp anahtar saha özelliği gösteren sahalarda tanımlayabiliriz.

Primary key

Primary Key verileri tekil olmalı ve NULL değer içermemelidir.

Bunlardan ilki *Primary Key (Birincil Anahtar)* denilen tanımdır. Birincil anahtar bir tabloda sadece bir tane olabilir ve içindeki bilgiler Unique (Tekil) olmak zorundadır. Yani bu alandaki bilgiler tekrar edemez ve yazılan bir bilgi başka bir kayıtta tekrara yazılamaz. Bu alan eğer birden fazla alanın birleşimi ile oluşturulmuşsa aynı kurallar burada da geçerlidir. Bu sahalarda NULL değer

olamaz yani kayıt oluşturulurken mutlaka bir değer girilmesi gereklidir. Bazı veri tabanı yazılımlarında Primary Key otomatik oluşturulacak şekilde de tanımlanabilir. Bu durumda veri tabanı yönetim sistemi bu alanı belli bir değerden başlatır ve artırarak tanımlar.

Bu anahtar tablolarda kayıt silme ve düzeltme işlemlerinde büyük kolaylık sağlar. Bu anahtar kullanılarak doğrudan ilgili kayda ulaşım sağlanabilir.

Personel Tablosu

Sicil	AdıSoyadı	Telefon	Satış	Amiri
125	Ali Okur	2345	10500	*1*
150	Veli Yazar	2445	12500	*125*
160	Ayşe Oku	3456	2000	*125*
180	Fatma Yaz	2367	3000	*150*

Satışlar Tablosu

İşlem	ÜrünNo	Adet	Tutar	Satıcı
1	*100.010*	5	500	*125*
2	*100.011*	10	120	*125*
3	*200.100*	6	300	*180*
4	*200.090*	7	140	*150*

Ürünler Tablosu

ÜrünNo	ÜrünAdı	Adet	A.Fiyat	S.Fiyat
100.010	Kırmızı Kalem	1000	0,50	0,70
100.011	Mavi Kalem	980	0,50	0,70
...	...	...	...	...
200.100	Kareli Defter	555	2,50	3,50

Şekil 1.4. Tablolarda Anahtar Saha

Örnek:

- Şekil 1'de verilen tablolarda "Sicil", "İşlem" ve "Ürün No" Birincil Anahtar saha olarak tanımlanmıştır.
- Bu alanlarda aynı veriler kesinlikle tekrar edemez.
- Yine bu tablolardan görüleceği gibi Personel Tablosu ve Ürünler tablosunda Birincil Anahtar saha değerleri rastgele verilmişken Satışlar Tablosundaki Birincil Anahtar saha değeri 1'den başlayarak artımlı olarak kullanılmıştır.
- Bu değerler Veri Tabanı Yönetim Sistemi tarafından verilebileceği gibi programcı tarafından da kontrollü olarak verilebilir.

UNIQUE olarak tanımlanmış bir alana bir veri sadece 1 kez kaydedilebilir.

Unique key

Bu tam olarak bir anahtar olmayıp belli sütunların tekil veri barındırabilmesi için yapılan tanımdır. Böylece kayıtlarda hatalı veri ve tekrarlı veri girilmesi önlenmiş olmaktadır. Bu alana örnek olarak ise öğrenci kayıtlarının tutulduğu bir veri tabanına öğrenci cep telefonlarının tutulduğu alanın Unique olarak tanımlanması verilebilir. Bu sayede bir cep telefonu numarası sadece 1 kez veri tabanına kaydedilebilir. Birincil anahtarda NULL değeri bulunamazken bu alanda bulunabilmektedir.

İl plaka kodlarının girildiği bir alan için 1 ile 81 arası Constraint tanımlanabilir.

Zorlayıcı (Constraint)

Herhangi bir alan için girilebilecek verileri kısıtlayıcı kurallara zorlayıcılar denir. İlgili alana girilebilecek değerleri sınırlayan bir deyim yazılarak tanımlanabilir. Kullanımı bazı veri giriş işlemlerinde faydalıdır ve özellikle yanlış bilgi girişini engelleyerek verilerin doğru girilmesini zorunlu hâle getirir. Kullanıcı, zorlayıcıda belirtilen kural dışında bir veriyi tabloya yazmaya çalıştığında, VTYS hata verir. Böylelikle veri tabanına kullanıcının yanlış değerler girmesi önlenmiş olur ve veri tabanında tutarlılık sağlanmış olur.

Örnek:

- Örneğin, bir öğrencinin sınıf bilgisine ait değerler yazılırken bu alan için rakamsal 1 ile 6 arasında bir zorlayıcı değer tanımlanırsa veri girişi sırasında 1 ile 6 arasındaki değer dışında bir değerin sınıf bilgisi alanına yazılması engellenmiş olur.
- Dolayısı ile sınıf için yazılmaması gereken bir değer, bilgi girişi başlangıcında kontrol edilmiş olur.

İndeks (Index)

Veri tabanlarında indeks oluşturularak veriler veri tabanındaki kayıtlı oldukları sıradan başka bir sırada gösterilebilir ve tıpkı kütüphanedeki bir kitab-

ulaşmada olduğu gibi istenilen veriye daha kısa sürede ve kolayca ulaşılabilir.

Temelde indekslerin ilişkisel veri tabanında şu üç işlevi vardır:

- Tekil indeksler, veri ilişkilerini ve veri bütünlüğünü sağlayan birincil anahtar alanlar oluşturmada kullanılır.
- İndeks olan alanın değerine göre bir kaydın kayıtlar arasındaki sırasını gösterir.
- Sorguların neticelenme sürelerini kısaltır.

View (Görüntü)

Bazen, tabloları olduklarından farklı gösterecek filtrelere ihtiyaç duyulur. Bu türden işlevler için VIEW kullanılır. VIEW 'ler, saklanmış sorgulardan ibarettir.

Aslında tablo gibi kullanılsa da hâlihazırda böyle bir tablo veri tabanında bulunmaz, sadece view (görüntüsü) bulunur. VIEW'ler şu görevler için kullanılır:

- Kullanıcıların bazı kritik tabloların sadece belli sütunlarını veya satırlarını görmesi istenildiğinde,
- Kullanıcıların, çeşitli birim dönüşümlerinden geçmiş değerler görmeleri gerektiğinde,
- Hâlihazırdaki tablolarda var olan verilerin başka bir tablo formatında sunulması gerektiğinde,
- Çok kompleks sorguları basitleştirmek için kullanılabilmektedir.

Joining (ilişkilendirme)

İki veya daha fazla tabloyu birlikte sorgulama işlemine join ismi verilir. İlişkisel veri tabanının en temelinde birden fazla tablo üstünde birlikte işlem yapabilmek yatar. Bu sayede verilerin tekrarlanması önlenmiş olur ve sonuçta veri yönetimi kolaylaşır.

VERİ TABANI KULLANICILARI

Veri tabanı kullanıcıları birkaç grup altında toplanabilir. Bu kullanıcıların bazıları işletmelerin büyüklüğüne bağlı olarak birleştirilebilmektedir. Bazı durumlarda da aşağıdaki kullanıcılardan birkaçını program hazırlayan programcılar yerine getirmektedir.

Veri Tabanı Yöneticisi çok büyük veri tabanlarını yönetmezse verilerin kullanımında performans sorunları yaşanır.

Veri Tabanı Yöneticisi

Veri Tabanı yöneticisi veri tabanındaki en yetkili kullanıcıdır. Genellikle İngilizce ismi olan Database Administrator'un baş harfleri olan DBA ismi ile anılır. DBA'nın birçok görevi vardır. Bunlar:

- Veri tabanı Tasarımı,
- Performans Analizi,
- Erişim Yetkilerini Düzenleme ve Erişim Sağlama,
- Yedekleme ve Geri Yükleme,
- Veri Bütünlüğü Sağlama,
- Sistem Sürekliliği Sağlama şeklinde özetlenebilir.

Uygulama Programcısı

Bu kullanıcıların görevleri ise son kullanıcılara yönelik uygulama yazılım geliştirmek ve veri işleme dilini kullanmak şeklinde özetlenebilir.

Sorgu Dili Kullanıcıları

Veri tabanından alınması istenen raporlara ilişkin sorgular hazırlamak ve programlar dışında yapılacak veri güncellemesi veya düzenlemelerini yapacak sorguları hazırlamak olarak tanımlanabilir. Bu kullanıcılar yapacakları sorguları SQL dili ile hazırlar.

Son Kullanıcılar

Uygulama programcıları tarafından geliştirilen yazılımları veri tabanı yöneticisinin yetkilendirdiği tablolar üzerinde kullanan ve işlemler yapan kişiler olarak özetlenebilir.

Özet

- **VERİ TABANI İLE DOSYALARIN KARŞILAŞTIRILMASI**
- Verilerin bilgisayarlarda saklanma süreci ile başlayan veri saklama ortamlarına bakıldığında, veri tabanlarından önce klasik dosyalama sistemlerinin kullanıldığı görülmektedir. Bu dosyalama sistemlerinin zaman içinde ortaya çıkan kısıtları ve eksiklikleri olduğundan veri tabanı kullanımı hızla yaygınlaşmıştır.
- Klasik dosyalama sistemlerinde ki yazılım bağımlılığı, veri yapsının bilinme zorluğu, verileri güvenliği, veriye erişim hızının düşük olması ve kullanıcı sayısındaki kısıtlar veri tabanlarına geçişi hızlandırmış ve kolaylaştırmıştır.
- **VERİ TABANI TANIMI**
- Veri tabanları, verilerin saklanmasında en önemli yazılımların başında gelmektedir. Veri tabanları farklı konular için oluşturulduğundan bu farklı veri tabanlarını yönetmek, barındırmak ve güvenliğini sağlamak gibi konularda kolaylık sağlayan Veri Tabanı Yönetim Sistemleri kullanılmaktadır.
- **Veri Tabanı Yönetim Sistemleri**
- Veri tabanı yönetim sistemleri adından da anlaşılacağı gibi farklı yazılım veya uygulamalara yönelik verileri barındıran veri tabanlarının tek bir yazılımla yönetilmesini sağlayan sistemlerdir.
- **VERİ MODELLERİ**
- Zaman içinde gelişen teknoloji ve ihtiyaçlara göre çeşitli veri tabanı modelleri kullanıma sunulmuştur. Sıra düzensel veri modeliyle başlayan süreç ağ veri modeli bir süre yoluna devam etmiş ve günümüzde yaygın olarak ilişkisel veri modeli kullanılmaktadır. Ancak programlama dillerinin ve teknolojinin yazılımsal nesneleri kullanım ihtiyaçları ile Nesneye Yönelik Veri Modellerini destekleyen veri tabanı yönetim sistemleri de kullanılmaktadır.
- **VERİ TABANI VERİ TÜRLERİ**
- Veri tabanlarında bilgisayar yazılımlarının desteklediği veri türlerinin tümü desteklenmektedir. Desteklenmemesi hâlinde programlarda veri tabanı kullanımlarında sorunlar yaşanacağından veri tabanı yönetim sistemi geliştiricileri güncel veri türleri için yazılımlarını yenilemektedir. İlk veri tabanları ASCII gibi 256 karakterlik kodları desteklerken günümüz veri tabanları UNICODE ile on binlerce karakter kodunu destekler hâle gelmiştir.
- Veri tabanlarında kişisel verilerden ölçümlerle elde edilen verilere kadar çeşitli veriler saklanmaktadır. Dolayısıyla da veri tabanlarında ad, soyad gibi metinsel veriler yanında çeşityli ölçümlere ait değerlerin saklandığı sayısal değerlerde saklanmaktadır. Günümüzde ise web adreslerinde XML verilerine kadar farklı türde verilerde saklanmakatadır.
- **VERİ TABANI TABLOLARI**
- Veri tabanlarının mantıksal yapısında temel veri saklama nesnesi olan tablolar ve bunlara ait tanımlar olan satır, sütun, kayıt, alan, anahtar saha ve diğer kavramların bu ders notunun sonraki bölümlerinin daha iyi anlaşılması açısından önemlidir.
- **Anahtar Saha**
- Veri tabanı yönetim sistemlerinde verinin hızlı, güvenli ve doğru bir şekilde erişiminde etkin olan anahtar sahalar vardır. Bunlardan primary key bir tablodaki en önemli veri erişim anahtarı olarak kullanılmaktadır.

Özet (devamı)

- Veri tabanlarında tabloların yanında view, index gibi çok sayıda farklı nesne de bulunmaktadır. Bu nesnelerin tümü veri tabanı yönetim sistemi içinde yer alan mantıksal tanımlardır aslında.
- Veri tabanlarında büyük veri yığınlarından işe yarayacak verileri süzmek için view, joining ve index gibi kavramlar kullanılmaktadır. Bu mantıksal yapılar verilerin daha güvenli bir şekilde ve hızlı olarak kullanılabilmesini sağlamaktadır.
- **VERİ TABANI KULLANICILARI**
- Veri tabanı yönetim sistemleri veri tabanı ve verilerin güvenliğini sağlamak için de çeşitli yöntemler uygulamaktadır. Bunlardan en önemlisi kullanıcı tanımları ve bu kullanıcılara verilen yetkiler olarak düşünülebilir. Bu yetkiler ile kim hangi veriye, tabloya veya nesneye erişecek veya erişemeyecek türünden çok ayrıntılı yetkilendirmeler yapılabilmektedir.

DEĞERLENDİRME SORULARI

1. Geleneksel dosyalar ile veri tabanlarının karşılaştırılmasında aşağıdakilerden hangisi söylenemez?
   a) Veri tabanları farklı programlara veri sunabilir.
   b) Dosya sistemleri her program için ayrı tasarlanır.
   c) Dosyalama sistemleri daha güvenlidir.
   d) Veri tabanları veriye daha hızlı erişim sağlar.
   e) Veri tabanları çok sayıda kullanıcıya hizmet verebilmektedir.

2. Veri tabanı ile veri tabanı yönetim sistemi için aşağıdakilerden hangisi doğrudur?
   a) Veri tabanı yönetim sistemi birden çok veri tabanı barındırabilir.
   b) Veri tabanı farklı verileri barındırır.
   c) Veri tabanı veri tabanı yönetim sistemlerini içerir.
   d) Veri tabanı yönetim sistemleri en küçük veri tabanıdır.
   e) Access bir veri tabanı yönetim sistemidir.

3. Aşağıdaki veri modellerinden hangisi ilk veri tabanlarında kullanılmıştır?
   a) Ağ Veri Modeli
   b) Hierarchical Data Model
   c) İlişkisel Veri Modeli
   d) Object-Oriented Data model
   e) Relational Data Model
4. Aşağıdaki veri türlerinden hangisi ASCII veri türüdür?
   a) nchar
   b) datetime
   c) Int
   d) varchar
   e) real

5. Aşağıdaki veri türlerinden hangisi Byte olarak daha fazla alan kaplar?
   a) Int
   b) Tinyint
   c) Smalint
   d) Real
   e) Float
6. Veri tabanında bir öğrenciye ait veriler aşağıdaki yapılardan hangisinde saklanır?
   a) Tablo
   b) Sütun
   c) Alan
   d) View
   e) Kayıt

7. Bir tabloda tanımlanan primary key için aşağıdakilerden hangisi yanlıştır?
   a) Tekrarlı veri içerir.
   b) NULL değeri içeremez.
   c) Bir tabloda bir alanda olabilir.
   d) Unique olmalıdır.
   e) Kayıt düzeltme ve silme gibi işlemleri kolaylaştırır.

8. Aşağıdakilerden hangisi tabloların istenilen kısımlarını göstermeye yarayan mantıksal yapılardır?
   a) Index
   b) Primary Key
   c) Foreign Key
   d) View
   e) Constraint

9. Aşağıdakilerden hangisi veri tabanı yöneticisinin görevlerinden biri değildir?
   a) Veri Tabanı Tasarımı
   b) Performans Analizi
   c) Uygulama Programı
   d) Yedekleme ve Geri Yükleme
   e) Veri Bütünlüğü Sağlama

10. Büyük veri tabanı yönetim sistemlerinde en fazla kaç tür kullanıcı bulunabilir?
   a) 3
   b) 4
   c) 1
   d) 2
   e) 5

Cevap Anahtarı
1.c, 2.a, 3.b, 4.d, 5.e, 6.e, 7.a, 8.d, 9.c 10.b

YARARLANILAN KAYNAKLAR

Kaya, Y. ve Tekin, R. (2007). Veri tabanı Uygulamaları. Papatya Yayınları, İstanbul.

Özseven, T. (2012). Veri Tabanı Yönetim Sistemleri-1. Murathan Yayınları, Trabzon.

VERİ TABANI VE NORMALİZASYON

İÇİNDEKİLER

- HEDEFLER
  - İlişkisel Veri Tabanı Tasarımı
  - İlişkisel Veri Tabanının Kavramsal Tasarımı
  - Varlık İlişki Modeli
  - Varlık kümesi
  - Zayıf Varlık Kümesi
  - Veri Tabanı Normalizasyon Kuralları
  - Bağımlılıklar
  - Normal Formlar

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Veri tabanı verileri arasındaki ilişkiyi kavrayabilecek,
  - Veri tabanı yapısını anlayabilecek,
  - Veri tabanlarında yapılabilecek, normalizasyon türlerini ve kullanım alanlarını kavrayabileceksiniz.

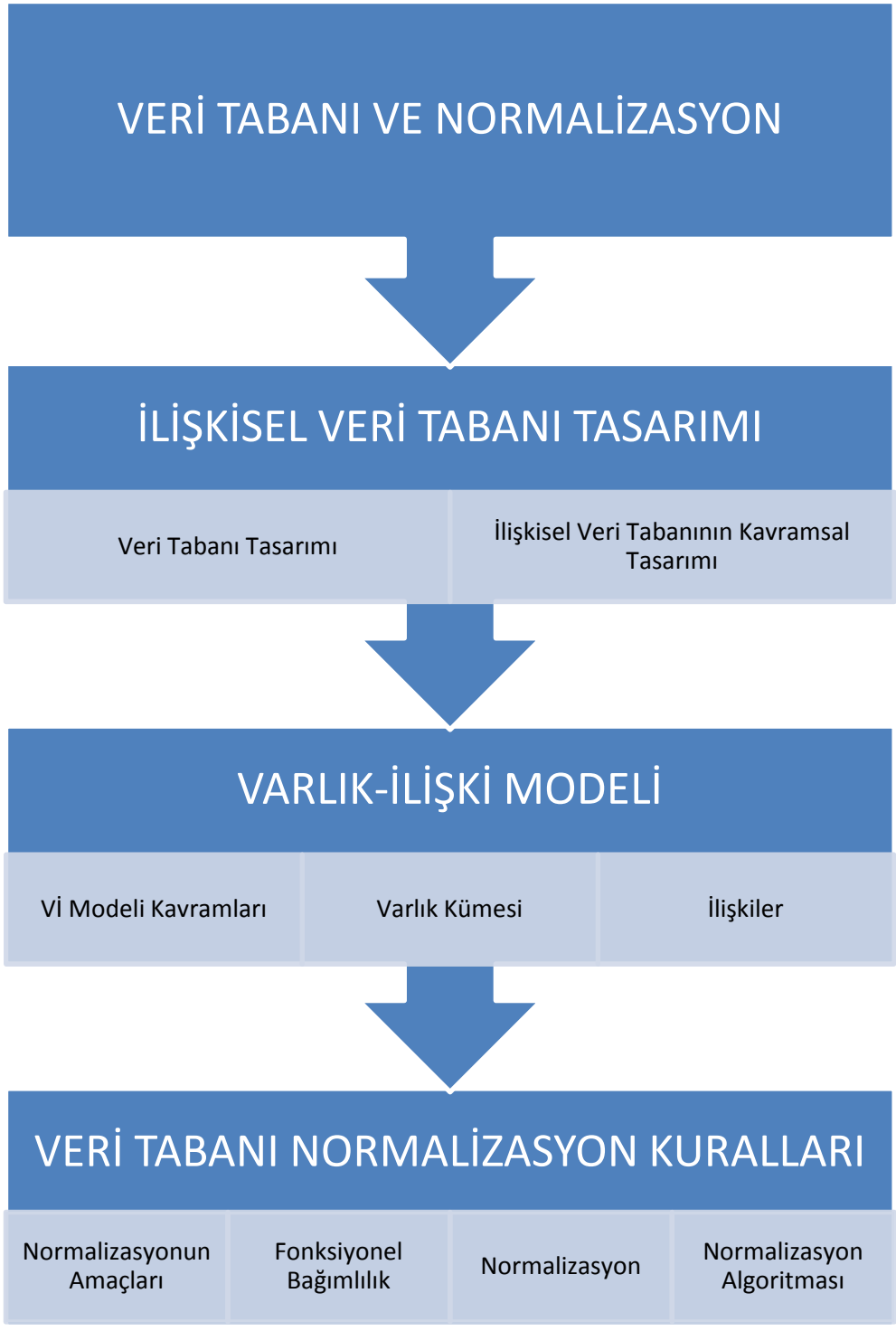
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Doç. Dr. Abdulkadir ÖZDEMİR

ÜNİTE

2


VERİ TABANI VE NORMALİZASYON
İLİŞKİSEL VERİ TABANI TASARIMI
Veri Tabanı Tasarımı
İlişkisel Veri Tabanının Kavramsal Tasarımı
VARLIK-İLİŞKİ MODELİ
Vİ Modeli Kavramları
Varlık Kümesi
İlişkiler
VERİ TABANI NORMALİZASYON KURALLARI
Normalizasyonun Amaçları
Fonksiyonel Bağımlılık
Normalizasyon
Normalizasyon Algoritması

GİRİŞ

Veri tabanları verilerin organize edildiği ve sonradan erişilip üzerinde güncelleme gibi işlemler yapıldığından erişim hızı ve doğru veri güncellemenin önemli olduğu yapılardır.

Verilerin hızlı bir şekilde erişilip üzerlerindeki işlemlerin doğru ve tek bir yerde yapılabilmesi için ise veri tabanının iyi tasarlanmış ve tablolarında normalizasyon işlemlerinin yapılmış olması gereklidir.

Veri tabanı tasarımı verileri saklanacak konuya uygun bir veri yapısı tasarımı ile başlar. Veri yapısının uygun olarak seçilmemiş olması verilerin istenilen ayrıntıda kaydedilmesine engel olduğu gibi sonrasında veri tabanında yapısal değişikliklerin yapılmasına yol açacaktır.

Veri tabanları tasarlanırken bazen gerçek uygulamadaki veri yapıları ve ilişkilere göre tasarlanır. Bunun sonucu olarak ta veri tabanlarında performans kaybına ve hatta veri bütünlüğünün bozulması nedeniyle veri kaybına yol açabilir.

Bu gibi durumların önüne geçmek için de veri tabanları tasarlandıktan sonra normalizasyon kurallarına göre yeniden yapılandırılır. Bu yapılandırmalar veri tabanı tablolarının durumuna bir adımda veya durumuna göre birden fazla adımda yapılabilir. Normalizasyon 5 ana ve bir alt işlemden oluşur ve bunlar *normal form* olarak isimlendirilir. Birçok veri tabanı için ilk üç normal form yeterli olurken çok sayıda ve karışık yapılarda veriler barındıran veri tabanları için diğer normal formların işletilmesine ihtiyaç duyulabilir.

Veri tabanı tasarımcıları tecrübe kazandıkça normal formlara uygun tasarımlar yapmaya başlayacaklarından normalizasyona ihtiyacı olmayan tasarımlarda yapabilmektedir . Ancak tasarım konusunda tecrübesi olmayanların, normalizasyon kurallarına göre veri tabanı yapılarını düzeltmeleri gerekir.

İLİŞKİSEL VERİ TABANI TASARIMI

Veri tabanı tasarımında yetenek, bilgi ve tecrübe önemlidir. Ancak veri tabanı sistemi ilişkileri ve normalizasyon kuralları da bilinmelidir. Bu kurallar veri tabanın hangi şartlara göre tasarlanması gerektiğini belirler ve veri tekrarını, veri kaybını veya veri yetersizliğini önler. Bu kurallar depolanan veri miktarı artıkça önem kazanmaktadır.

Veri Tabanı Tasarımı

Veri tabanı projenin başında tasarlandığından, çok iyi tasarlanmalıdır ki daha sonraki uygulamalarda iş sürecini aksatmasın. (Kaya, 2007)

Aşağıda veri tabanı tasarımı yapılırken izlenecek adımlar verilmiştir:

- *Depolanacak Veriler:* Veri tabanı içerisinde tutmak istediğiniz bilgiler belirlenerek bunlar gruplandırılmalıdır.
- *Tabloların Oluşturulması:* Belirlenen veri grupları ve sütunlar doğrultusunda tablolar oluşturulur.

- *Anahtar Sütunların Belirlenmesi:* Kayıtların birbirinden ayırt edilebilmesi için anahtar sütun oluşturulur. Anahtar sütunun tanımlanma zorunluluğu yoktur ama verilere daha çabuk ulaşmak ve tekrar eden kayıtların önlenmesi için kullanılabilir.
- *Tabloları Bölme:* Tabloda tekrar eden kayıtlarla karşılaşılacaksa tekrar eden sütun için yeni tablo oluşturulur.
- *İlişkilerin Kurulması:* Projelerin büyük kısmında tablolar arasında kesinlikle ilişki oluşturulmaktadır.

İlişkisel Veri Tabanının Kavramsal Tasarımı

Kavramsal tasarım, veri tabanında tutulacak verilerin daha üst seviyede gösterilmesi için kullanılır. Kavramsal tasarım için en çok kullanılan model Entity Relationship (ER) [Varlık İlişki (Vİ)] modelidir.

Varlık İlişki Modeli, veri tabanı tasarım aşaması için önemlidir.

Varlık-ilişki modeli kavramsal tasarım için kullanılan en popüler modeldir. Bu model kullanılarak VTYS'den bağımsız modelleme yapılarak ve ilişkiler tanımlanarak herhangi bir VTYS veri tabanına dönüştürülebilir.

Varlık-ilişki modelinde kullanılan şekiller veri tabanlarının şematik olarak tasarlanması için kullanılır.

Varlık-ilişki modelinde temel üç öge vardır. Bunlar (Özseven, 2012):

- *Varlık:* Modelin en temel öğesidir. Var olan ve benzerlerinden ayırt edilebilen her şey varlıktır. Örneğin, kitap, öğrenci, araba birer varlıktır. Birden fazla varlığın oluşturduğu kümeye varlık kümesi denilir. Veri tabanı olarak düşünülürse her bir tablo bir varlık kümesidir.
- *Nitelik:* Varlıkların her bir özelliği bir nitelik olarak ifade edilir. Örneğin, öğrencinin numarası ve bölümü öğrenci varlığının nitelikleridir. Nitelik bağlı olduğu varlığa düz bir çizgi ile birleştirilir. Veri tabanı olarak düşünülürse tablonun her bir sütunu bir varlığı gösterir. Bir niteliğin değeri her bir varlık için farklıysa bu nitelik anahtar nitelik olarak belirlenir. Anahtar nitelik şema içerisinde niteliğin altı çizilerek gösterilir. Örneğin, her bir öğrenci varlığı farklı öğrenci numarası niteliklerine sahip olacağı için öğrenci no niteliği anahtar nitelik olarak belirlenebilir.
- *Domain (Etki Alanı):* Niteliklerin alabileceği değer aralığıdır. Örneğin, öğrenci notlarını içeren sınav niteliği için alacağı değerleri 0 ile 100 arasında belirlemek etki alanı oluşturmaktadır. Etki alanı Vİ şeması içerisinde gösterilmez.
- *İlişki:* Farklı varlıklar arasındaki ilişkileri ifade eder. Örneğin, öğrenci ve dersler ayrı varlık kümeleridir ama öğrenciler ders almak zorunda olduğu için iki varlık arasında ders alma ilişkisi vardır. Baklava dilimi ilişkili olduğu varlıklarla düz çizgi ile bağlanır. Tablolar arasında kurulan ilişkiler (1-n, 1-1, n- m) model içerisinde ilişki olarak geçmektedir. İki varlık kümesi arasında birden fazla ilişki bulunabilir.

VARLIK-İLİŞKİ MODELİ

Veri tabanı tasarım işlemi birkaç aşamadan oluşur (Özseven, 2012):

- İlk olarak veri tabanı kurulacak kuruluşun ihtiyacı olan bilgilerin analizi yapılır.
- Veriler toplanıp sistem analizi yapıldıktan sonra, ilişkisel veri modeline göre veri tabanı şeması oluşturulur. Bu şema, veri tabanı kullanıcılarının ihtiyaç duydukları verilerin kısa açıklamasıdır.
- Bu şema içerisinde varlık tipleri ve ilişkilerin açıklaması bulunur. Daha sonra bu şemaya göre tablolar oluşturulur.

İlişkisel veri modeli türlerinden olan ER (Entity-Relationship) veya Türkçedeki kullanımı ile Varlık-İlişki (Vİ) modeli günümüzde yaygın olarak kullanılmaktadır.

Vİ Modeli Kavramları

Varlık ve nitelikleri

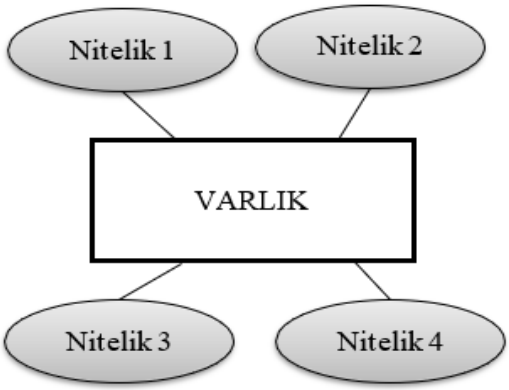
Gerçek hayatta diğerlerinden ayırt edilebilen nesnelere varlık denir. Bir varlık, kişi, araba, ev veya çalışan gibi fiziksel nesneler olabileceği gibi, şirket, iş veya ders gibi fiziksel olmayan nesneler de olabilir. Her varlık kendini tanımlayan kendine has özelliklere sahiptir. Örneğin, bir çalışan varlığı; çalışan adı, yaşı, adresi, maaşı ve görevi özellikleri ile tanımlanır. Her varlığın niteliğinin bir değeri olur.

Gerçek hayatta olduğu gibi varlıkların nitelikleri de kendileri kadar önemlidir.

Varlık Kümesi

Veri tabanında benzer varlıklar ve nitelik değerlerinden oluşan kümeye varlık kümesi denir.

Vİ şemasında varlık kümeleri dikdörtgen içinde belirtilir. Nitelikler ise oval bir daire içinde belirtilerek ilgili varlık kümesine çizgi ile bağlanır. Şekil 2.1.de bir varlığa ait Vİ gösterilmiştir. Aşağıdaki örnekte varlığa ait 4 nitelik olduğu gösterilmiştir.



Şekil 2-1. Varlık İlişki Diyagramı örneği

İlişkiler

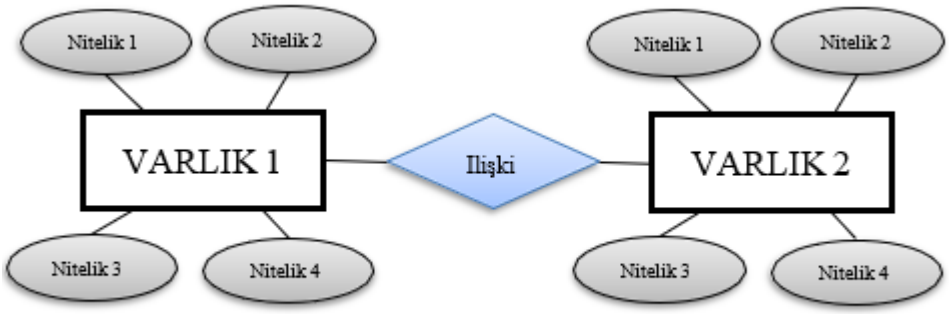
İki veya daha fazla varlık kümesi arasında kurulan anlamlı bağıntılara ilişki denir. İlişkiler Vİ şemasında eşkenar dörtgen ile gösterilir. Eşkenar dörtgen içine

ilişkinin adı yazılır. İki varlık kümesi arasında birden fazla ilişki de olabilir.

İlişki derecesi

İlişkide bulunan varlık kümelerin sayısı ilişkinin derecesini oluşturur.

Genelde ilişkiler iki varlık kümesi arasında yapılır. Ancak bazı durumlarda, ilişkide ikiden fazla küme yer alabilir. İlişkilerin gösterildiği şema Şekil 2.2.de gösterilmiştir.



Şekil 2.2. İlişki Diyagramı

İlişki türleri

Varlık kümeleri aralarında 3 türde ilişki kurulabilir.

Varlıklar arasında üç tür ilişkiden bahsedilebilir.

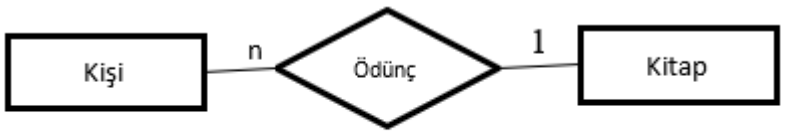
Bire-bir (1-1) ilişki

Bir varlık kümesinin bir elemanının başka bir varlık kümesindeki bir elemanla ilişkisi olarak tanımlanabilir. Buna örnek olarak bir kişinin sadece bir kişi ile evlenmesi gösterilebilir.

Şekil 2.3. Bire-Bir İlişki

Bire-çok (1-n) ilişki

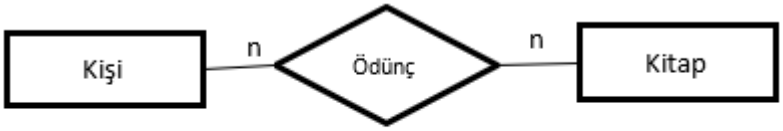
Bir varlığın başka bir kümedeki farklı varlıkla birden çok ilişkisi buna örnek olarak verilebilir. Bir kişinin birden çok kitabı kütüphaneden ödünç alması buna örnek olarak verilebilir.



Şekil 2.4. Bire Çok İlişki

Çoka-çok (n-n ya da n-m) ilişki

Bir varlığın başka varlıkla iki yönlü veya çoklu ilişkisidir. Çok sayıda kişinin ödünç kitap alması ve kitapların çok sayıda kişiye ödünç verilmesi buna örnek olarak verilebilir.

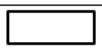


Şekil 2.5. Çoktan Çoka ilişki

Zayıf Varlık Kümeleri

Bir varlık kümesi anahtar niteliğe sahip değilse zayıf varlık kümesi olarak adlandırılır. Zayıf varlık kümeleri güçlü varlık kümeleri ile ilişkilendirilerek kullanılır. Yani zayıf varlık kümelerinin güçlü varlık kümelerine var olma bağımlılığı vardır. Zayıf varlık kümeleri çift çizgili dörtgen ile gösterilir.

Tablo 2.1. Varlık İlişki sembolleri

Sembol	Açıklama
	Varlık kümesi
	Nitelik
	Anahtar Nitelik
	İlişki
	Çok Değerli Nitelik
	Zayıf Varlık Kümesi

Örnek

- Bir üniversitenin personellerine ait araç plakalarının tutulduğun tablodaki bilgiler doğrudan ilgili personele bağlı olduğundan, yani personelin üniversitede olmaması halinde plakanın ilişkilendirileceği kayıt olmaması nedeniyle plaka verileri zayıf varlık kümesi olarak isimlendirilir.

VERİ TABANI NORMALİZASYON KURALLARI

Normalizasyon, veri tabanının tasarım aşamasında veri tekrarını, veri kaybını veya veri yetersizliğini önlemek için gerçekleştirilen işlemlerdir. (Özseven, 2012)

Tabloların oluşturulması, tablolar arasında artık veri ve tutarsız ilişkilerin ortadan kaldırılması normalizasyonun temel amacıdır.

Veri tabanı normalleştirmesi için birkaç kural bulunmaktadır. Her kurala "normal form" adı verilir. İlk kural kullanılıyorsa, veri tabanının "ilk normal formda" olduğu söylenir. İlk üç kural kullanılıyorsa, veri tabanı "üçüncü normal

formda" olarak nitelendirilir. Başka normalleştirme düzeyleri de kullanılabilmekle birlikte, çoğu uygulama için en yüksek düzey üçüncü normal formdur ama daha üst seviye olan dört ve beşinci normalizasyon kuralları da vardır.

Genel olarak normalleştirme için ek tablolar gerekir ve çoğunlukla bunun ek yük getirdiğini düşünülebilir. Normalleştirmenin ilk üç kuralından biri ihlal edildiği takdirde, uygulamalarda artık verilerin oluşturabileceği sorunlara karşı hazır olunması gerekir.

Bu kurallar, depolanacak verilerin neler olacağını belirler ve kayıtlar arasında veri tekrarını, veri kaybını ve yetersizlikleri engeller. Normalizasyon kurallarına uygun tasarım yapılmadığında kayıt güncelleme, kayıt bulma ve yeni kayıt işlemlerinde sorunlarla karşılaşılabilir.

Üst normalizasyon kuralları alt normalizasyon kurallarını kapsamaktadır. Yani, 2NF kuralları 1NF kurallarını ve 3NF kuralları 1NF, 2NF kurallarını kapsar.

Normalizasyonun Amaçları

- *Veri Bütünlüğünü Sağlamak:* Eğer bir sütun için gereksiz veri tekrarı varsa bu sütun bir süre sonra birbirinden farklı değerler içermeye başlayacaktır. Örneğin, bir veri tabanı içerisinde kişilerin adresleri birden fazla tabloda kullanılmışsa ve veri bütünlüğü sağlanmazsa zamanla aynı kişi için birden fazla adres bilgisi oluşacaktır. Böyle bir durumla karşılaşmamak için normalizasyon kuralları adresin tek bir tabloda saklanması ile veri bütünlüğü sağlanabilir.
- *Uygulamadan Bağımsızlık:* Hazırlanacak ilişkisel model kullanılacak uygulamaya göre değil saklanacak veriye göre hazırlanmalıdır. Böylece, kullanılan uygulama değişse de veri modeli tutarlı olarak çalışmaya devam edecektir.
- *Performansı Arttırmak:* Gerekli normalleştirilme işlemleri yapılmış veri tabanı, veri tekrarını en aza indireceği için veri tabanı boyutunu azaltacağından disk alanından da tasarruf sağlanmasına neden olacaktır. Bu ise veri tabanı içerisinde gerçekleştirilen arama ve veri güncelleme işlemlerinin daha kısa sürede gerçekleşmesini sağlayacaktır.

Üst normalizasyon kuralları alt normalizasyon kurallarını kapsar.

Fonksiyonel bağımlılık

R'nin ilişkiyi (relation) A ve B'nin bir özelliği ya da özellik kümesini temsil ettiği kabul edilir. Eğer R ilişkisinde her bir A değeri, tam olarak bir B değerine işaret ediyorsa B, A ya fonksiyonel olarak bağımlıdır denebilir. Bu durum sembolik olarak da A → B şeklinde gösterilebilir (A fonksiyonel olarak B'yi tanımlar.). Aşağıdaki örnek tabloda "M.No" birincil anahtar ise kişi "Ad-Soyadı" bu anahtara fonksiyonel olarak bağlıdır denir. Bu tablodan hareketle her M.No'su bilinen kişinin adı ve şehrini elde etmek mümkündür.

Tablo 2.2. Fonksiyonel Bağımlılık Örneği

M. No	Ad-Soyadı	Şehir
125	Ali Okur	Erzurum
126	Eda Koş	Ankara
130	Veli Yaz	Erzurum

Kısmi bağımlılık

Anahtar olmayan alan, birleşik anahtarın sadece bir kısmı ile belirlenebiliyorsa buna kısmi bağımlılık denir. Tablo 2.2.deki kişiler tablosunun müşterilere ait olduğu ve bu tablodan ayrı olarak birde aşağıdaki gibi satışlar tablosu olduğunu düşünürsek burada M. No'nun tekrar ettiğini ve bu kayıtlardan birinin olmaması durumunda veri bütünlüğü bozulmayacağından kısmi bağımlılıktan bahsedilebilir.

Tablo 2.3. Kısmi Bağımlılık Örneği

M. No	Ad-Soyadı	ÜrünKodu	Adet
125	Ali Okur	M0125	1
126	Eda Koş	M0225	1
130	Veli Yaz	M0125	2
125	Ali Okur	M0225	1

Normalizasyon

Normalizasyon yapılırken uyulması gereken kurulların her birine normal form adı verilir. 5 tip normalleştirme kuralından bahsedilebilir.

Veri tabanı normalizasyonunda ilk üç normal form daha çok kullanılır.

- Birinci Normal Form (1NF)
- İkinci Normal Form (2NF)
- Üçüncü Normal Form (3NF)
- Dördünü Normal Form (4NF)
- Beşinci Normal Form (5NF

Bu Normal formlardan ilk üçü çok kullanılırken son ikisinin kullanımı daha azdır. İlk üç normal form kayıt güncelleme, kayıt silme ve kayıt bulmada kolaylık sağlar.

3NF'da olan tablolar 1NF ve 2NF'ye uygundur. 2NF'da olan tablolar ise 1NF'ye uygundur.

Birinci Normal Form (1NF)

Her sütunda sadece bir veri saklanmalıdır.

Bu formda veri tabanında bulunan tablolar ilişkilendirilebilir bir şekilde tasarlanmalıdır.

- Her sütunda sadece bir veri tutulmamalıdır yani birden fazla bilgi tek bir sütunda olmamalıdır.
- Bir alan içerisindeki bilgi özel karakterlerle ayrılarak tutulmuş olmamalıdır. Bir veri tabanı yukarıda belirtilen kurallara uyuyorsa 1NF yani birinci normal form kuralı yerine getirilmiş denir.

Tablo 2.4.Örnek Tablo

Ogr_no	Ad-Soyad	Bolum_kodu	Bolum	Ders_kodu	Sınav_S
110012	Hasan Oku	BLGP	Bilgisayar Prog.	B101, B102, B103	70, 85, 45
110013	Eda Boz	BLGP	Bilgisayar Prog.	B102, B103, B104	25, 60, 55
110014	Ahmet Yaz	ELK	Elektrik	E201, E205, E204	45, 66, 74
110015	Ayşe Koş	ELK	Elektrik	E205, E204, E208	60, 78, 75

Bu tabloyu 1NF kurallarına göre inceleyecek olursak:

Ders_konu ve Sınav_S sütunlarındaki her bir alanda birden çok veri tutulmaktadır, veriler ise virgüllerle ayrılmış durumdadır. Bu ise yukarıdaki iki kuralı ihlal anlamına gelmektedir.

Yukardaki tabloya ilk normalizasyon kuralı uygulandığında kurala uymayan değerlerle tablo aşağıdaki gibi olacaktır:

Tablo 2.4. 1NF Uygulanmış tablo

Ogr_no	Bolum_kodu	Ad-Soyad	Bolum	Ders_kodu	Sınav_S
110012	BLGP	Hasan Oku	Bilgisayar Prog.	B101	70
110012	BLGP	Hasan Oku	Bilgisayar Prog.	B102	85
110012	BLGP	Hasan Oku	Bilgisayar Prog.	B103	45
110013	BLGP	Eda Boz	Bilgisayar Prog.	B102	25
110013	BLGP	Eda Boz	Bilgisayar Prog.	B103	60
110013	BLGP	Eda Boz	Bilgisayar Prog.	B104	55
110014	ELK	Ahmet Yaz	Elektrik	E201	45
110014	ELK	Ahmet Yaz	Elektrik	E205	66
110014	ELK	Ahmet Yaz	Elektrik	E204	74
110015	ELK	Ayşe Koş	Elektrik	E205	60
110015	ELK	Ayşe Koş	Elektrik	E204	78
110015	ELK	Ayşe Koş	Elektrik	E208	75

1NF sonucu elde edilecek tablonun yeni halinde tekrarlı satırlara sahip olacağı için bu tabloya satır ekleme, satır silme ve satır güncellemede sorunlar yaşanabilir.

İkinci Normal Form (2NF)

1NF'de karşılaşılan sorunlardan güncelleme sorununu çözmek için tablo 2NF kuralına uygun hâle getirilir. 2NF, nitelikler arasındaki fonksiyonel bağımlılıktan yararlanılarak tabloların birden fazla tabloya dönüştürülmesi ile sağlanır.

Dönüştürmede uygulanacak kurallar ise şöyledir.

Kurallar:

- Bir tablo içinde tanımlı ancak anahtar olmaya uygun sütunlar anahtar olarak tanımlanmalı ve tanımlı birincil anahtar sütunlara bağlanmalıdır. Anahtar sütunun ihtiyaç duyduğu bilgileri içermelidir. Örneğin, öğrenci bilgilerinin tutulduğu bir tabloda not ve ders bilgisinin olması gereksizdir. Çünkü notlarla ilgili asıl erişim noktası öğrenci numarası olacaktır. Bölüm veya ad gibi bilgiler ayrıntı olarak kalacaktır. Bunu çözmek için öğrenci bilgileri ve not bilgileri ayrı tablolarda tutulmalıdır.
- Anahtar sütun birden fazla sütunun birleşiminden oluşuyorsa tabloda yer alacak veriler iki sütuna da bağımlı olmalıdır. Tek sütuna bağımlı ise ayrı

Tablolarda aynı kayda ait tekrar eden veriler ayrı bir tabloda tutulmalıdır.

bir tabloda tutulmalıdır. Örneğin, bir ders farklı bölümlerde veriliyorsa ders kodu bölüm kodu ile birlikte bir anahtar olarak tanımlanmalıdır.

Tablo 2.5.e 2NF uygulandıktan sonra tablolarımız şu şekli alacaktır. Buradan da görüldüğü gibi öğrencilerin bilgilerini tutan bir özlük tablosu oluşturulmuştur. Bu tabloda Ogr_no birincil anahtar saha olarak tanımlanmalı ve dersler tablosundaki yabancı anahtar sahası olarak tanımlanacak olan Ogr_no sütunu özlük tablosundaki Ogr-No alanı ile ilişkilendirilmelidir.

Tablo 2. 5. 2NF Sonrası Öğrenci Özlük Tablosu

Ogr_no	Ad-Soyad	Bolum_kodu	Bolum
110012	Hasan Oku	BLGP	Bilgisayar Prog.
110013	Eda Boz	BLGP	Bilgisayar Prog.
110014	Ahmet Yaz	ELK	Elektrik
110015	Ayşe Koş	ELK	Elektrik

Tablo 2.6. 2NF Sonrası Dersler Tablosu

Ogr_n	Ders_kodu	Sınav_S
110012	B101	70
110012	B102	85
110012	B103	45
110013	B102	25
110013	B103	60
110013	B104	55
110014	E201	45
110014	E205	66
110014	E204	74
110015	E205	60
110015	E204	78
110015	E208	75

İkinci Normal Form sonucu güncelleme sorunu çözülmüştür ancak kayıt ekleme ve kayıt silme sorunu devam etmektedir.

Üçüncü Normal Form (3NF)

Tablolar arasında anahtar alanlar ile bağımlılıklar belirlenir.

2NF'de karşılaşılan sorunları çözmek için geçişli bağımlılıkları da ortadan kaldırmak gerekmektedir. 2NF'de sadece anahtar sütunlara göre bağımlılıklar kullanılmıştı, 3NF'de ise bir tablo içerisinde anahtar olmayan bir sütun, başka bir tablonun anahtar sütunu veya bulunduğu tablonun sütunlarıyla ilgili olmalıdır veya bir tablo için anahtar olmayan bir sütun anahtar olmayan başka hiçbir sütuna bağımlı olmamalıdır. Ayrıca, veri tabanındaki ilişkiler 2NF kuralına uymalıdır.

Ogrenci tablosu için Bolum → Bolum_kodu geçişli bağımlılığı mevcuttur. Ogrenci tablosundaki sorunlardan kurtulmak için bu bağımlılığın da kaldırılması gereklidir. Bağımlılığı kaldırmak için bölümler ayrı bir tablo olarak oluşturulmalıdır.

Bu düzenlemeden sonra Öğrenci Özlük tablosunu Öğrenci tablosu ve bölümler tablosu olarak 2 ayrı tablo hâline getirebiliriz. Tablolar arasındaki ilişkiler ise Bolum_kodu ve Bolum_k arasında oluşturulacak ilişki ile veri bütünlüğü sağlanmış olur.

Birçok yazılım uygulamasında üçüncü normal form ile normalizasyon işlemi tamamlanmış olmaktadır. Ancak çok kapsamlı ve karmaşık veri yapılarında üçüncü

normal form yeterli olmayacağından diğer normal form işlemleri de yapılmalıdır.

Boyce-Codd Normal Form(BCNF)

Bir tablonun BCNF olup olmadığını anlamak için tablonun tüm belirleyicileri tespit edilip, her birinin aday anahtar özelliği taşıyıp taşımadığını kontrol edilir.

Veri tabanı tabloları 1NF, 2NF ve 3NF işlemine uygun olarak düzeltildikten sonra bazı tablolarda tüm alanların aday anahtar olarak kaldığı görülebilir. Bu durumda BCNF işlemine göre tablo veya tabloların parçalanması gerekir.

Veri tabanı tasarımında A → B şeklinde bir fonksiyonel bağımlılık bulunuyorsa bu bağımlılıktaki B birincil anahtar olmak zorundadır. 3NF tasarımında A anahtarı bir aday anahtar (candidate key) olmak zorunda değildir. Ancak BCNF'de bunun tersine A → B şeklindeki bir fonksiyonel bağımlılık durumunda A bir aday anahtar olmalıdır.

Tablo 2.6.daki Eda Boz silindiğinde Tablo 2.7.deki Eda Boz'a ait dersler yok olacaktır. Ayrıca yeni bir öğrenci kayıt olana kadar başka bir ders ortaya çıkmayacaktır.

Dördüncü Normal Form (4NF)

Birincil anahtar olan sütunlar ile anahtar olmayan sütunlar arasında birden fazla bağımsız 1-n ilişkiye izin verilmez. 4NF sağlamak için her bağımsız 1-n ilişki için ayrı tablo oluşturmak gerekir.

4NF'yi açıklamak için dersleri veren hoca tablosunu ele alalım. Bu tabloda her hocanın bir dersi verdiği öngörülmüştür.

Kayıtlar arasında birden çoğa ilişkiler varsa bunlar 4NF ile ortadan kaldırılır.

Tablo 2.9. 4NF Uygulanacak Tablo

HocaNo	Hoca Ad-Soyad	Bölümü	Ders
125	Hakan Bilen	BLGP	Veri Tabanı
130	Ayşe Soykan	ELK	Devreler
140	Ahmet Kul	BLGP	Programlama

Tablo incelendiğinde ilk olarak herhangi bir sorun olmadığı görülür. Bu tablo için birincil anahtar HocaNo sütunudur. Bu tabloya göre her öğretim elemanı sadece bir derse girebilir. Aynı öğretim elemanı için iki veya daha fazla ders girilmek istendiğinde birincil anahtardan dolayı aynı HocaNo kodu kullanılamayacak dolayısıyla da yeni ders girilemeyecektir ve 1-n ilişki söz konusu olmayacaktır. 4NF kuralını ve 1-n ilişkiyi sağlamak için mevcut tablo iki ayrı tabloya bölünür. Aşağıda 4NF kuralına uygun tablolar verilmiştir.

Tablo 2.10. 4NF Uygulamasından Sonraki Hoca Tablosu

HocaNo	Hoca Ad-Soyad	Bölümü
125	Hakan Bilen	BLGP
130	Ayşe Soykan	ELK
140	Ahmet Kul	BLGP

Tablo 2.11. Dersler Tablosu

HocaNo	Ders
125	Veri Tabanı
130	Devreler
140	Programlama
125	Donanım
125	Sistem Analizi
130	Tasarım

Görüldüğü gibi hocaların verebileceği dersler için ayrı bir tablo oluşturulmuş olup, burada her hocaya birden fazla ders girmek mümkün olmuştur.

Tekrarları önlemek için her tabloyu mümkün olduğunca küçük parçalara bölmek gerekir. Aslında ilk 4 kural bu işlemi gerçekleştirir ama bu kurallar kapsamında olmayan tekrarlamalar da beşinci normalizasyon kuralı ile giderilir.

Beşinci normal formda olan bir kayıt aynı zamanda dördüncü, üçüncü, ikinci ve birinci normal formlardadır. Beşinci normal form dördüncü normal formdan simetrik bir kısıtlama olmadığı sürece farklı değildir. Böyle bir kısıtlamanın yokluğunda, dördüncü normal formdaki bir kayıt tipi her zaman beşinci normal formdadır.

Tekrarlamaları ortadan kaldırmak için her bir tablonun mümkün olduğunca küçük parçalara bölünmesi gerektiğinden daha önce bahsedilmişti. İlk dört normal formda olmayan tekrarlamalar beşinci normal formlarla giderilebilir.

Örneğin bir firma için stok kaydını tutması istenilen bir veri tabanı programı hazırlandığında, ilgili firma daha sonraki bir zamanda ürünün stoktan çıkış sebebini de sisteme girmek istediğini belirttiğinde bu sütuna girilecek olan bilgiler bellidir.

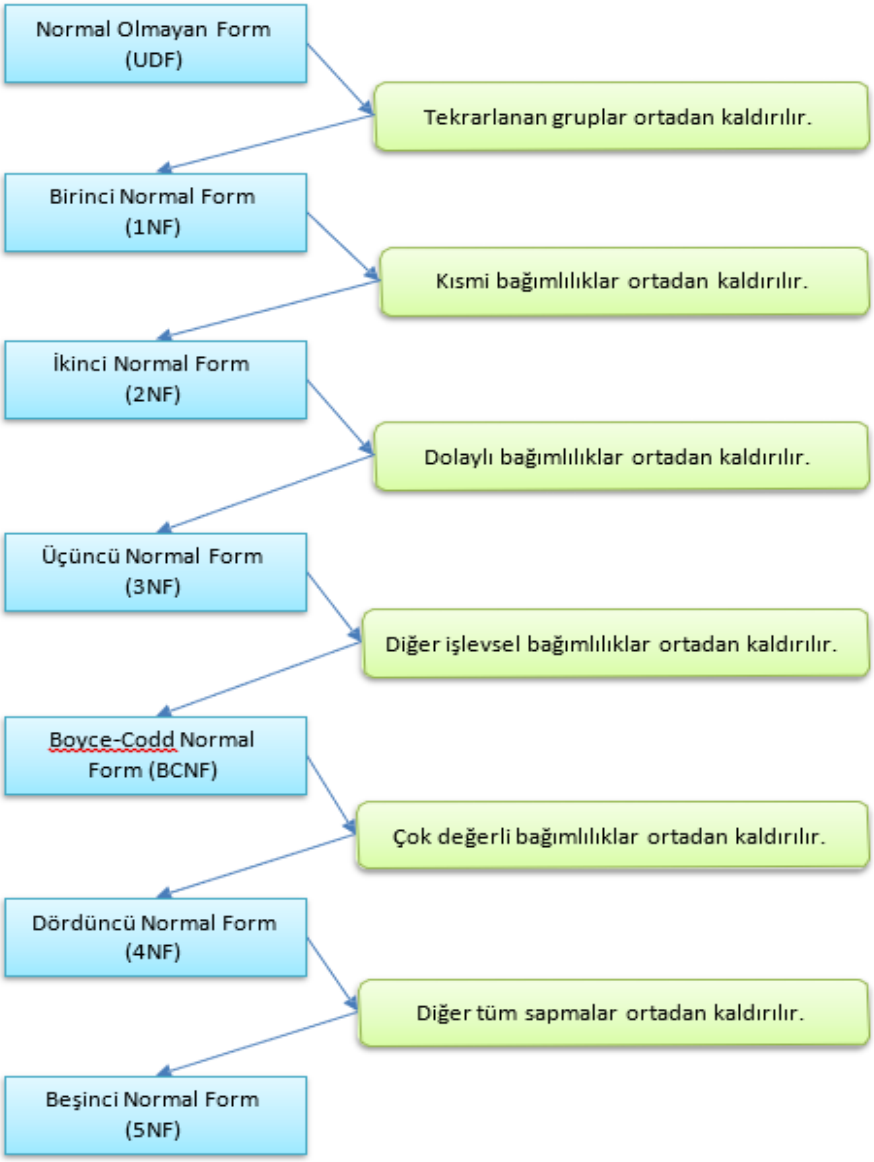
Bireysel Etkinlik

- Normalizasyon yapılmayan tablolarda verilerin işlenmesi daha zordur.
- Normalizasyon tablo sayısını artırsa da sonuçta tablolara erişim ve veri tekrarını ortadan kaldırması nasıl bir tasarruf sağlar.

Normalizasyon Algoritması

Öğrenciler için hazırlanan ilk tabloda normalizasyon yapılmadığından verilerle ilgili sorunlar vardı. Bu sorunları ortadan kaldırmak ve veri tabanına erişimi kolaylaştırmak için normalizasyon kuralları uygulandı. Bu kuralları şekilsel olarak göstermek gerekirse aşağıdaki akış şeması ortaya çıkar. Bunu ise

normalizasyon algoritması olarak kullanabiliriz.



Şekil 2.6. Normalizasyon Akış Şeması

Özet

- **VERİ TABANLARI**
- Veri tabanları çok sayıda güncel ve erişilebilen verinin saklandığı ve istenildiğinde de kullanıcılara veya sistemlere sunulduğu yazılımlardır.
- **İLİŞKİSEL VERİ TABANI TASARIMI**
- Veri tabanları zaman içerisinde farklı veri modellerine göre tasarlanmış olsalar da günümüzde en yaygın kullanım alanı bulan ilişkisel veri tabanlarıdır.
- **Veri Tabanı Tasarımı**
- Veri tabanı tasarımındaki ilk aşama varlık ilişki diyagramlarıdır.
- Bu diyagramlar sayesinde varlıkların verileri arasındaki ilişkiler ortaya konulduğundan veri tabanının mantıksal tasarımı da kolaylıkla yapılabilmektedir.
- **İlişki türleri**
- Varlık ilişki diyagramlarında üç tür ilişki bulunabilir. Bire-bir, Bire-çok veya çoğa-çok ilişki. Bu ilişki türüne göre mantıksal tasarım şekillenir.
- Veri tabanları tasarlanırken yazılımlarda kullanılacak veya saklanmak istenilen verilere göre tasarım yapılır.
- Veri tabanı tasarımında depolanacak veri türleri, kullanılacak tablo sayıları, tablolar arasındaki ilişkiler göz önüne alınarak tasarım yapılmalıdır.
- Bu tasarıma kavramsal tasarım denir ve veri tabanlarının kullanımında ortaya çıkabilecek eksiklik, tekrar ve veri kayıplarını en aza indirmek açısından önem taşır.
- Veri tabanlarında yer alan ve en önemli veri saklama ortamı olan tablolarda aynı özellikteki verilerin bir sütunda saklanmasına dikkat edilir.
- Tablolarda saklanan verilerde birbirleriyle ilişkili başka bir ifade ile aynı varlığa ait verilerin aynı satırda tutulması kuralı uygulanır.
- Bu nedenle de bu tür veri tabanlarına yapısal veri tabanları denir. Çünkü veriler belli bir yapıda saklanmakta ve kullanılmaktadır.
- **Normalizasyon**
- Yukarıda sayılan işlemler yapılsa dahi çoğunlukla veri tabanı tablolarında birçok tekrarlı veri olduğu görülebilir.
- Bu tür verilerin atılması verilerde anlam kaybına neden olacağından veriler atılamaz, ancak bu tür veriler farklı tablolarda saklanarak veri tekrarları ortadan kaldırılır.
- Tekrarlı veya aynı varlığa ait olup kaç kez tekrar edeceği belli olmayan türdeki verilerin veri tabanı mantığını bozmadan saklanmasına yönelik düzenlemeye normalizasyon denilmektedir.
- Normalizasyon da amaç veri bütünlüğünü sağlamak, uygulama bağımsızlığını oluşturmak ve veri erişiminde performans artışını sağlamak şeklinde de özetlenebilir.
- Normalizasyon kuralları tablolara uygulandığında tablolarda sorgulama ve güncellemelerde daha dikkatli olmak gerekir.
- Normalizasyon işlemlerinde fonksiyonel ve kısmi bağımlığında göz önüne alınması gerekir.
- Normalizason işlemleri normal formlar şeklinde isimlendirilmektedir.
- Birden beşe kadar normal formlar vardı. Genellikle bunlardan ilk üçü uygulandıktan sonra diğerlerine ihtiyaç duyulmaz ve veri tabanında normalizasyon sağlanmış olunur.

Özet (devamı)

- **Birinci Normal Form**
- Birinci normal form ile verilerde yer alan tekrarlar ortadan kaldırılır.
- **İkinci Normal Form**
- İkinci normal form ile kısmi bağımlılıklar ortadan kaldırılır.
- **Üçüncü Normal Form**
- Üçüncü normal form ile dolaylı bağımlılıklar ortadan kaldırılır.
- **Dördüncü Normal Form**
- Çok karmaşık veri yapılarında ise dördüncü ve beşinci normal formlar uygulanabilir ancak genellikle ilk üç normal form ile ihtiyaç duyulan normalizasyon sağlanmış olunmaktadır.
- Bir varlığa ait veriler birden çok tabloda saklandığından ilgili tüm tablolara da gerekli işlemlerin yapılmasında özen gösterilmelidir.

DEĞERLENDİRME SORULARI

1. Veri tabanı tasarlanırken aşağıdakilerden hangisi kavramsal tasarımda yapılır?
   a) Varlık İlişki Modeli
   b) Birinci Normalizasyon Formu
   c) Beşinci Normalizasyon Formu
   d) Bire-Bir İlişki
   e) Bire-Çok İlişki

2. Veri tabanındaki varlıkların özelliklerine ne denir?
   a) İlişki
   b) Nitelik
   c) Etki Alanı
   d) Varlık
   e) Domain

3. Varlık ilişki modelinde varlıklar aşağıdakilerden hangisi ile gösterilir?
   a) Elips
   b) Kare
   c) Silindir
   d) Çizgi
   e) Dikdörtgen

4. Bir dersi birden çok öğrencinin alması aşağıdaki varlık ilişkilerinden hangisidir?
   a) Bire-bir
   b) Çoğa-çok
   c) Bire-çok
   d) İkiye-bir
   e) İkiye-iki

5. Varlık ilişki diyagramlarında anahtar nitelik nasıl gösterilir?
   a) Kare içinde
   b) Çift çizgili elips
   c) Tek çizgili elips
   d) Altı çizilerek
   e) Koyu yazılarak

6. Veri tabanında veri bütünlüğünü sağlamak, uygulama bağımsızlığı ve performansı artırmak için aşağıdakilerden hangisi yapılmalıdır?
   a) Varlık ilişki diyagramı
   b) Tekilleştirme
   c) Anahtar saha belirleme
   d) Normalizasyon
   e) Sıkıştırma

7. Anahtar olmayan sahaların birleştirilerek anahtar saha olarak kullanılması aşağıdakilerden hangisidir?
   a) Fonksiyonel bağımlılık
   b) Birincil anahtar saha
   c) Kısmi bağımlılık
   d) İkincil anahtar saha
   e) Varlık ilişki diyagramı

8. Genel olarak tablolarda normalizasyon adımlarının hangisi yeterli olmaktadır?
   a) 3NF
   b) 2NF
   c) 5NF
   d) 4NF
   e) 1NF

9. Birinci normal formda aşağıdaki işlemlerden hangisi yapılır?
   a) Sütunları birleştirmek
   b) Uygun sütunları anahtar alan yapmak
   c) Tablolar arasında ilişkiler tanımlamak
   d) Bir sütundaki farklı verileri farklı tabloya aktarmak
   e) Bir tabloda tekrar eden verileri başka bir tabloya kaydetmek

10. İkinci normal formda aşağıdaki işlemlerden hangisi yapılır?
    a) Sütunları birleştirmek
    b) Uygun sütunları anahtar alan yapmak
    c) Tablolar arasında ilişkiler tanımlamak
    d) Bir sütundaki farklı verileri farklı tabloya aktarmak
    e) Bir tabloda tekrar eden verileri başka bir tabloya kaydetmek

Cevap Anahtarı

1.a, 2.b, 3.e, 4.c, 5.d, 6.e, 7.c, 8.a, 9.d, 10.b

YARARLANILAN KAYNAKLAR

Kaya, Y. Ve Tekin, R. (2007). Veritabanı Uygulamaları. Papatya Yayınları, İstanbul.

Özseven, T. (2012). Veri Tabanı Yönetim Sistemleri-1. Murathan Yayınları, Trabzon.

VERİ TABANI ARAÇLARININ KURULUMUNU YAPMAK

İÇİNDEKİLER

- SQL Server 2008 Veri Tabanı Sürümleri
- İşletim Sistemi Seçimi
- Güvenlik
- Sistem Gereksinimleri
- SQL Server 2008 Kurulumu

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Veri tabanı araçlarından olan SQL Server 2008'in kurulumunun nasıl yapıldığını kavrayabilecek,
  - Kurulum sırasında karşılaşabileceğiniz olası problemlerin çözümleri hakkında bilgi sahibi olabileceksiniz.

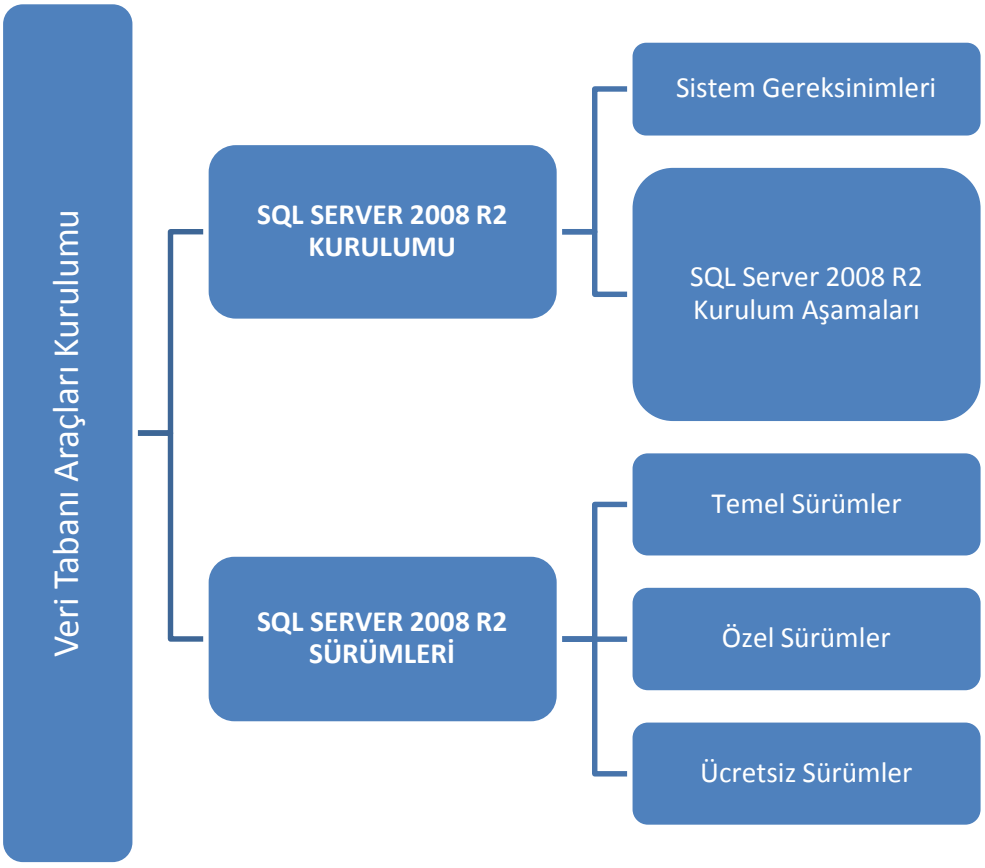
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi
Serdar AYDIN

ÜNİTE

3


Veri Tabanı Araçları Kurulumu
SQL SERVER 2008 R2 KURULUMU
Sistem Gereksinimleri
SQL Server 2008 R2 Kurulum Aşamaları
SQL SERVER 2008 R2 SÜRÜMLERİ
Temel Sürümler
Özel Sürümler
Ücretsiz Sürümler

GİRİŞ

Bu bölümde veri tabanı yönetim sistemi yazılımı kurulumunu ve veri tabanı yönetim araçlarının kurulumlarını yaparak veri tabanı programlamanın temel teoremleri hakkında fikir sahibi olacağız. Veri tabanı yönetim sistemleri kullanıcılarına veriyi oluşturma, güncelleme, yönetme, yedekleme gibi sistematik veri yönetimi imkân sağlar. Bu bölümde yaygın kullanımı, gelişmiş veri tabanı araçları ve yönetim sistemi nedeniyle SQL Server 2008 R2'nin kurulum aşamaları anlatılacaktır. SQL Server Windows Server 2008 ve üzeri sistemleri üzerinde çalıştırılabilen istemci/sunucu tabanlı bir ilişkisel veri tabanı yönetim sistemidir. Microsoft Backoffice ve .NET Enterprise Server ailesi içerisinde bir üründür. İ stemci/sunucu (client/server) iki ayrı bölüm olarak çalışan bir uygulamaya benzetilebilir: Birinci bölüm sunucu üzerinde çalışır, diğer bölüm de istemci üzerinde çalışır. Uygulamanın sunucu üzerinde çalışan kısmında güvenlik, yedekleme, depolama, performans gibi faaliyetler gerçekleştirilir. İstemci tarafında ise kullanıcı ara yüzü ile sunum, raporlama veya görüntüleme sağlanır. Öncelikle SQL Server 2008 R2 yazılımının sürümlerini, bu sürümler arasındaki farkları öğrenecek projelerinize uygun, optimum veri tabanı sürümlerinin seçimini yapabileceksiniz.

Kurulum için asgari donanım ve yazılım gereksinimleri hakkında bilgi sahibi olacak, eksik yazılımların nasıl yükleneceğini öğreneceksiniz.

SQL Server 2008 R2 kurulumundaki seçeneklerin her birini tek tek ele alacağımız bu bölümde kurulumun ilk adımından son adımına kadar resimli anlatımla veri tabanı yazılımı içermeyen bir bilgisayara veri tabanı sunucusunun kurulumunu yapıp ihtiyaç duyabileceğimiz veri tabanı servislerinin kurulumları hakkında bilgi sahibi olacak, ardından veri tabanına erişecek kullanıcı ayarlarını yapacak ve ihtiyacımız olan veri tabanı yönetim araçlarının kurulmasını sağlayacağız. Veri tabanı oluşturma, tablo oluşturma gibi işlemleri sonraki bölümlerde ele alacağız.

SQL SERVER 2008 R2 SÜRÜMLERİ

Temel Sürümler

Kurulum için SQL SERVER Standard'ı kullanacağız.

SQL Server 2008 Enterprise: SQL Server 2008 Enterprise, kurumsal çevrimiçi işlem ve veri ambarı uygulamalarının yüksek taleplerini karşılayan kapsamlı bir veri platformudur.

SQL Server 2008 Standard: SQL Server 2008 Standard departman uygulamalarını çalıştırmak için sınıfının en iyisi kullanım ve yönetim kolaylığı sağlayan tam bir veri yönetimi ve kurumsal zeka platformudur.

Özel Sürümler

SQL Server 2008 Workgroup: Şubeleri, güvenli uzaktan eşitleme ve yönetim özellikleri sağlayan bu güvenilir veri yönetimi ve raporlama platformunda çalıştırın.

SQL Server 2008 Web: Yüksek kullanılabilir Internet tarafı web hizmeti ortamlarıyla yüksek düşük maliyetli, yüksek ölçekli, yüksek kullanılabilir web uygulamaları veya veri barındırma çözümleri sağlayın.

SQL Server 2008 Developer: Sadece geliştirme, test ve gösterim amacıyla geliştirici başına lisans verilen düşük maliyetli SQL Server 2008 Enterprise sürümüdür. Üretim kullanımı için değildir.

Ücretsiz Sürümler

SQL Server 2008 Express: Ücretsiz olarak yükleyebileceğiniz SQL Server 2008 Express, masaüstü ve küçük sunucu uygulamalarını öğrenmek ve oluşturmak ve ISV'ler tarafından dağıtım için idealdir.

SQL Server Compact 3.5: Ücretsiz olarak yüklenebilen SQL Server Compact, geliştiricilerin SQL Server'ı doğrudan uygulamalarına dahil etmesini sağlar ve tüm Microsoft Windows platformlarındaki mobil aygıtlar, masaüstü sistemleri ve web istemcileri için arada bir bağlanan ve bağımsız uygulamalar sağlar.

Tablo 3.1. Senaryolara Uygun Sürüm Seçimi

	Temel Sürümler		**Özel Sürümler**			**Ücretsiz Sürümler**	
Sürüm	Enterprise	Standard	Çalışma Grubu	Web	Geliştirici	Express	Compact 3.5
Hedef Senaryolar	Yedekleme ve dahili Kurumsal Zeka gerektiren kurumsal iş yükleri	Departmanlar ve küçük-orta ölçekli işletmelerdeki paylaşılan veri senaryoları	Şirket verilerinin yerel örneklerine ihtiyaç duyan uzaktaki ofisler	Web uygulaması barındırma için	Sadece geliştirme ve test için tam özellikli sürüm	Öğrenim ve ISV dağıtımı için ideal olan giriş düzeyi veri tabanı	Masaüstü ve mobil uygulamalar geliştirmek için yerleşik veri tabanı
İşlemci	Sınırsız	4 CPU	2 CPU	4 CPU	Sınırsız	1 CPU	Sınırsız
Bellek	Sınırsız	Sınırsız	4 GB	Sınırsız	Sınırsız	1 GB	Sınırsız
Veri Tabanı Boyutu	Sınırsız	Sınırsız	Sınırsız	Sınırsız	Sınırsız	4 GB	4 GB

SQL SERVER 2008 R2 KURULUMU

Sistem Gereksinimleri

SQL Server 2008 R2 kurulumuna başlamadan önce aşağıdaki noktalara dikkat etmeniz gerekmektedir:

- NTFS dosya sistemi kullanın.
- Sıkıştırılmış sürücüye kurulum yapmaya çalışmayın. Kurulum sonlandırılacaktır.
- Güvenlik duvarınızı SQL Server'a erişime izin verecek şekilde ayarlayın.
- SQL Server kurulumu yapacak kullanıcı tam yönetici (administrator) haklarına sahip olmalıdır.
- Windows management instrumentation servisinin çalışmakta olduğuna emin olun.
- Kurulumun yapılacağı bilgisayarın internete bağlı olduğundan emin olun.
- SQL Server 2008 R2'yi SQL Server 7 ile birlikte kurmayın çünkü bu desteklenmemektedir.

Tablo 3.2.de SQL server sürümleri için tavsiye edilen minimum donanım gereksinimleri ve Tablo 3.3.te ise disk alanı gereksinimleri

Veri tabanı büyüklüğüne bağlı olarak SQL Server 2008 donanımsal gereksinimler aratabilir.

Tablo 3.2. Donanım Gereksinimleri

Bileşen	Gereksinim
Bellek	**Minimum:** • Express Sürümler: 256 MB • Diğer Tüm Sürümler: 1 GB **Tavsiye Edilen:** • Express Sürümler: 1 GB • Diğer Tüm Sürümler: Optimum performansı yakalayabilmek için artırılabilir en az 4GB ram
İşlemci Hızı	**Minimum:** • x86 İşlemci: 1.0 GHz • x64 İşlemci: 1.4 GHz **Tavsiye edilen:** • 2.0 GHz veya üzeri
İşlemci Tipi	• x64 İşlemci: AMD Opteron, AMD Athlon 64, Intel Xeon (Intel EM64T destekli), Intel Pentium IV (EM64T destekli) • x86 İşlemci: Pentium III-uyumu işlemci veya daha üzeri

Tablo 3.3. Disk alanı gereksinimleri

Özellik	Disk Alanı Gereksinimi
Veri tabanı motoru ve veri dosyaları, aynalama ve Full-Text Arama	711 MB
Analysis Service ve veri dosyaları	345 MB
Reporting Services ve Report Manager	304 MB
Integration Services	591 MB
İstemci bileşenleri (Integration Services araçları ve Kitaplar hariç)	1823 MB
SQL Server çevrim içi kitapları	157 MB

SQL Server 2008 R2 kurulabilmesi için bilgisayarlarınızda hâlihazırda kurulu olması gereken yazılımlar:

- .Net Framework 3.5 SP1 kurulu olmalıdır.
- Windows Installer 4.5 veya üst sürümleri gereklidir.
- Internet Explorer 6 SP1 veya üst sürümleri gereklidir.
- Advanced Services ile SQL Server Express için PowerShell gereklidir.
- Eğer bilgisayarınızda Visual Studio 2008 var ise Visual Studio 2008 SP1'e yükseltmeniz gerekmektedir.
- Desteklenen İşletim Sistemleri: Windows Server 2012, Windows Server 2008, Windows Server 2003 Service Pack 2, Windows Server 2003 Small Business Server R2, Windows Vista, Windows XP Service Pack 2.

SQL Server 2008 R2 Kurulumu

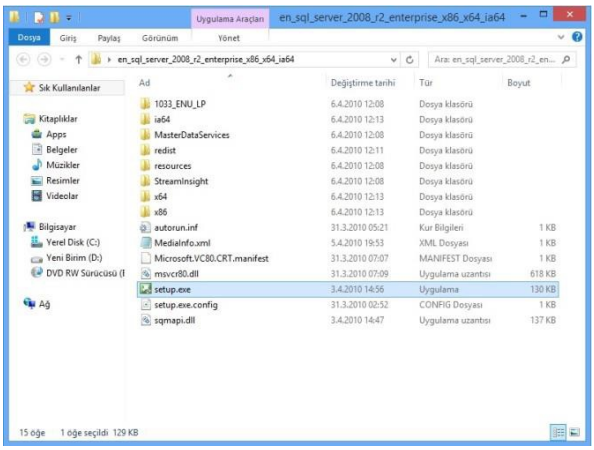
Kurulum başlatılması

Kurulumu "setup.exe" dosyasını çalıştırarak başlatıyoruz.

Kurulum DVD'sini taktığınızda kurma programını çalıştırmanız önerilecektir.

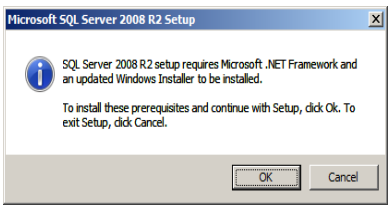
Şekil 3.1. Otomatik çalıştırma

DVD'den değil de yerel diskinizden kurulum yapmak istiyorsanız kurulum dosyalarının bulunduğu dizine gidip setup.exe'i çalıştırmanız gerekmektedir.



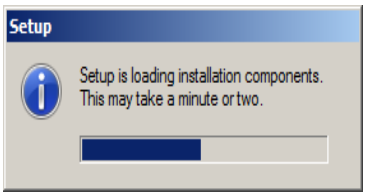
Şekil 3.2. SQL Server dosyalarının bulunduğu dizin

Biz otomatik olarak gelen bu ekranda sistem üzerine SQL 2008 R2 kurulumunu başlatmak için Setup.exe dosyasını çalıştırmak için tıklıyoruz. Bilgisayarımızda eğer .Net Framework 3.5 SP1 ve Windows Installer daha önce kurulu değil ise karşımıza SQL 2008 R2 kurulumu için eğer.NET Framework ve Windows Installer bileşenlerinin kurulumu için bir uyarı penceresi gelir.

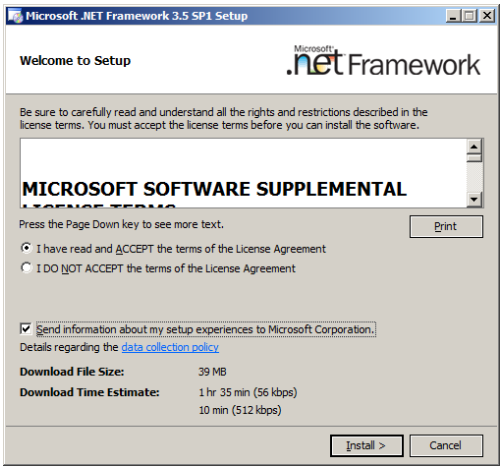


Şekil 3.3. .Net Framework ve Windows Installer gereksinim uyarısı

Yukarıda belirttiğimiz gibi SQL Server 2008 R2 kurabilmemiz için Windows Installer 4.5 ve .Net Framewrk 3.5 SP1'in hâlihazırda kurulu olması gerekmektedir. Ok butonunu tıklayarak .Net Framework kurulumunu başlatalım.



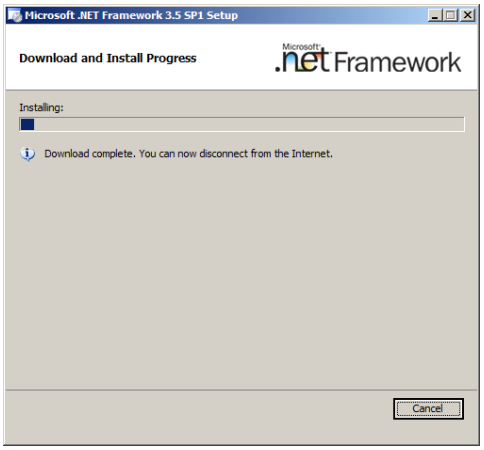
Şekil 3.4. Kurulum bileşenleri toplanma bildirimi



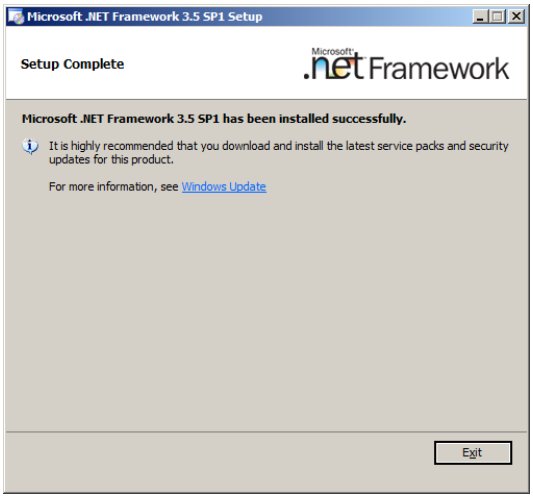
Şekil 3.5. . Net Framework için Lisans anlaşması kabul ekranı

Yazılımsal gereksinimler tamamlanmadan SQL Server 2008 kurulumu başlatılamaz.

Lisans anlaşmasını kabul edip INSTALL butonunu tıklıyoruz.



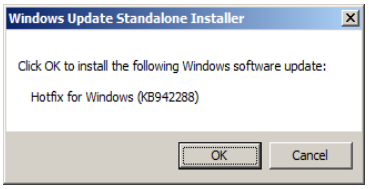
Şekil 3.6. .Net Framework internetten bilgisayarınıza indiriliyor



Şekil 3.7. . Net Framework kurulumu başarı ile tamamlandı

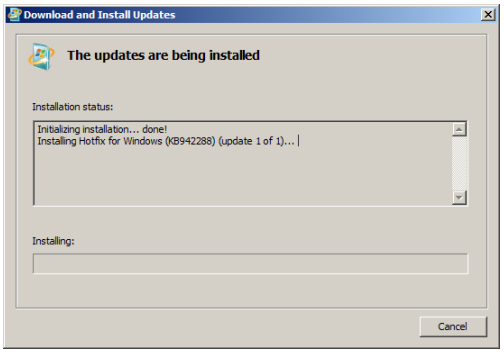
SQL Server 2008 kurulumu için sistemde .Net framework 3.5 kurulu olmalıdır.

Kurulum, .Net Framework 3.5 SP1 kurulumu ardından, eğer kurulu değil ise Windows Installer (KB942288) kurulumuna başlamanızı önerecektir.

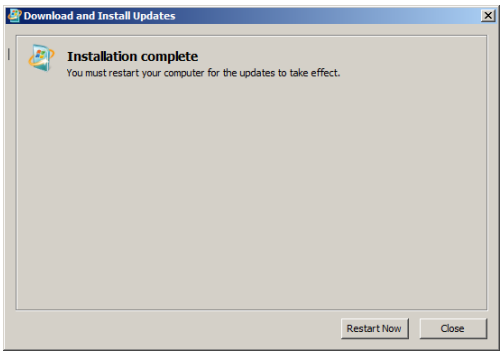


Şekil 3.8. Windows Installer kurulum onayı

Windows Installer (KB942288) kurulumuna başlamak için OK butonunu tıklıyoruz.



Şekil 3.9. Windows Installer kuruluyor



Şekil 3.10. Windows Installer kurulumu tamamlanıyor

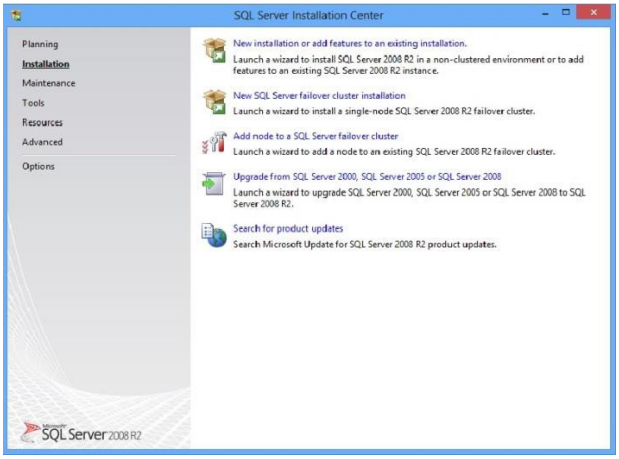
SQL Server 2008 R2 kurulumuna sorunsuz olarak devam edebilmek için *Restart Now* butonuna tıklıyoruz ve bilgisayarımız yeniden başlatılıyor. Hotfix kurulumu sonrasında, SQL Server kurulum yardımcısı çalışacaktır.

Ön yazılımsal gereksinimler yüklendikten sonra Bilgisayar yeniden başlatılmalıdır.

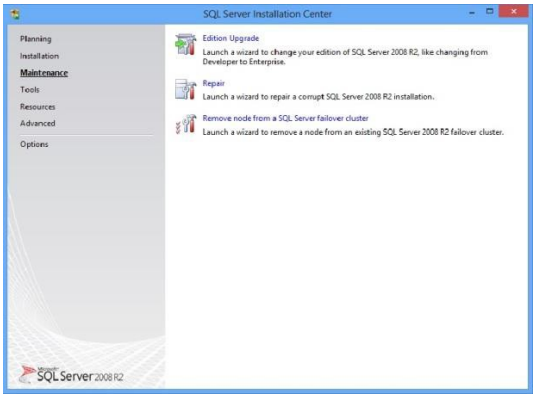
Şekil 3.11. SQL Server Installation Center Planning Sekmesi

Kurulumun ilk aşamasında karşımıza SQL Server Installation Center ekranı gelecektir.

Bu ekranda SQL Server 2008 R2 ile ilgili çevrim içi/çevrim dışı dokümanlara ulaşmanız için gerekli bağlantıları göreceksiniz. Sol tarafta farklı kategoriler sıralanmıştır. Planning kategorisinde SQL 2008 R2'nin kurulumuna hazırlık için kullanacağınız dokümanlar ve araçları bulabilirsiniz.



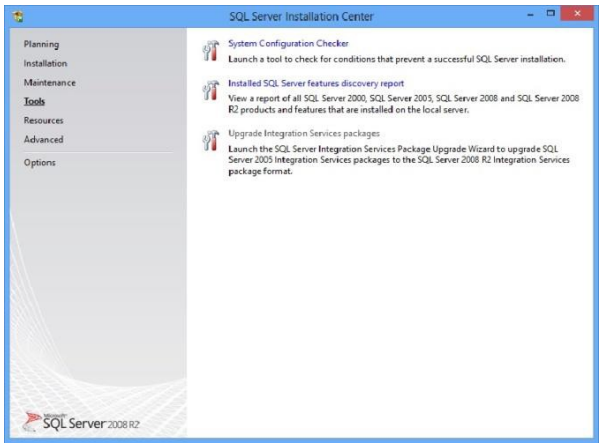
Şekil 3.12. SQL Server Installation Center Installation Sekmesi



Şekil 3.13. SQL Server Installation Center Maintance Sekmesi

Maintenance kategorisinde SQL 2008 R2 sürümlerine yükseltme, SQL 2008 R2'in tamir edilmesi ve cluster yapısında çalışan node'lar üzerinden SQL 2008 R2'i kaldırmakla ilgili linklere ulaşabilirsiniz.

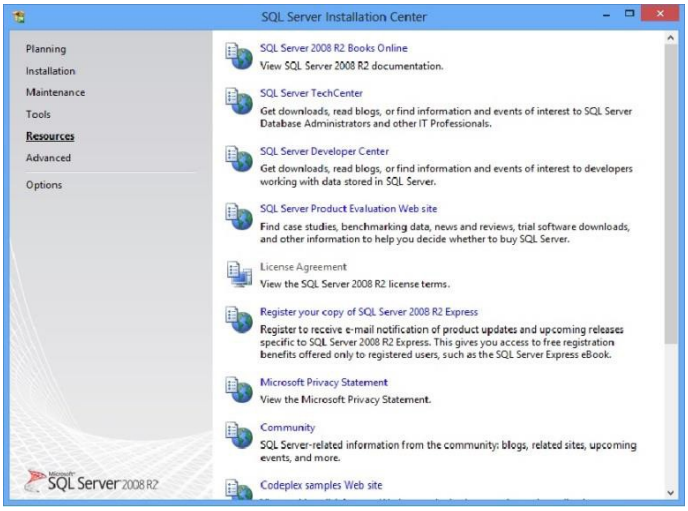
SQL Server 2008 tamiri ve sistemden kaldırılması işlemlerini yine setup.exe dosyasını çalıştırarak yapabilirsiniz.



Şekil 3.14. SQL Server Installation Center Tools Sekmesi

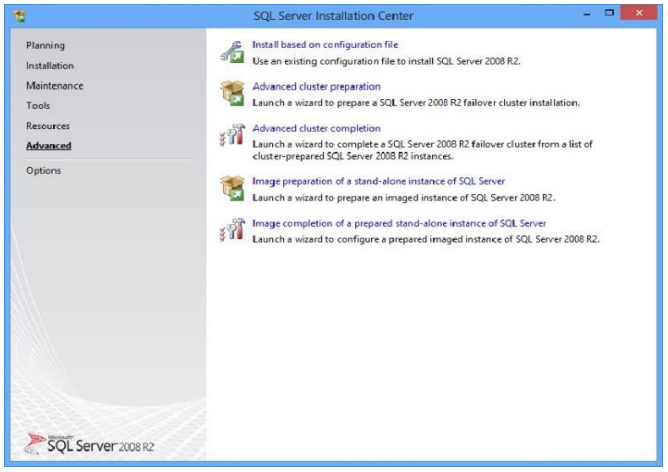
Tools bölmesinde SQL Server 2008 R2 için geliştirilmiş olan araçları gerek kurulum öncesi gerekse de kurulum sonrası farklı ihtiyaçlarınız için kullanabilir Örneğin, System Configuration Check aracı ile mevcut sisteminizin SQL Server

2008 R2 kurulumu için hazır olup olmadığını test edebilirsiniz.



Şekil 3.15. SQL Server Installation Center Recources Sekmesi

Resources kategorisinde SQL Server 2008 R2 ile yardım dosyalarına, doküman kütüphanesine ve online kaynak sayfalarına ulaşabileceğiniz linkleri göreceksiniz.

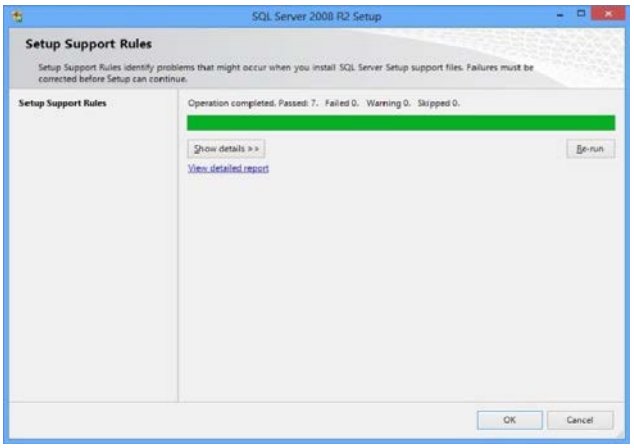


Şekil 3.16 SQL Server Installation Center Advanced Sekmesi

Gelişmiş SQL Server kurulum seçenekleri "Advanced" bölümünde yer alır.

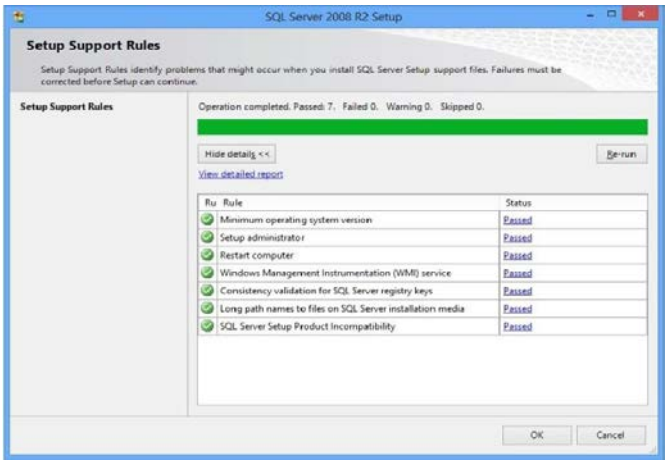
Advanced kategorisinde SQL Server 2008 R2 gelişmiş kurulum seçeneklerini bulabilirsiniz. SQL Server kurulum ayar dosyası temel alınarak otomatik kurulum yaptırma, gelişmiş cluster hazırlama, hazır clusterları çalıştırma, başka bilgisayarlara kolay kurulum sağlayabilmek için kurulum imajı oluşturma veya oluşturulmuş imaj ile kurulumu tamamlama için kullanılan araçları bulabilirsiniz.

Biz yeni ve sade bir kurulum yapmak için Şekil 2.1.de yer alan *New Installation or add features to an existing installation* bağlantısını tıklıyoruz. Kur, bu noktadan sonra sistemimizi yazılım ve ayar gereksinimlerine karşı tarayacaktır.



Şekil 3.17. SQL Server Kur, gereksinim taraması

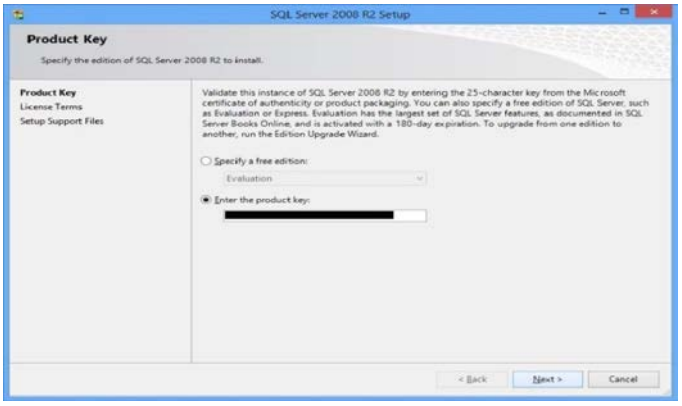
Bu taramanın sonuçlarını görmek için Şekil 2.5.te yer alan Show details butonunu tıklayabilirsiniz.



Şekil 3.18 SQL Server Kur, gereksinim taraması detaylı görünüm

Kuruluma devam edebilmek için sistemin SQL Server gereksinimlerini sağlaması gerekir.

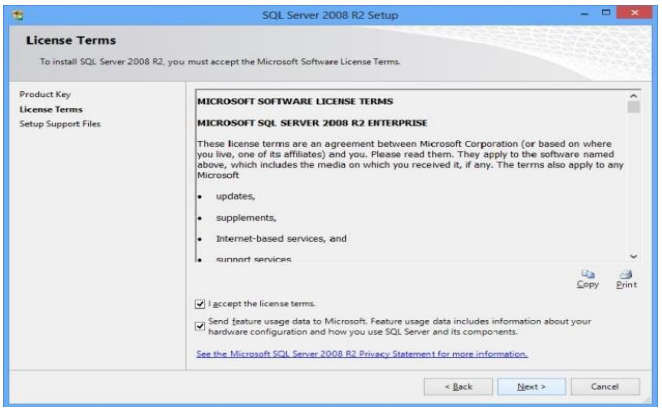
Varsa eksik gereksinimleri tamamladıktan sonra *Re-Run* butonunu tıklayarak kurulumun sistemi tekrar test etmesini sağlayabilirsiniz. *Failed* olarak işaretlenmiş gereksinim yok ise kurulumunuza OK butonunu tıklayarak devam ediyoruz.



Şekil 3.19. SQL Server Kur ürün anahtarı ekranı

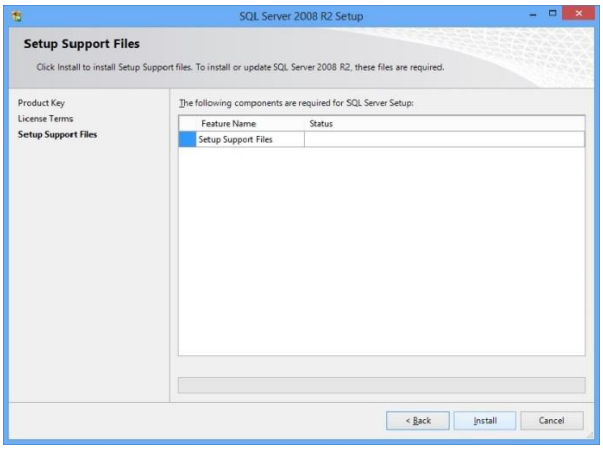
Ücretsiz sürümler için lisans anahtarına gerek yoktur.

Karşımıza Product Key ekranı gelecektir. Bu ekranda ürün anahtarınızı girmeniz gerekmektedir eğer ürün anahtarına sahip değilseniz. SQL Server kurulumuna Free Edition (Ücretsiz Sürüm) seçerek devam edebilirsiniz.



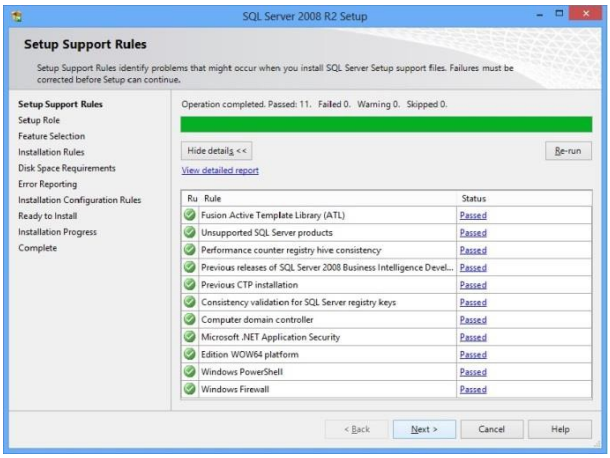
Şekil 3.20 SQL Server için Lisans anlaşması ekranı ve Microsoft Geri besleme kayıt ekranı

I accept the license terms ile kabul ediyoruz. Bu aşamada lisans anlaşmasının Print ile çıktısını alabilir, Copy ile de kopyalayabilirsiniz. Biz Next ile bir sonraki aşamaya geçiyoruz.



Şekil 3.21 SQL Server kurulumu için gerekli dosyaların sisteme yüklenmesi

Karşımıza *Setup Support Files* penceresi gelir ve bize SQL 2008 R2 kurulumuna devam etmemiz için Kurulum yardım dosyalarının öncelikle kurulması gerektiği uyarısı gelir. *Install* butonuna tıklayarak SQL Server 2008 R2 kurulum destek dosyalarının yüklenmesi ve ayarlanması işlemini başlatıyoruz.



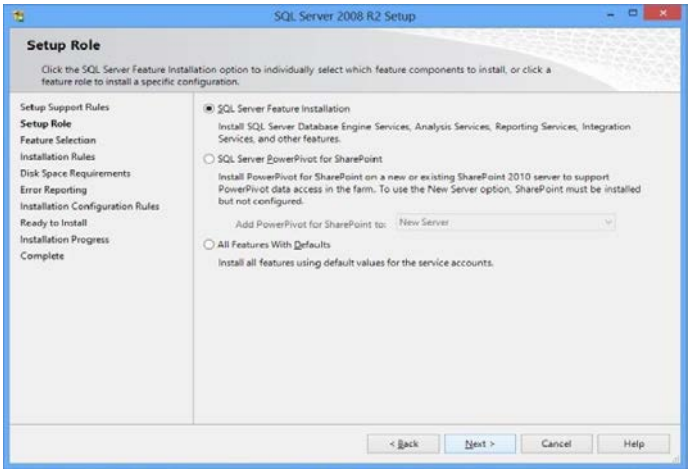
Şekil 3.22. SQL Server kurulumu için gerekli işletim sistemi ayarlarının kontrolü

SQL Server'ın sorunsuz çalışabilmesi için güvenlik duvarında gerekli izinlerin verilmesi gerekir.

Burada System Configuration Check aracı çalışarak bize sistemimizdeki başarılı, başarısız ya da uyarı içeren bilgiler gelecektir. NOT: Windows'da firew aktif ise Windows Firewall ve IIS 7.0'da varsayılan uygulama güvenlik ayarların

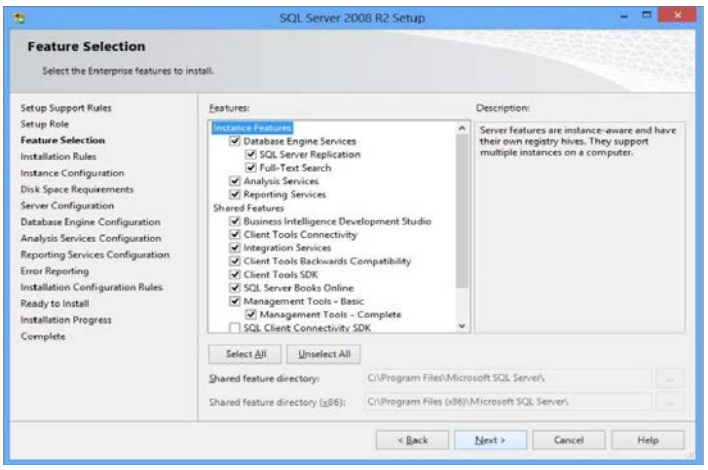
dolayı Microsoft .NET Application Security Warning yani uyarı gelebilir.

Bu uyarılar SQL Server 2008 R2 kurulumu için herhangi bir engel teşkil etmeyeceği için *Next* ile kuruluma devam edebilirsiniz ancak SQL Server'ınıza dışardan erişim izni vermek istiyorsanız Güvenlik duvarı ayarlarınızı buna göre tekrar yapılandırmanız gerekir.



Şekil 3.23. SQL Server kurulum modu seçimi

Bu ekranda Sql Serverınızın varsayılan ayarlarla kurulmasını veya PowerPivot modunda kurulmasını veya adım adım sizin belirleyeceğiniz ayarlarla kurulmasını sağlayabilirsiniz. Biz bu ekrandaki *SQL Server Feature Installation* seçeneği kullanarak kurulumumuza devam edeceğiz. Bu seçenek; SQL Server veri tabanı motoru, Analiz servisleri, Rapor servisleri, Entegrasyon servisleri ve diğer özellikleri yükleyebileceğimiz seçenektir.



Şekil 3.24. SQL Server kurulumu gelişmiş özellik seçim ekranı

Her bir SQL kurulumuna SQL Instance diyeceğiz.

Karşımıza *Feature Selection* ekranı gelecektir. Bu ekranda *Instance Features* altındaki seçenekler bu bilgisayara kurulacak tüm SQL Server için ayrı ayrı olan bileşenleri ve servisleri içerir. Kurulan her SQL Server için Database Engine, Analysis Services ve Reporting Services bileşenlerinin kurulup kurulmayacağına bu noktada karar veriyoruz. *Shared Features* altındaki seçenekler bu bilgisayara kurulacak tüm SQL kurulumları için ortak kullanılacak bileşenleri ve servisleri içerir. Örneğin Management Tools seçeneği ile kurulan SQL Server 2008 yönetim araçları bütün farklı SQL kurulumları için ortak kullanılan bir bileşendir. Yani her kurula SQL kurulumu için farklı bir Management Tool kurulmaz.

Şimdi de bu ekranda karşımıza gelen bileşenleri genel olarak açıklayalım:

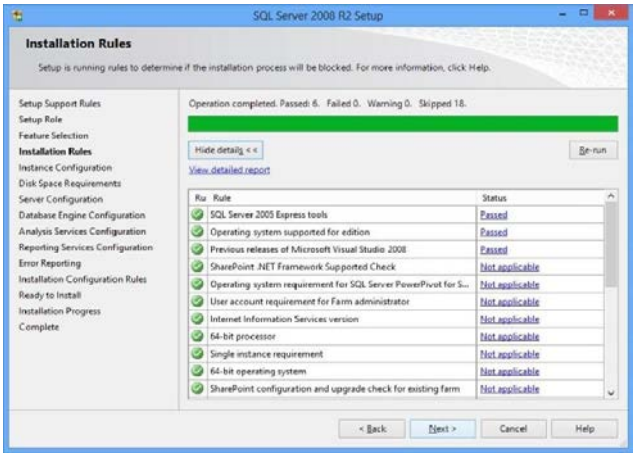
SQL Server Database Engine Services: SQL Server veri tabanı servislerini kurmayı sağlar.

Analysis Services : SQL Server üzerinde OLAP küp tasarımları ve veri madenciliği (data mining) uygulamaları geliştirmeyi kolaylaştıran SQL analiz servislerinin kurulumunu sağlar.

Reporting Services: SQL Server üzerinde raporlama uygulamaları geliştirmek için kullanılır.

Management Tools: Bilgisayarınıza SQL Server 2008 R2 yönetim araçlarını kurmak için bu seçenek kullanılır. Böylece hem kendi bilgisayarınızda çalışan uygulamalara hem de ağınızda çalışan diğer bilgisayarlara bu seçenek ile SQL Server 2008 R2 yönetim ve geliştirme araçlarını kurup bağlanarak SQL Server 2008 yönetimini yapabilirsiniz.

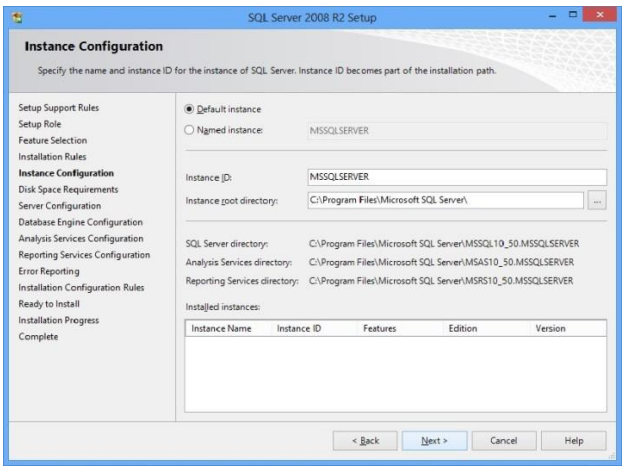
Shared Feature Directory, kısmından SQL Server 2008 R2'nin Shared Features kategorisi altında bulunan ve tüm kurulu SQL kurulumlarının ortak olarak kullandığı bileşenlerin yükleneceği konum belirlenir. Biz bütün bileşenleri kuracağımız için *Select All* tıklıyoruz ve *Next* ile bir sonraki adıma geçiyoruz.



Şekil 3.25. SQL Server kurulumu seçilen ekstra özellikler için sistem gereksinim test ekranı

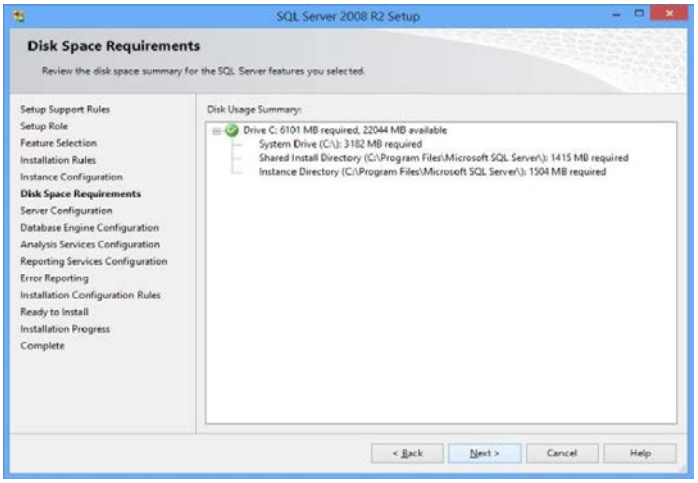
Next butonuna basıp devam ediyoruz.

SQL kurulum klasörünü bu ekrandan değiştirebilirsiniz.



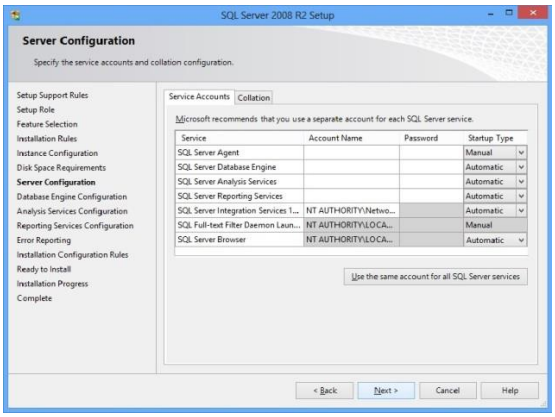
Şekil 3.26. SQL Server kurulumu seçilen ekstra özellikler için sistem gereksinim test ekranı

Karşımıza Instance Configuration ekranı gelecektir. Bu ekranda SQL Server adı yani Instance Name, Instance ID ve kurulan Instance'a ait veri ve log dosyalarının kaydedileceği Root dizininin yeri belirlenir. Şekilde de görüldüğü gibi bilgisayarınıza ilk SQL Server kurulumunda otomatik olarak Default seçeneği aktif olarak gelir. Bu seçeneği kullanarak SQL Server kurulumu yaptığınız zaman kullanılan Instance adı MSSQLServer içindir. (Eğer SQL Server Express Edition kuruyorsanız, Default Instance adı SQLExpress olacaktır.) Ve SQL Server bağlantı adınız bilgisayar adınız ile aynı olacaktır. Aynı bilgisayara birden fazla SQL server kurulumu yapılması gerektiğinde *Instance ID farklı* olmalıdır.



Şekil 3.27. SQL Server kurulumu için gerekli disk alanı kontrol ekranı

Karşımıza Disk Space Requirement ekranı gelir. Sistemimizdeki disk alanını kontrol ederek yeterli olup olmadığı bilgisini getirir. *Next* ile bir sonraki adıma geçiyoruz.



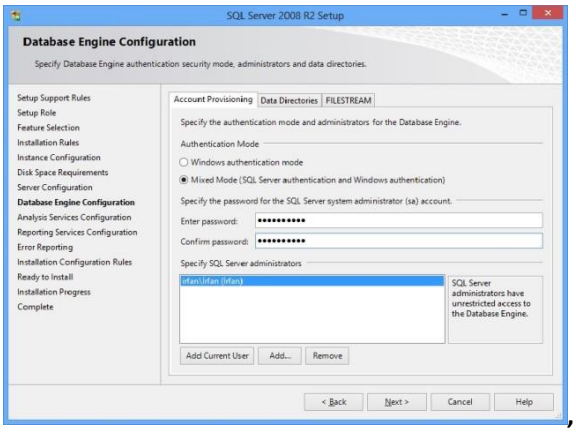
Şekil 3.28. SQL Server'ın çalışacağı servis hesapları ve veri tabanı sıralama ayar ekranı.

SQL Serveri çalıştıracak kullanıcıyı buradan belirliyoruz.

Karşımıza Server Configuration ekranı gelecektir. Yukarıdaki resimde görülen *Service Accounts* tabında SQL Server 2008 R2 Servislerini çalıştıracak kullanıcı hesapları belirlenir. SQL Server 2008 ile ilgili temelde iki önemli servis bulunmaktadır. Bunlar *SQL Server Database Engine* servisi ve *SQL Server Agent* servisidir. SQL Server Database Engine servisi bilgisayarınızın veri tabanı sunucusu olarak faaliyet göstermesini, kendisine gelen istekleri, sorguları alıp cevaplamasını sağlayan servistir. Ve SQL Server bilgisayarının hizmet vermesi için SQL Server Database Engine servisinin mutlaka çalışması gerekir. SQL Server Agent servisi ise SQL Server üzerinde job, alert ve operator gibi yönetimsel görevleri tanımlama

için kullanılır. Bu ekranda tüm SQL servisleri için aynı kullanıcı hesabını mı, yoksa farklı kullanıcı hesabını mı kullanacağımız belirlenir. Eğer aynı hesabı kullanacaksanız *Use the same account for all SQL Services* butonuna basılmalıdır. Böylece hem SQL Server hem de SQL Server Agent vb. diğer servisler aynı kullanıcı hesabı üzerinden çalıştırılacaktır.

Sıralama düzeni, karakter setini, sorgulanan karakter verisi ile karşılaştırır. Böylece sorgunun sonucunda dönecek olan verileri etkilemiş olur. Aynı zamanda performansı da etkilemiş olur. Örneğin, büyük küçük harf ayrımı olmayan bir sıralama düzeninde daha az karşılaştırma yapılır. Bu durum performansı artırır.



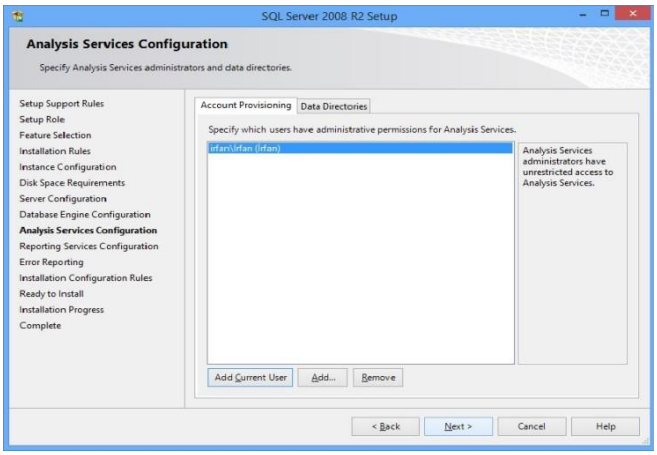
Şekil 3.29. SQL Server kimlik denetimi ayarları

Bu adımda yetkilendirme modu seçilmektedir. Yetkilendirme türünü belirlemekle kurduğunuz SQL Server güvenliğini belirlemiş olursunuz. Bunun için, Windows Authentication Mode veya Mixed Mode seçenekleri vardır.

- *Windows Authentication Mode:* Windows kimlik denetiminden geçen kullanıcılara, SQL Server 2008'e erişme hakkı tanınır. Erişim, Windows kullanıcı güvenlik sınırları içinde gerçekleşir.
- *Mixed Mode:* Normal Windows kullanıcısı haklarıyla veya "sa" kullanıcı adı ve kendi belirleyeceğimiz şifre ile yönetici haklarına sahip olarak veri tabanları üzerinde işlem yapabilmek için bu mod seçilir. Mixed Mode seçildiğinde, varsayılan sistem yöneticisi hesabı olan "sa" kullanıcısına şifre verilerek güvenlik sağlanır. Şifre boş bırakılabilir ama tabi ki güvenlik açısından tavsiye edilmez.

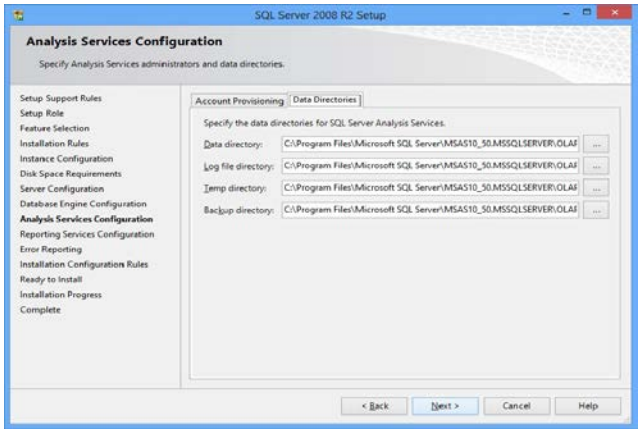
Gelen ekranda *Mixed Mode* seçeneğini işaretleyerek bir şifre giriyoruz. Daha sonra aşağıdaki *Add Current User* butonuna basarak mevcut Windows kullanıcısının Sql Server yöneticisi olarak atanmasını sağlıyoruz. *Next* butonuna basarak yolumuza devam ediyoruz.

İstemcilerin SQL server' a bağlanılabilmesi için gerekli kullanıcı adı ve şifre ayarlarını buradan yapıyoruz.



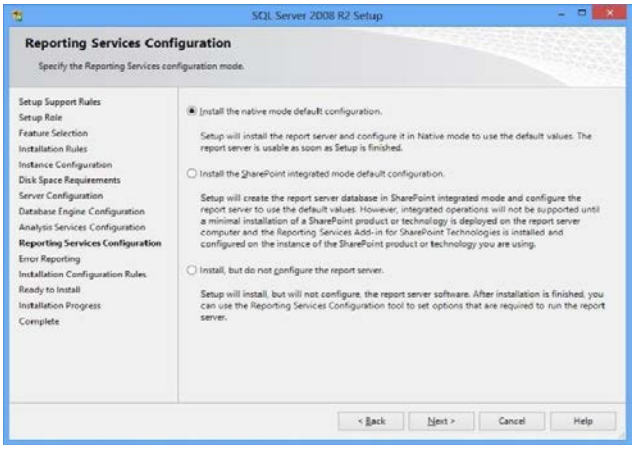
Şekil 3.30. SQL Server Analysis Servislerine erişebilecek kullanıcı ayarları

Bu gelen ekrandan da Add Current User butonuna basarak Analysis Services'e erişebilecek kullanıcılar arasına oturum açtığımız kullanıcıyı ekliyoruz. *Next* butonuna basarak yolumuza devam ediyoruz.

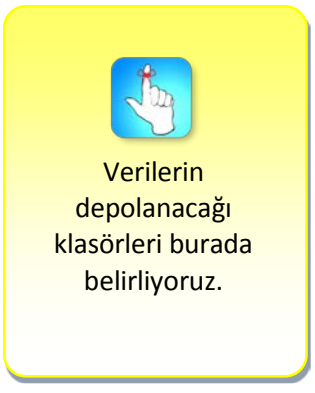


Şekil 3.31. SQL Server Analysis Servisi Veri dizini ayarları

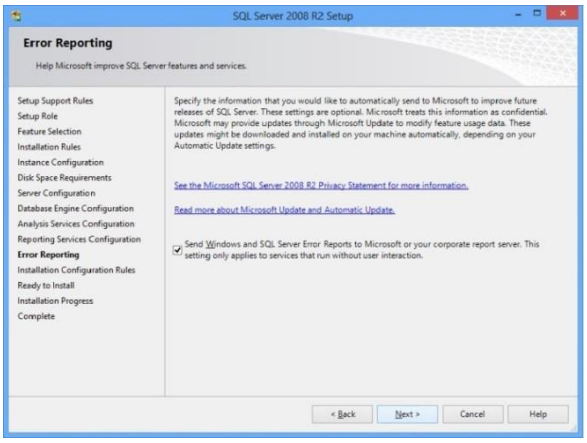
Bu ekranda veri dosyaları, Log dosyaları, geçici dosyaları ve backup dosyalarının nereye kaydedileceğini gösterir. *Next* butonuna basıp devam ediyoruz.



Şekil 3.32. SQL Server Reporting Servisleri çalışma modu belirlenmesi.

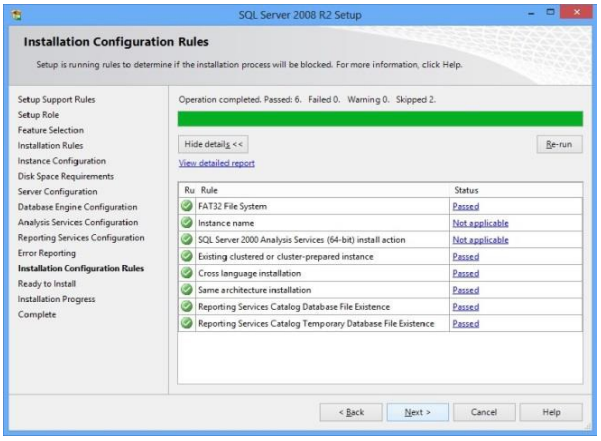


Reporting servislerinin modunu belirlediğimiz bu aşamada varsayılan ay kullanarak devam ediyoruz.

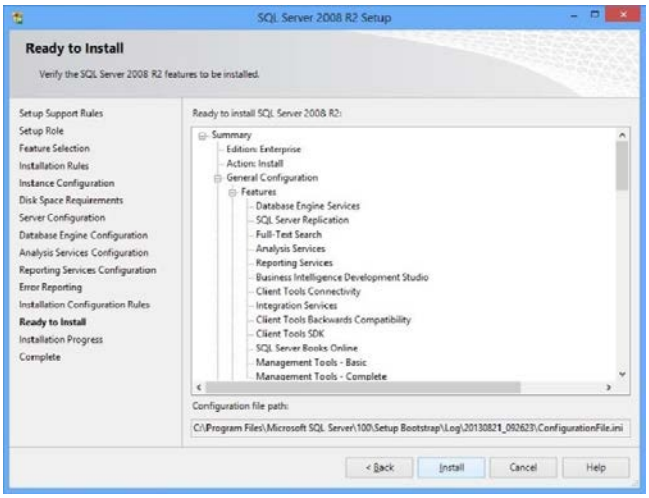


Şekil 3.33. SQL Server hata raporlama ayar ekranı.

Error Reporting (Hata Raporlama) ekranındaki kutucuk işaretlenirse *"SQL Server 2008 R2"* ile ilgili hatalar *"Microsoft"*a raporlanacaktır. Bu kutucukları işaretleyip işaretlememek kişisel bir tercihtir.



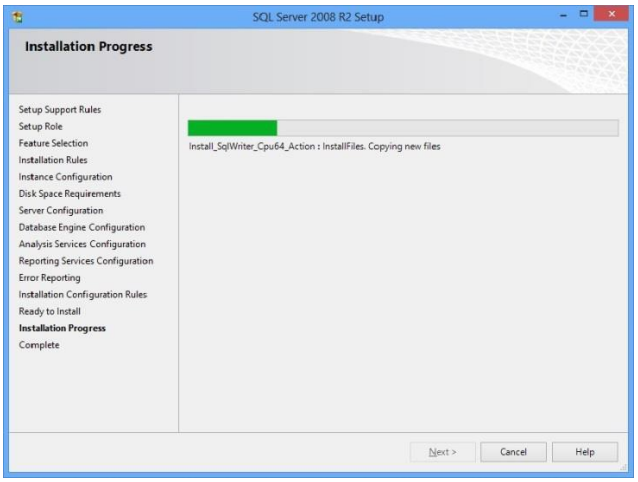
Şekil 3.34. Yaptığınız düzenlemeler sonrası sistem gereksinimlerinin tekrar kontrol edilmesi



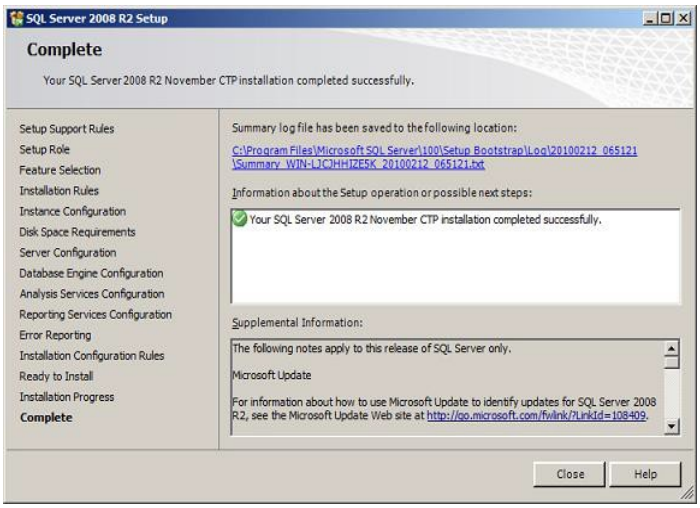
Şekil 3.35. SQL Server kurulum seçenekleri gözden geçirme

SQL Server kurulumu için son kez sistem uyumluluğu burada denetleniyor.

Bu ekranda Sql Server 2008 R2 kurulumu için özellikleri ve bileşenleri gösterir. Devam etmek için *Install* seçeneğini tıklayıp kuruluma devam ediyoruz.



Şekil 3.36 SQL Server kuruluyor



Şekil 3.37. SQL Server kurulumu başarı ile tamamlandı

Bu ekran SQL Server 2008 R2'nin başarılı bir şekilde yüklendiğini göstermektedir.

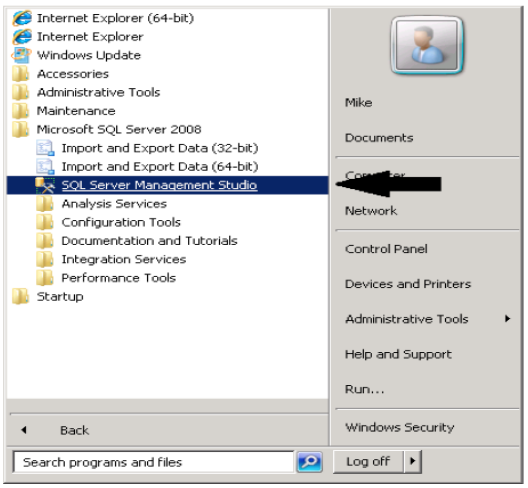
Bireysel Etkinlik

- SQL Server 2008 R2 kurarken karşılabileceğiniz muhtemel hataları ve bunların çözümlerini araştırınız.
- SQL Server 2008 R2'nin diğer veritabanlarına göre avantajlarını ve dezavantajlarını araştırınız.
- SQL Server 2008 R2 Sürümleri arasındaki farkları tespit ediniz.

SQL Server'a ilk bağlantıyı Management Studio ile yapıyoruz.

Başlama noktası

Sırasıyla Başlat (Start), Tüm Programlar (All Programs), Microsoft SQL Server 2005, SQL Server Management Studio yolunu izleyerek programı başlatabiliriz.



Şekil 3.38 SQL Server Management Studio

Veri tabanı motoruna bağlanma

Aşağıdaki ekrandaki giriş değerleri sırasıyla şunları ifade eder:

- Server type: Sunucu türü demektir. "Database Engine" olarak seçiniz.
- Server name: Sunucu ismi demektir. Kurulum sırasında verdiğimiz ismi kullanmalıyız.
- Authentication: Yetkilendirme demektir. Hangi yetkilendirme türüyle veri tabanı motoruna erişeceğimizi belirtiriz.
- Login: Kullanıcı ismi demektir. Sistem yöneticisi iseniz "sa" yazarak değilseniz sistem yöneticinizin sizin için belirlediği kullanıcı ismini yazınız.
- Password: Sistem yöneticisi iseniz kurulum sırasında belirlediğiniz şifreyi giriniz, değilseniz sistem yöneticisinin size verdiği şifreyi kullanınız.

Bu bilgiler girildikten sonra *connect* düğmesine basarak isteme giriş yapabilirsiniz.

Şekil 3.39. SQL Server Management Studio SQL Server ile bağlantı kurma

Kurulum esnasında oluşturduğumuz kullanıcı adı ve şifreyi burada kullanıyoruz.

Bireysel Etkinlik

- Bu bölümdeki kurulum aşamalarının takip ederek siz de bilgisayarınıza SQL Server 2008 R2 kurunuz.
- SQL Server Management Studio ile kurduğunuz veri tabanı motoruna, kendi belirlediğiniz kullanıcı ve şifre ile bağlanınız.

Özet

- **SQL SEVER 2008 R2 KURULUMU**
- Bilgisayarımıza SQL 2008 R2 kurmak için aşağıdakı adımları izlememiz gerekir.
  - Öncelikle ihtiyacımıza uygun sürümü belirlemeliyiz.
  - **SQL SERVER 2008 R2 SÜRÜMLERİ**
  - Seçtiğimiz ürünü bilgisayarımız destekliyor mu kontrol etmeliyiz.
  - Kurulum için SQL Server 2008 R2 dosyalarının bulunduğu dizine gidip setup.exe yi çalıştırırız.
- Kurulum adımlarını izler kurulum için varsa eksik yazılım veya donanımı bilgisayarımıza ekleriz.
- Yeni kurulum için SQL Server Feature Installation bağlantısını tıklayarak kurulumu başlatırız.
- Kurulum adımlarını izlerken hangi SQL Server bileşenlerini kuracağımızı Instance Configuration ekranından seçeriz.
  - **SQL Server Database Engine Services:** SQL Server veri tabanı servislerini kurmayı sağlar.
  - **Analysis Services :** SQL Server üzerinde OLAP küp tasarımları ve veri madenciliği (data mining) uygulamaları geliştirmeyi kolaylaştıran SQL analiz servislerinin kurulumunu sağlar.
  - **Reporting Services:** SQL Server üzerinde raporlama uygulamaları geliştirmek için kullanılır.
  - **Management Tools:** Bilgisayarınıza SQL Server 2008 R2 yönetim araçlarını kurmak için bu seçenek kullanılır. Böylece hem kendi bilgisayarınızda çalışan uygulamalara hem de ağınızda çalışan diğer bilgisayarlara bu seçenek ile SQL Server 2008 R2 yönetim ve geliştirme araçlarını kurup, bağlanarak SQL Server 2008 yönetimini yapabilirsiniz.
  - **Shared Feature Directory, k**ısmından SQL Server 2008 R2'nin Shared Features kategorisi altında bulunan ve tüm kurulu SQL kurulumlarının ortak olarak kullandığı bileşenlerin yükleneceği konum belirlenir. Biz bütün bileşenleri kuracağımız için Select All tıklıyoruz ve Next ile bir sonraki adıma geçiyoruz.
- Kimlik denetimi ayarlarından SQ Server a bağlanabileceğimiz kullanıcı ayarlarını yaparız.
- SQL Server'ın çalışacağı veri ve kurulum dizinlerini belirleriz.
- Sonraki aşamada karşımıza Server Configuration ekranı gelecektir. Yukarıdaki resimde görülen Service Accounts tabında SQL Server 2008 R2 Servislerini çalıştıracak kullanıcı hesapları belirlenir. SQL Server 2008 ile ilgili temelde iki önemli servis bulunmaktadır. Bunlar SQL Server Database Engine servisi ve SQL Server Agent servisidir. SQL Server Database Engine servisi bilgisayarınızın veri tabanı sunucusu olarak faaliyet göstermesini, kendisine gelen istekleri, sorguları alıp cevaplamasını sağlayan servistir. Ve SQL Server bilgisayarının hizmet vermesi için SQL Server Database Engine servisinin mutlaka çalışması gerekir.
- Son olarak kurulum ayarlarımızı gözden geçirdikten sonra kurulumu başlatırız.
- Kurulum tamamlandıktan sonra SQL Management Studio ile kurduğumuz SQL server a bağlanabiliriz.

DEĞERLENDİRME SORULARI

1. SQL Server 2008 R2 kurulumu başlayabilmesi için aşağıdaki yazılımların hangileri sistemde kurulu olmalıdır?
   a) .Net Framework 3.5 ve Windows Installer
   b) .Net Framework 2.0 ve Windows Setup
   c) .Net Framework 4.0 ve SQL Server 2008 Engine
   d) .Net Framework 1.0 ve SQL Runtime
   e) .Net Framework 4.5 ve Windows Media Player

2. Aşağıdakilerin hangisi SQL Server bileşenlerinden biri değildir?
   a) Analysis Services
   b) Integration Services
   c) Database Engine
   d) Reporting Services
   e) Administrating Services

3. Aşağıdakilerden hangisi SQL Server 2008 Yönetici hesabı varsayılan kullanıcı adıdır?
   a) Admin
   b) Administrator
   c) Sa
   d) Root
   e) Sysadmin
4. Aşağıdakilerden hangisi SQL Server 2008 kurulumu için gerekli değildir?
   a) Minumum 1 Ghz işlemci
   b) En az 256 Mb ram
   c) Boş sabit disk alanı
   d) En az 256 Mb ekran kartı
   e) Kurulum dosyaları
5. Aşağıdakilerden hangisi SQL Server 2008 veri tabanı sürümü değildir?
   a) SQL Server 2008 Express Edition
   b) SQL Server 2008 XP Edition
   c) SQL Server 2008 Standard Edition
   d) SQL Server 2008 Enterprise Edition
   e) SQL Server 2008 Datacenter Edition

6. Aşağıdakilerden hangisi SQL Server 2008 kurulurken dikkat edilmesi gerekenlerden biri değildir?
   a) Fat Dosya Sistemi kullanılması
   b) NTFS Dosya Sistemi kullanılması
   c) Internet bağlantısı olması
   d) Sıkıştırılmış sürücü olmaması
   e) Güvenlik duvarında erişimlerin verilmiş olması

7. Yeni SQL Server kurulumu yapmak için aşağıdaki bağlantılardan hangisi kullanılır?
   a) New Cluster installation
   b) New Installation or add features to an existing installation
   c) Remove installation
   d) Tools
   e) Resources

8. Management Studio kurulması için sql server aşağıdaki bileşenlerin hangisini kurmalıdır?
   a) SQL Server Database Engine Services
   b) Analysis Services
   c) Reporting Services
   d) Management Tools
   e) Shared Features

9. SQL Server kurulurken veri tabanı yöneticisi kullanıcı adı ve şifresinin belirlenebilmesi için aşağıdaki doğrulama seçeneklerinden hangisi kullanılır?
   a) Windows Authentication Mode
   b) Mixed Mode
   c) Single Mode
   d) Dual Mode
   e) Hiçbiri

10. Aşağıdakilerden hangisi birden Fazla SQL Server kurulumunu aynı bilgisayarda yapmak istiyorsanız farklı olmalıdır
    a) Varsayılan veri klasörü
    b) Varsayılan log klasörü
    c) Instance ID
    d) SQL kurulum cd'si
    e) Server ID

Cevap Anahtarı
1.a, 2.e, 3.c, 4.d, 5.b, 6.a, 7.b, 8.d, 9.b, 10.c

YARARLANILAN KAYNAKLAR

Sql Server 2008 Kurulum, (2008). 10 Temmuz 2019 tarihinde https://www.cozumpark.com/sql-server-2008-kurulum/ adresinden erişildi.

Sql Server Books Online, (2012). 20 Temmuz 2019 tarihinde https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms130214(v=sql.105) adresinden erişildi.

SQL Server Installation Guide Step by Step, (2012). 14 Temmuz 2019 tarihinde https://blog.sqlauthority.com/2008/06/12/sql-server-2008-step-by-step-installation-guide-with-images/ adresinden erişildi.

TABLOLARI OLUŞTURMAK VE ÖZELLİKLERİNİ BELİRLEMEK

Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI VE YÖNETİM SİSTEMLERİ

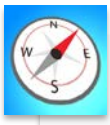
Dr. Öğr. Üyesi
Serdar AYDIN

ÜNİTE
4

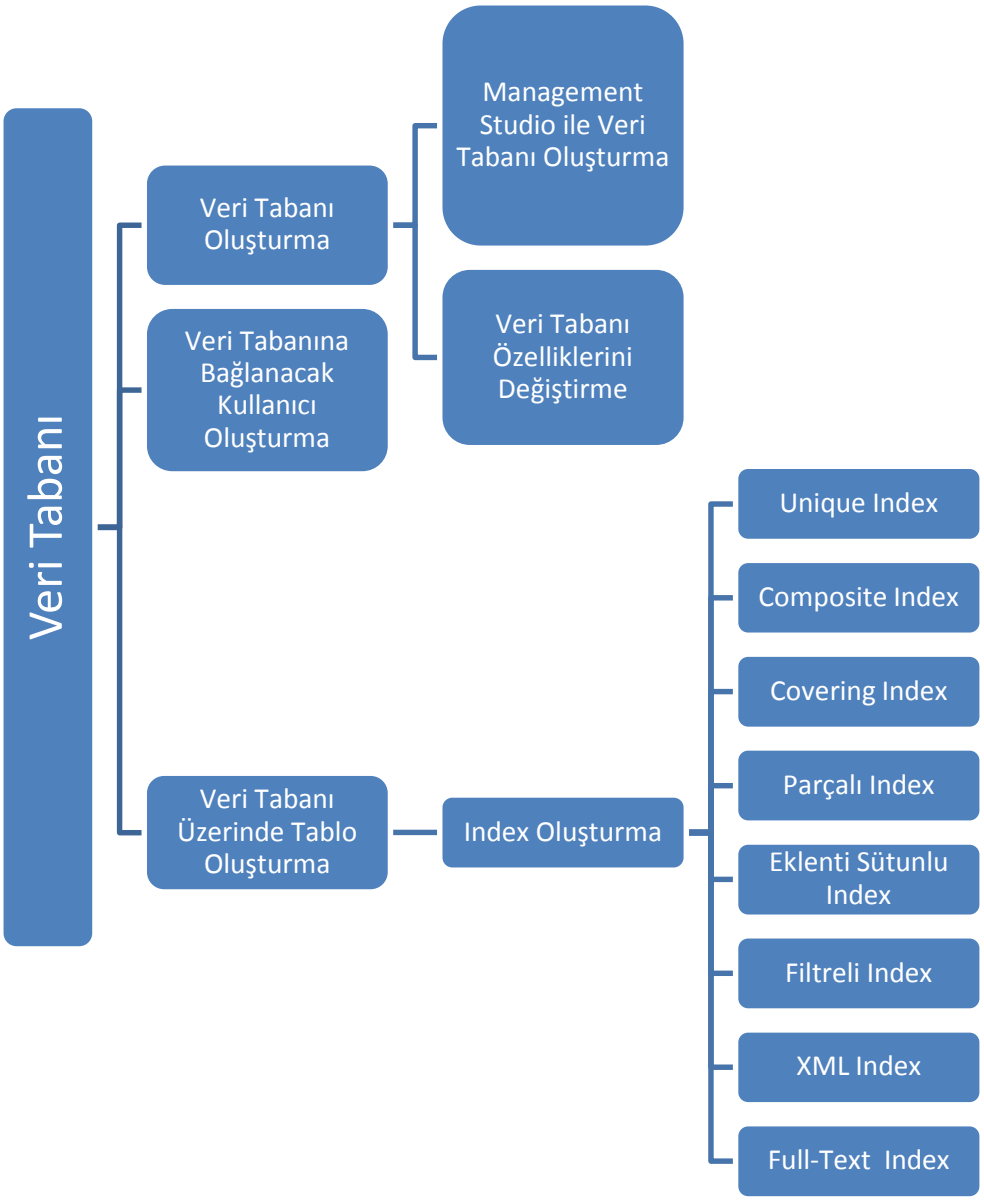
İÇİNDEKİLER

- SQL Server 2008 R2 Ortamında Tablo Oluşturmak
  - Veri Türleri
  - Tabloya Kayıt Girmek
  - Arama Sihirbazını Kullanmak

HEDEFLER



- Bu üniteyi çalıştıktan sonra;
  - SQL Server 2008 R2 ortamında tablo oluşturabilecek,
  - Tablolara çeşitli veri türlerinde alanlar tanımlayabilecek,
  - Arama sihirbazı kullanarak veri türü tanımlamanın nasıl gerçekleştirildiğini kavrayabileceksiniz.


Veri Tabanı
Veri Tabanı Oluşturma
Management Studio ile Veri Tabanı Oluşturma
Veri Tabanı Özelliklerini Değiştirme
Veri Tabanına Bağlanacak Kullanıcı Oluşturma
Veri Tabanı Üzerinde Tablo Oluşturma
Index Oluşturma
Unique Index
Composite Index
Covering Index
Parçalı Index
Eklenti Sütunlu Index
Filtreli Index
XML Index
Full-Text Index

GİRİŞ

Bu bölümde veri tabanı yazılımı kurulumunu yaparak veri tabanının en önemli ögesi olan tabloları ve tabloların özelliklerini öğrenecek, tablolarla ilgili çeşitli *işlemler* yapabileceksiniz.

Veriyi sistematik bir şekilde tutmak ve ondan faydalanmayı hızlı, esnek hâle getirmek belki de insanlık tarihindeki en güçlü medeniyet gereksinimlerinden biridir. Bu gereksinimin bilgisayarların gelişimine düşen izdüşümü olarak "veri tabanı" kavramını ele almak gerekir. Bu bölümde, veri tabanı programlamanın temel teoremleri hakkında fikir sahibi olunacak.

Management Studio aracılığı ile veri tabanı ve tablolar oluşturulması ve oluşturulurken girilmesi gereken parametreler adım adım görsellerle basitleştirilerek anlatılacaktır. Veri tabanınızın, sunucularınızın donanımına uygun şekilde oluşturulabilmesi ve yüksek performans ile çalışabilmesi için gerekli olan ayarlamalardan bahsedilecektir. Veri tabanından okunması istenen bilginin daha az veri okuyarak daha kısa zamanda getirilmesini sağlayan Indexler hakkında bilgi verilecek bu sayede tamamlanması saatler süren bir sorgunun uygun indexler kullanılarak nasıl saniyeler seviyesinde getirilmesinin sağlandığı anlatılacaktır. Bu bölüm sayesinde veri tabanı ve tablo oluşturmanın yanı sıra optimum veri tabanı, tablo ve index oluşturmayı öğrenmiş olacaksınız.

"Veri tabanı" kavramını netleştirirken bir veri tabanı yönetim sistemi olan SQL Server 2008'in temel mimarisini de kavramaya yönelik bilgi verilecektir. Veri tabanı programlarken kullanılan yegâne dil SQl'i ve bu dilin MS SQL Server lehçesi olan T-SQL dilini kavram olarak bir sonraki bölümde anlatılacaktır.

VERİ TABANI

Veri tabanı düzenli veriler topluluğudur. Kelimesinin anlamı; bilgisayar ortamında saklanan düzenli verilerle sınırlı olmamakla birlikte, daha çok bu anlamda kullanılmaktadır.

Veri tabanımızı Management Studio ile veya T-SQL ile oluşturabiliriz.

Bilgisayar terminolojisinde, sistematik erişim imkânı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesidir.

Bir başka tanımı da, bir bilgisayarda sistematik şekilde saklanmış, programlarca istenebilecek veri yığınıdır.

Veri Tabanı Oluşturma

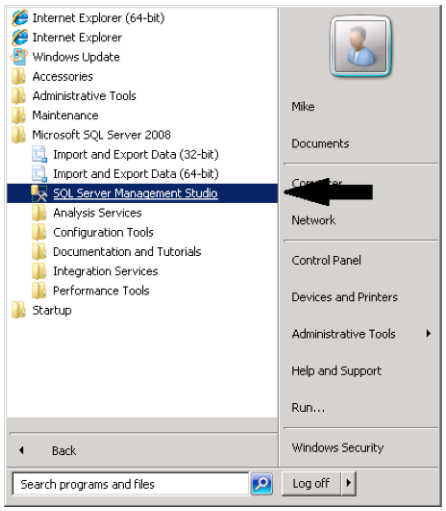
Veri tabanı iki şekilde oluşturulabilir:

- Management Studio kullanılarak,
- T-SQL ifadesi olan *CREATE DATABASE* deyimi kullanılarak. (T-SQL üzerinde 5. ünitede durulacaktır.)

Management Studio ile Veri Tabanı Oluşturma

SQL Server Management Studio, herhangi bir SQL altyapısını yönetmek için geliştirilmiş bir ortamdır. Veri tabanı ve SQL Veri Ambarının tüm bileşenlerine erişmek. yapılandırmak, yönetmek ve geliştirmek için gerekli ortamı sağlar. Managemer

bünyesinde, gelişmiş grafik araçlarını ve SQL script editörünü barındırır. Management studio ile veri tabanı oluşturmak için SQL SERVER Management Studio'yu açalım.



Şekil 4.1. Management Studio Açılması

Management Studio açıldığında bizleri ilk olarak bağlantı ekranı karşılar. Bu ekranda eğer kendi bilgisayarımızda çalışmakta olan Microsoft SQL Server'a bağlanmak istiyorsanız Authentication seçeneğini "Windows Authentication" seçerek bağlantı yapabiliriz. Aynı etki alanında (domain) bulunmadığımız ve bir kullanıcı adı ve şifre ile giriş yapılacak şekilde yapılandırılmış olan Microsoft SQL Server servislerine "SQL Server Authentication" seçeneği ile kullanıcı adı ve şifre belirtilerek bağlanabiliriz. Authentication Mode seçiminde kullanılabilecek seçenekler ünitenin ilerleyen kısımlarında "Kullanıcı Ekleme" başlığı altında ele alınacaktır.

Şekil 4.2. Management Studio Bağlantı Ekranı

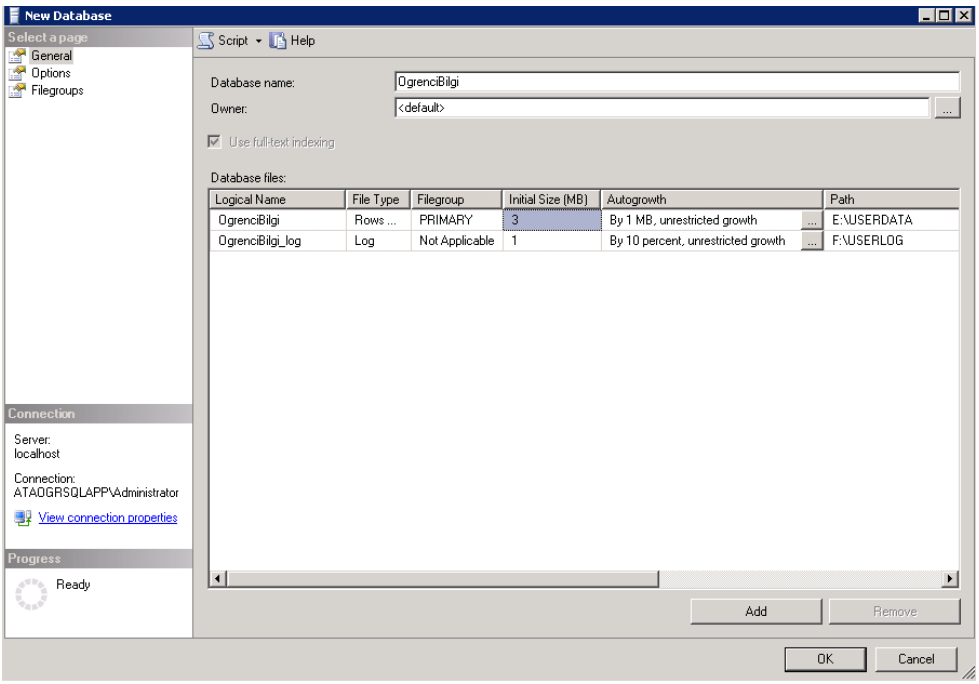
Başlamadan önce belirtmemiz gerekir ki SQL Server üzerinde maksimum 32767 veri tabanı oluşturulabilir.

Management Studio ile veri tabanı oluşturmak için SQL Server Management Studio'yu açıyoruz. Object Explorer'daki Databases üzerinde fareyle sağ tıklayarak NewDatabase komutunu seçiyoruz.

Şekil 4.3. New Database komutu seçimi

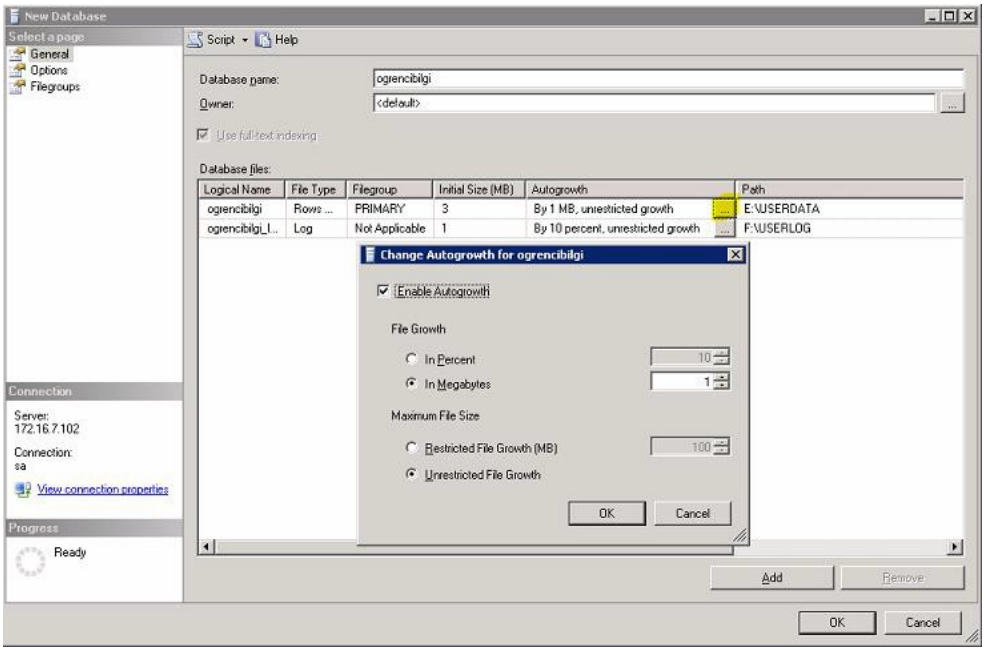
Veri tabanınızın özelliklerini bu ekrandan düzenleyebilirsiniz.

Database Name kısmına veri tabanına vermek istediğimiz ismi Şekil 4.4.te yer alan alana yazıp OK düğmesine bastığımızda varsayılan değerler ile bir veri tabanı oluşturabiliriz.



Şekil 4.4. Veri tabanı ismi belirlenmesi

Veri tabanımızı oluşturulduktan sonra *Owner* kutusundan veri tabanı sahibi olacak kullanıcı hesabı gösterilebilir. Default olarak veri tabanı sahibi o anda veri tabanını oluşturan kişidir. Alt kısımda öğrenci bilgi veri tabanına ait veri tabanı dosyası ve log dosyası otomatik olarak oluşur. Bu dosyalar için *Initial Size* kolonunda başlangıç boyutu ayarlanabilir. *Autogrowth* ile bu dosyaların büyüme oranları ve maksimum dosya boyutu ayarlanabilir.



Şekil 4.5. Autogrowth Ekranı

Autogrowth seçeneğini düzenleyerek günlük dosyaların gereğinden fazla büyümesine engel olabilirsiniz.

Autogrowth yanındaki üç nokta butonuna tıklanınca yukarıdaki şekilde görülen ekran karşımıza gelir. Burada *Enable Autogrowth* ile veri tabanı dosyasının *Initial Size* boyutu dolduğunda otomatik olarak dosyanın kendi kendini büyütmesi aktifleştirilir. *File Growth* kısmından bu büyümenin yüzde olarak mı (*In Percent*) yoksa MB olarak mı *(In Megabytes*) olacağı belirlenir. Bir dosya dolduğunda 1'er MB büyümesi için MB seçeneğini seçip, 1 MB olarak bırakıyoruz. *Maximum File Size* ile de veri tabanı dosyası için maksimum boyut belirlenir.

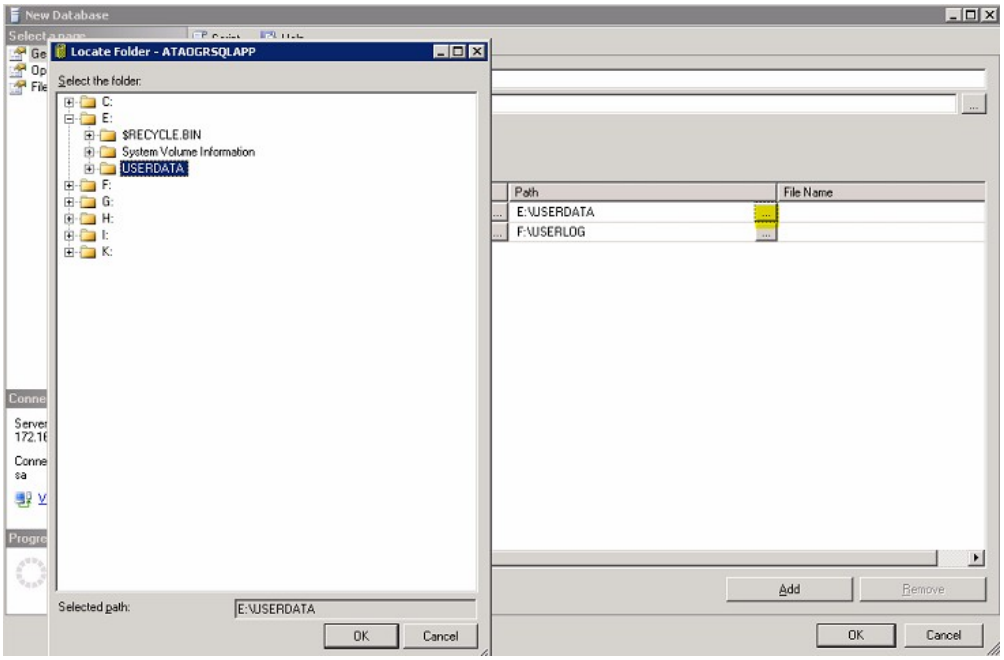
Herhangi bir maksimum dosya boyutu limiti uygulamayacaksanız, şekilde de olduğu gibi *Unrestricted File Growth* seçilir. Bilgisayarınızdaki disk kapasitesinde sorun olabilecek bir durum varsa *Restricted File Growth* ile maksimum bir dosya boyutu belirlenebilir.

Otomatik büyüme ayarlarını Log dosyası için de ayrıca benzer şekilde yapabilirsiniz.

Options sekmesinde veri tabanında saklanacak veriler için karakter ve dil kurallarının belirleneceği "collation" ayarı, veri tabanı onarım modunun belirlendiği "recovery mode" ayarı ve veri tabanının önceki veri tabanı yazılımı sürümlerinde çalışabilmesi için gerekli olan uyumluluk ayarı başta olmak üzere veri tabanının bir sonraki ünitede anlatılacak bir çok özelliği ayarlanabilir.

Bireysel Etkinlik

- Kendi bilgisayarınızda kuracağınız SQL Server 2008 üzerinde "OgrenciBilgi" isimli bir veri tabanı oluşturarak bu veri tabanının Türkçe verileri saklayabilmesi ve veri tabanı dosyalarının otomatik olarak kendini büyütmesini sağlayacak gerekli ayarlarını yapınız.



Şekil 4.6. Path Sütunu

Veri tabanınızın dosyalarının bulunacağı dizinleri bu ekrandan değiştirebilirsiniz.

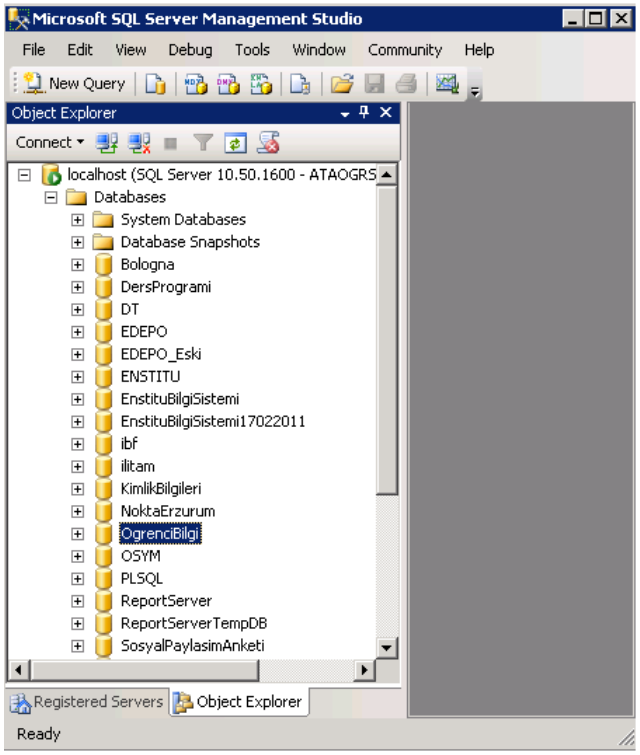
Path sütununda veri tabanı ya da log dosyasının depolanacağı konum belirlenir. Varsayılan olarak

%systemdrive%\Program Files\Microsoft SQLServer\MSSQL.1\MSSQL\Data

konumuna kaydedilir. İsterseniz, bu aşamada farklı bir konuma alabilirsiniz. Veri tabanı oluştuktan sonra normal şartlarda veri tabanı dosyalarının konumunu veri tabanı özelliklerine girerek değiştiremezsiniz. Ancak Detach-Attach yöntemi kullanılarak bu işlem yapılabilir. Aynı şekilde log dosyasının da konumunu değiştirebilirsiniz.

Yine alt kısımda bulunan ADD butonunu kullanarak ikinci, üçüncü vb. veri tabanı dosyası (NDF) ya da ikinci ya da üçüncü vb. Log dosyası oluşturulabilir.

Birden fazla veri tabanı ve log dosyası oluşturmanın amacı özellikle büyük veri tabanları için performans artımını sağlamaktır. Veri tabanı üzerinde işlem yapılırken veri tabanının saklandığı dosyalar üzerinde sürekli okuma - yazma işlemleri gerçekleştirilir. Bir den fazla dosyaya veri tabanı dosyalarının ve loğlarının yayılması, bu dosyaların da farklı sabit diskler üzerinde barındırılması okuma - yazma işlemlerinde fiziksel limitlerin artırılmasını sağlayabilir.

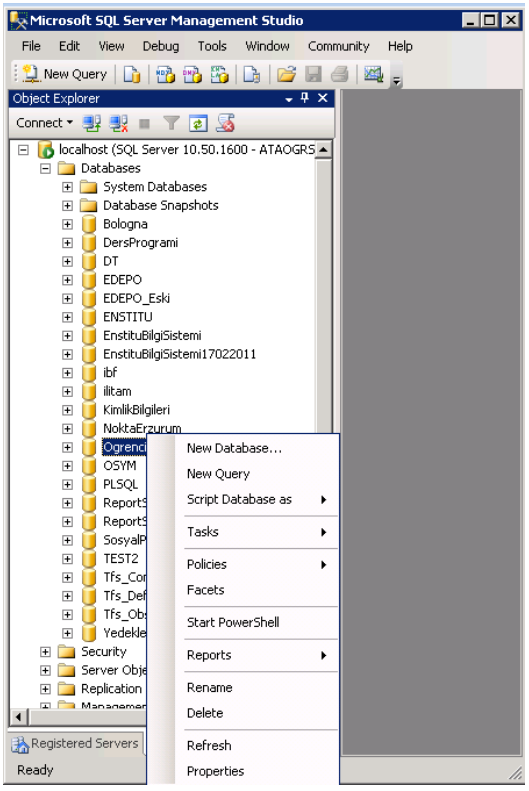


Şekil 4.7. Veri Tabanının Management Studio'da görünümü

Veri Tabanı Özelliklerini Değiştirmek

Veri tabanının erişim ayarları ve nesnelerin çalışma ayarlarının yapılması için veri tabanının özelliklerinin değiştirilmesi gerekebilir. Bu özelliklerin detaylarını bir sonraki ünitede Tablo 5.1.de görebilirsiniz. Bunu yapabilmek için

Veri tabanınızın temel özelliklerini Management Studio ile bu ekrandan değiştirebilirsiniz.



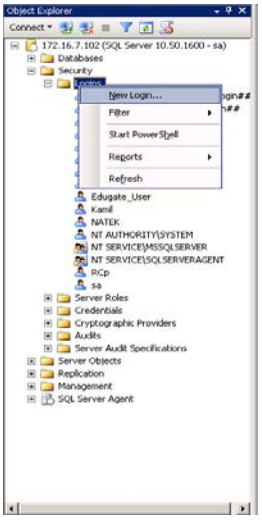
Şekil 4.8. Veri tabanı özellikleri değiştirme

Şekil 4.8.de görüldüğü gibi özelliklerini değiştirmek istediğimiz veri tabanının ismini sağ tıklayıp *Properties* bağlantısını tıklıyoruz.

Veri Tabanına Bağlanacak Kullanıcı Oluşturma

İlk olarak SQL Server 2008 Management Studio programını açıyoruz ve sırasıyla aşağıdaki adımları izliyoruz:

- Kullanıcının ekleneceği veri tabanı sunucusu açılır.
- Security sekmesinin altında yer alan Logins öğesi sağ tıklanır.
- "New Login" seçeneği tıklanır.

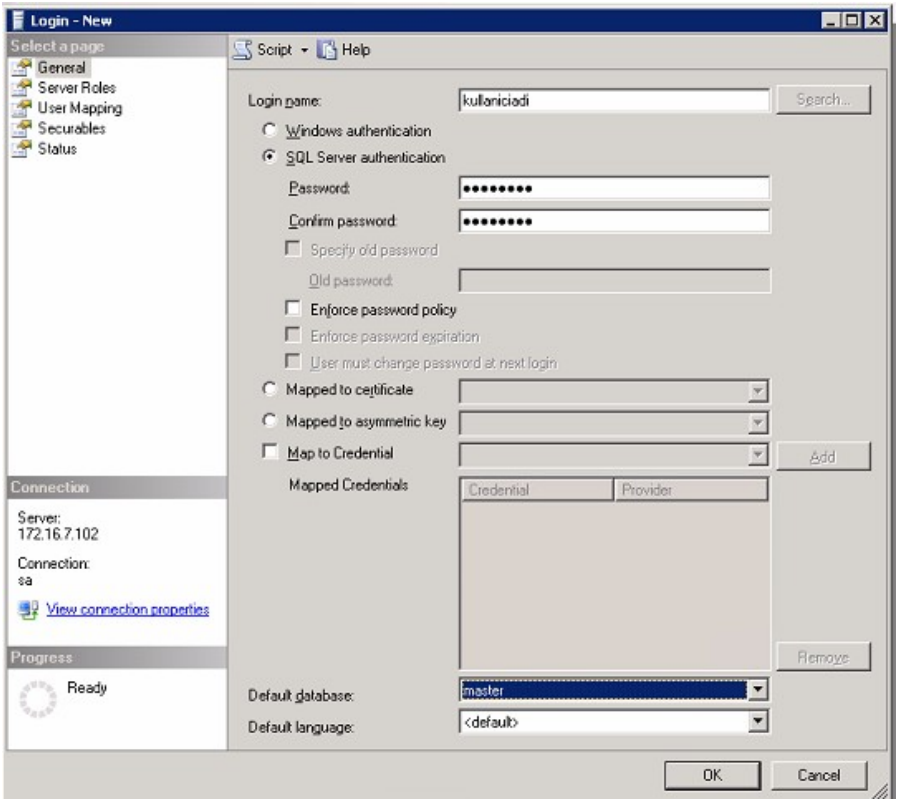


Şekil 4.9.1. Adım

İstemcilerin veri tabanına erişebilmesi için kullanıcı tanımlamanız gerekir.

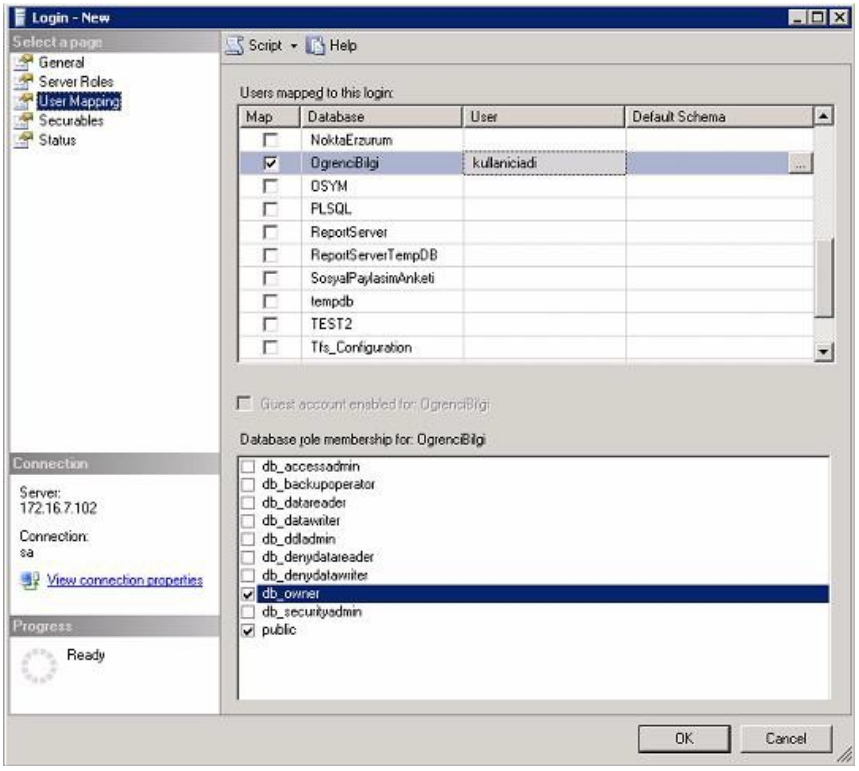
- Açılan form üzerinde "Login name" oluşturmak istediğimiz kullanıcı için kullanıcı adı olacaktır.
- Kullanıcının veri tabanına bağlanırken hangi doğrulama yöntemini kullanacağı "Windows authentication" veya "Sql Server Authentication" seçilir.
- Windows authentication: Bu seçenek tercih edilirse erişecek olan kullanıcının doğrulama işlemi "Windows kullanıcı doğrulama" alt yapısı ile yapılır. Bu kullanıcı ismine sahip Windows kullanıcısına veri tabanına erişim yetkisi verilir. Windows authentication modu varsayılan kimlik doğrulama modudur ve biraz sonra bahsedilecek olan Sql Server Authentication modundan daha güvenlidir. Windows authentication mod Kerberos güvenlik protokolünü kullanır.
- Sql Server Authentication: Bu seçenek tercih edildiğinde eklemek istediğiniz kullanıcı adı ve hemen altında yer alan Password (Şifre) ve Confirm Password (şifre doğrula) kutucuklarına girilecek olan şifre ile veri tabanına erişilebilir. Bu kullanıcı adı ve şifreler SQL Server bünyesinde barındırılır. Bu modu kullanan kullanıcılar, her bağlantı için kullanıcı adı ve şifre belirtilmek zorunluluğu vardır. Oluşturulan kullanıcının Windows işletim sisteminde dâhili bir kullanıcı olarak oluşturulmasına gerek yoktur.
- Enforce password policy: Bu seçenek aktifleştirildiğinde şifrenin karmaşıklığı (büyük harf içermeli bir özel karakter içermeli vb.) geçerlilik süresi ve sonraki oturum açılmasında şifre değiştirme zorunluluğu gibi bazı parola yönetme politikaları aktifleştirilebilecektir.

- Default database: Oluşturulan kullanıcının varsayılan veri tabanının belirlendiği seçenektir.
- Default language: Oluşturulacak olan kullanıcının varsayılan dil tercihinin ayarlandığı seçenektir.



Şekil 4.10. Veri tabanı kullanıcısı oluşturma ekranı

Veri tabanına erişecek olan kullanıcının erişim haklarını bu ekrandan belirleyebilirsiniz.



Şekil 4.11.2. Adım

- Sol menüde yer alan User Mapping sekmesi kullanıcının erişebileceği

tabanlarının ve bu veri tabanlarına hangi haklarla erişebileceğinin belirlendiği kısımdır.

- İlgili veri tabanı seçildikten sonra "role membership" seçeneklerinden verilmek istenen yetkiler seçilir ve "OK" seçeneğine tıklanınca kullanıcımız eklenmiş olur.

Bireysel Etkinlik

- Kendi SQL Server 2008 sunucunuzda "test" isimli yeni bir veri tabanı oluşturunuz.
- Oluşturduğunuz veri tabanına erişmesi için Sql Server Authentication modu kullanan yeni bir kullanıcı oluşturunuz.
- Bu kullanıcıya veri tabanı üzerinde sadece select komutu çalıştırabilme yetkisi veriniz.

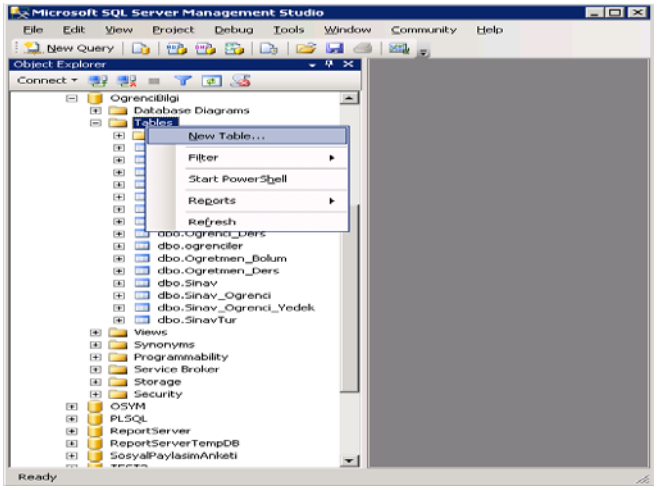
Tablo 1.1. Verilebilecek yetki tipleri ve açıklamaları aşağıdaki gibidir.

db_owner	**DB üzerinde verilebilecek en yüksek level'lı yetkidir. DB üzerinde bütün konfigürasyonları yapabilir ayrıca DB'yi silebilir (drop da edebilir).**
db_securityadmin	Yetkileri yönetebilir, role yetkilerinde değişiklik yapabilir.
db_accessadmin	Windows ve SQL Server Login'lerinin DB'ye erişimlerinde ekleme ya da çıkarma yapabilir.
db_backupoperator	DB'yi backup'layabilir.
db_ddladmin	DB üzerinde herhangi bir DDL (Data Definition Language) komut çalıştırabilir.
db_datawriter	User tabloları üzerinde delete, insert, update komutlarını çalıştırabilir.
db_datareader	User tabloları üzerinde select komutu çalıştırabilir.
db_denydatawriter	User tablolarında delete, insert, update komutları çalıştırılamaması için kullanılır.
db_denydatareader	User tabloları üzerinde select komutu çalıştırılamaması için kullanılır.

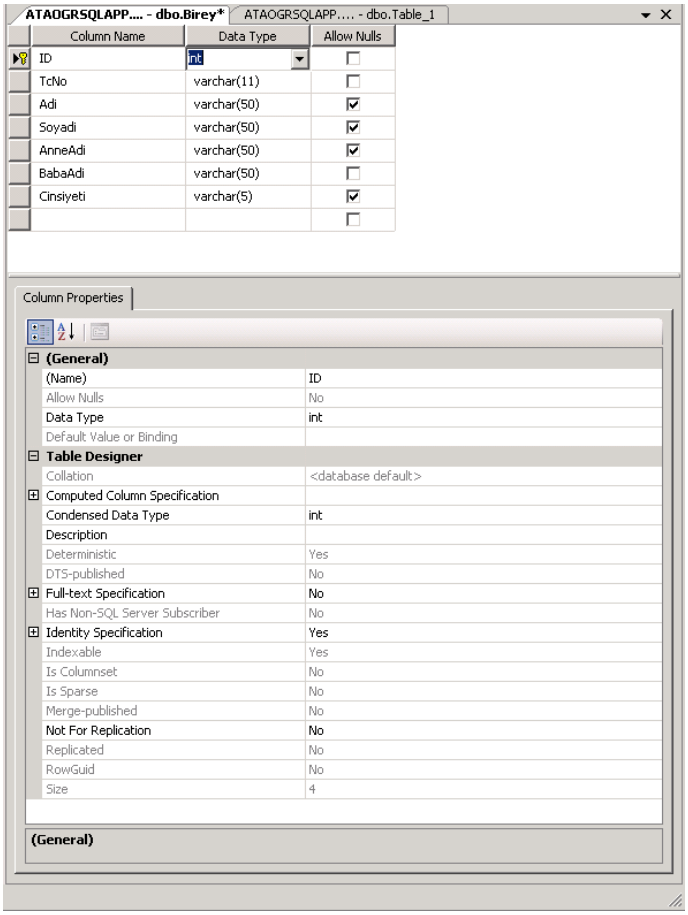
Veri Tabanı Üzerinde Tablo Oluşturmak

Veri tabanınızın altında yer alan Tables dizinini sağ tıklayıp açılan menüden "New Table" seçeneğini tıklıyoruz.

Management Studio ile tablo oluşturma ekranı



Şekil 4.12. Yeni veri tabanı oluşturma ekranı Şekil



Şekil 4.13. Oluşturulan Örnek Tablo

Açılan pencereyi kullanarak tablo oluşturabilmemiz için tablomuzun yapısını planlamış olmamız ve bu yapıyı Şekil 4.13.te görüldüğü şekilde "Sütun – Veri Tipi" ilişkisi içerisinde işlememiz gerekir. "Column Name" sütunlara vereceğimiz isimin, "Data Type" sütunların barındıracağı veri tipinin, "Allow Nulls" ise bu tabloya veri girişi yapılırken belirtilen sütunun boş geçilip geçilemeyeceğinin belirlendiği bölümlerdir. Seçilen sütunlara ilişkin özellikler hemen formun alt kısmında yer alan "Column Properties" sekmesinde belirlenebilmektedir.

Index oluşturma

SQL Server açısından index kullanımının en önemli amacı, istenen bilginin daha az veri okuyarak daha kısa zamanda getirilmesini sağlamaktır. Index kullanarak bir tablonun tamamını okumaktansa index key vasıtasıyla okumak istediğimiz kayda daha hızlı ulaşmak mümkündür. Tamamlanması saatler süren bir sorgunun uygun index'ler kullanılarak saniyeler seviyesinde getirilmesi sağlanabilir.

Veri tabanınızın performans artırımını indexler ile sağlayabiliriz.

Peki ama tam olarak Index nedir? Gerçek hayattan bir örnek vererek Index kavramını daha net açıklamaya çalışalım.

Bir telefon rehberi düşünün. Hani şu eskiden PTT'nin dağıttıklarından. İstanbul'da yaşayan herkesin telefon bilgilerinin bu rehberde kayıtlı olduğunu düşünelim. Mehmet Yılmaz adında bir arkadaşım var ve telefon numarasını kaybettiğim için kendisine uzun süredir ulaşamıyorum. Telefon rehberini kullanarak Mehmet'in telefon numarasını bulmak istiyorum.

İlk durumda şunu düşünelim (Index olmayan bir tablo). Telefon rehberi

değil. Yani ne ada göre ne de soyada göre herhangi bir sıralama yapılmamış, isimler karışık olarak bulunmaktadır. Böyle bir durumda ben Mehmet'i bulmak için *bütün telefon defterini okumak zorundayım.*

İkinci durumda ise telefon defterinin isme göre sıralı olduğunu düşünelim. (Index olan bir tablo) Bu durumda biliyorum ki Mehmet M harfinde. Kabaca rehberin ortasına gelirim. Ve oradaki harfe bakarak ileriye ya da geriye gitmem gerektiğine karar veririm. Yani bir önceki örneğe (Index olmayan durum) göre çok çok daha hızlı bir şekilde Mehmet'i bulabilirim.

İşte Index kullanımının amacı budur. *İstenilen bilgiye daha az veri okuyarak daha hızlı bir şekilde erişmektir.*

Veri tabanı üzerine kayıt sayısı arttıkça çekilen sorguların döndüreceği değerlerin karşımıza gelmesi de o kadar zaman alacaktır. Örnek olarak

" Select * from Tbl_Personel Where Adi Like 'a%' " şeklinde çekilen bir sorguda tablodaki kayıtları tek tek aramaya başlar ve ismi "a" harfi ile başlayan kayıtları listeler, bu işlemi yaparken tablonun en başından başlar ve bu yoğun bir tablodaki erişim hızını oldukça etkiler.

Veriler hızlı erişim için kullanılması gereken yöntem ise index'lemedir. SQL server sorgu çalıştırıldığında öncelikle index'leme var mı kontrol eder ve sorguyu bu duruma göre çalıştırır.

İki çeşit index vardır: "CLUSTERED ve NONCLUSTERED". Bir tabloda 1 tane CLUSTERED 999 adet de NONCLUSTERED index oluşturulabilir. Oluşturduğunuz bir tablo üzerinde eğer id kolonunu "SET PRIMARY KEY" yaparsanız, bu kolon otomatik olarak bir CLUSTERED INDEX oluşturur.

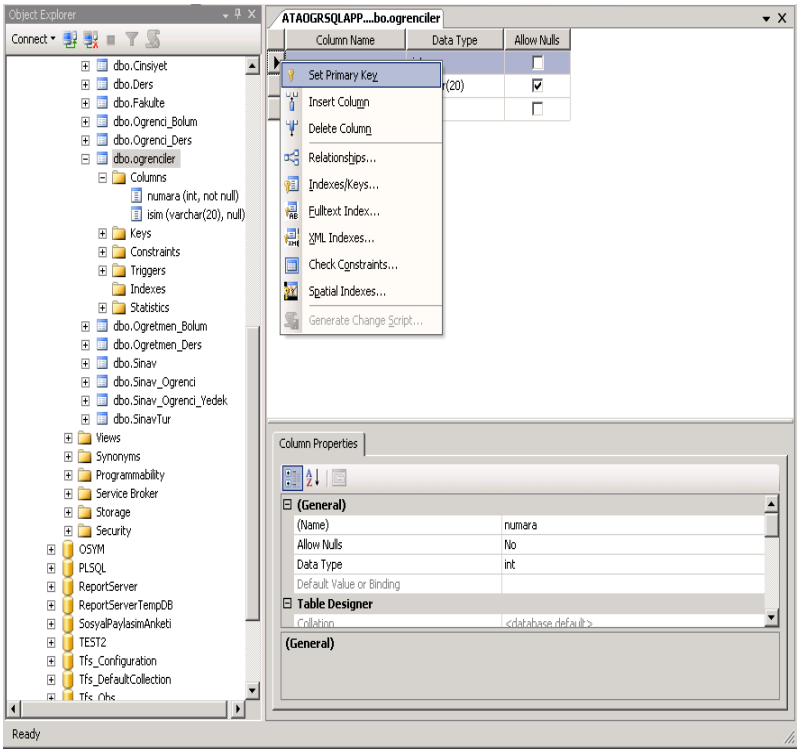
Clustered ındex

Clustered sıralanmış, küme hâline getirilmiş anlamına gelir. Tıpkı auto increment numara verdiğiniz ve identity olarak belirlediğimiz kolon gibi işlem görür. Sql Server 2008 tablolarında auto increment ve primary key olarak belirlediğiniz sütun otomatik olarak clustered index olarak kaydedilir. Bir tabloda sadece bir clustered index bulunur. Sıralı veya küme hâline getirilmiş indexin amacı kabaca yukarıda bahsedilen ansiklopedi öğelerinin alfabetik olarak sıralanması örneğinde olduğu gibi aranılan öğenin alfabetik sıra takip edilerek hızlı ve kolay bir şekilde bulunabilmesini sağlamaktadır.

Clustered index, b-tree yani dengeli ağaç yapısına sahiptir ve veriler leaf (yaprak) denilen seviyelerde tutulur.

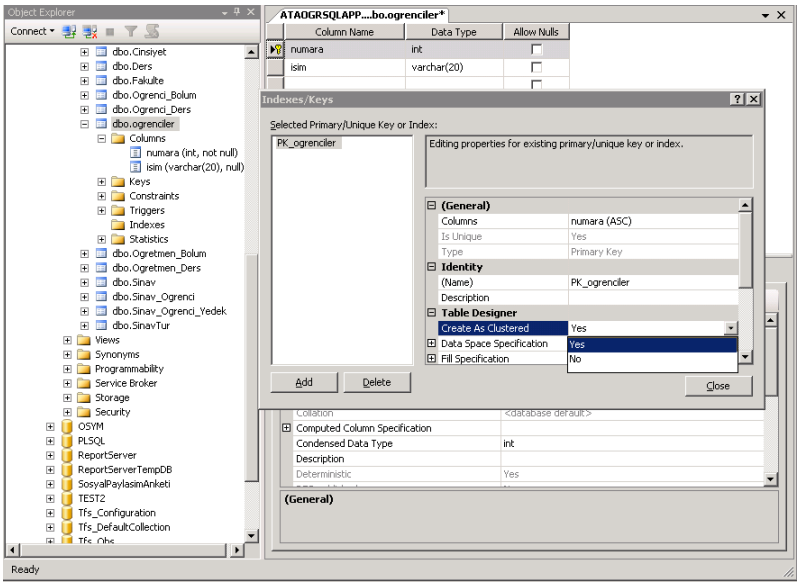
Non-clustered ındex

Clustered index'in aksine veriler sıralı şekilde tutulmaz. Yine b-tree yapısındadır, verinin kendisi değil nerede olduğu (adresi) b-tree ağacının yapraklarında tutulur. Buna en iyi örnekte bir kitaptır. Kitapların başında yer alan içindekiler kısmında hangi konuların hangi sayfalarda yer aldığı belirtilir. Kitapların sonlarında yer alan ve kavramların geçtiği sayfa numaralarının bulunduğu dizinler yine örnek olarak verilebilir. Aradığınız öğeleri tüm sayfaları tek tek okumak yerine dizin veya içindekiler yardımıyla çok daha hızlı bulabilirsiniz.



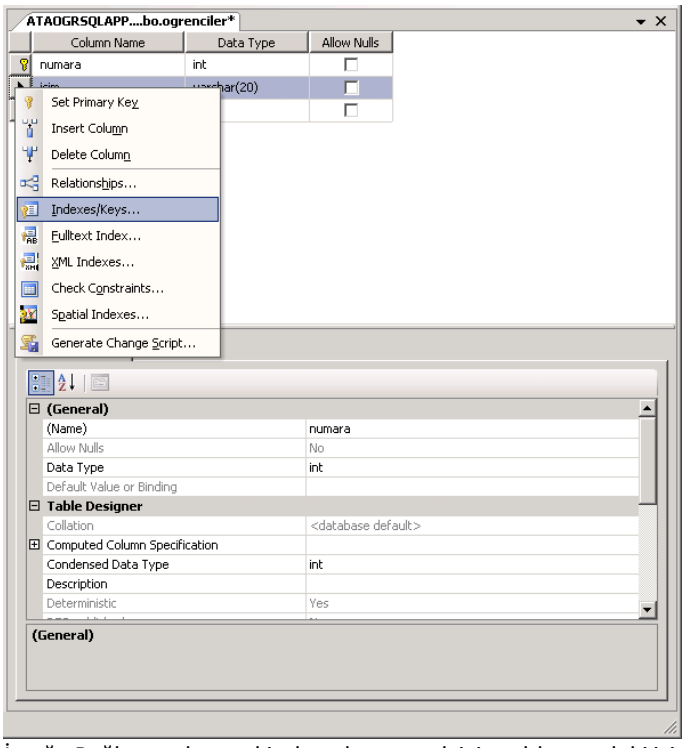
Şekil 4.14. Primary Key oluşturma

Sütunlardan birini primary key olarak indexlemeliyiz.

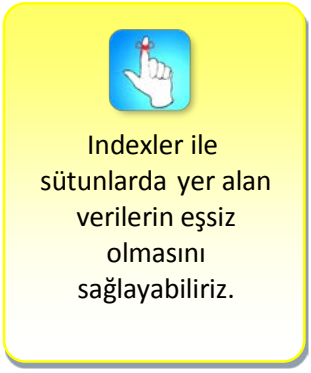


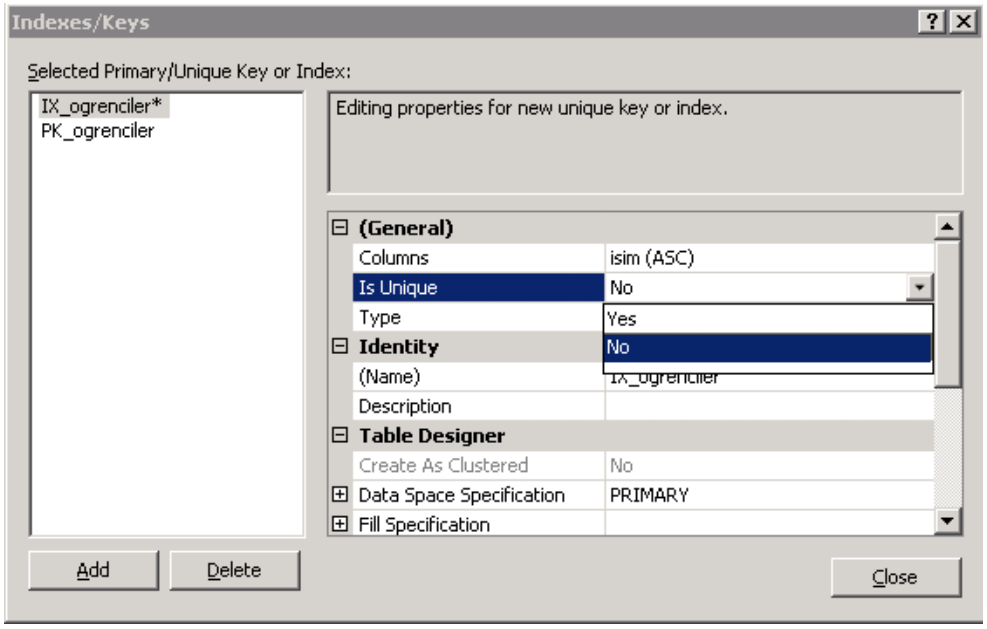
Şekil 4.15. Oluşturulan indexin clustered veya nonclustered yapıda olmasını belirleme

İsteğe bağlı nonclustred index oluşturalım ve tablomuzdaki isim sütunun indexlenmesini sağlayalım.



Şekil 4.16. İsteğe Bağlı nonclustred index oluşturmak için tablomuzdaki isim sütunun üzerinde sağ tıklanıp Index/Key seçilir.





Şekil 4.17.Is Unique "No" Olarak seçilir.

Böylelikle index oluşturduk, oluşturduğumuz indexin özelliklerini Indexes/Keys penceresinden değiştirebiliriz.

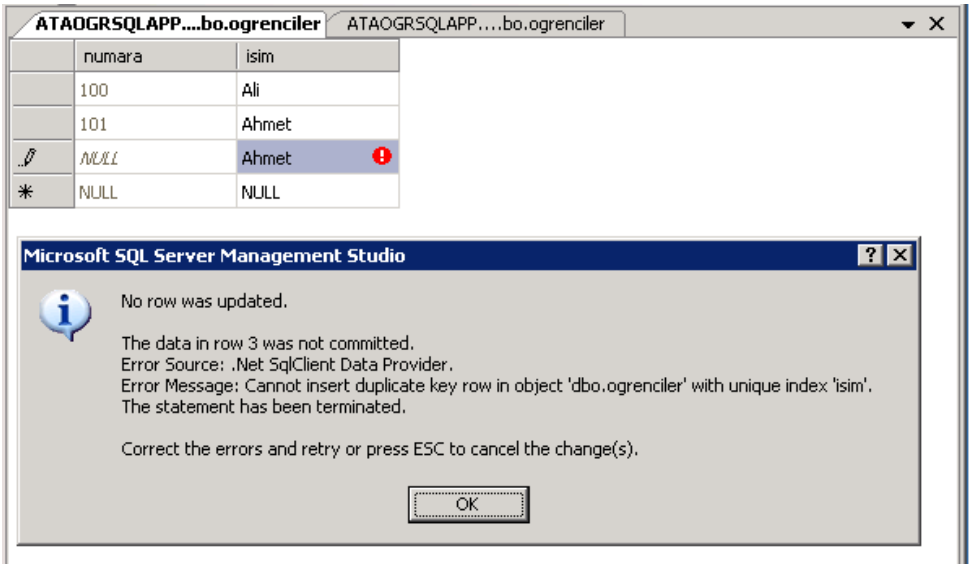
Bireysel Etkinlik

- Kendi SQL Server 2008 sunucunuzda oluşturduğunuz tabloların doldurulmasını sağlayınız ve indexlerini oluşturarak sorguların indexleme öncesindeki performansı ile indexleme sonrası performansını karşılaştırınız.

Unique sözcüğü

Bir tablo indexleme alanı olarak seçilen sütundaki verilerin tekrarlanması istenmiyorsa indexleme yapılırken "Is Unique" kutusu Yes olarak işaretlenmelidir. Biz isim sütunu için oluşturduğumuz indexin Unique index olmasını sağlarsak aynı isme sahip 2 farklı öğrencinin tabloya eklenmesine engel

olmuş oluruz. Unique olarak indexlenmiş olan sütuna aynı kayıttan 2. kez girmeye çalıştığımızda aşağıdaki gibi bir hata ile karşılaşırız.



Şekil 4.18. Unique olarak indexlenmiş olan sütuna aynı kayıttan 2. kez girmeye çalıştığımızda karşılaştığımız hata ekranı

SQL Server Index Türleri

Clustred ve non-clustred indexlerin her ikisi üstünde de geçerli olabilecek bazı durumlar vardır. Bu durumları ayrıca bu başlık altında ele alacağız.

Veri tabanı yapısına bağlı olarak oluşturacağımız index türü performans için önemlidir.

Unique ındex

Index'teki verilerin tekrarlanamaması maksadıyla kullanabiliriz. Bunun için, bir indexi UNIQUE deyimi ile tanımlamak yeterlidir. Hem veri çekme işlemlerini hızlandırmada hem de verinin tekrarlamamasını denetlemede aynı index kullanılabilir. UNIQUE index clustered türden olabileceği gibi nonclustered türden de olabilir. Primary Key Constaint veya Unique Constraint tanımı yapıldığı zaman, SQL Server bir Unique Index tanımlama işlemini gerçekleştirir. Bu türden bir Constranit tanımlanırken SQLServer'e clustered olup olmayacağı hakkında bilgi verilebilir.

Karma (Composite) ındex

İlişkisel veri tabanında, bir indexin ister tekil, ister çoğul index olsun bir tek alandan ibaret olması sık rastlanan bir durumdur ancak bu bir kural değildir. Gerektiğinde 16 sütun veya toplam uzunluklar 900 Byte'i aşmamak üzere birden fazla alanı kapsayan ve composite index olarak isimlendirilen bir index tanımlanabilir. Ayrıca SQL Server 2005'ten itibaren, non-clustred indexlerde anahtar olmamak koşulu ile uç seviye sayfalarda 900 Byte'dan daha fazla veriyi de *eklenti sütun* (included column) olarak içerebilir.

Kapsam (Covering) ındex

Bir sorgunun WHERE kısmı da dâhil olmak üzere, seçilen sütunları ile birlikte bir tek index olarak tanımlanmasına Covering Index denir. Bu türden bir non-clustred index tanımlandığında, SQL Server sadece bu indexi kullanır. Covering index genellikle oluşturulurken çok işlem gerektirir ama sorgunun çok hızlı neticelenmesini sağlar.

Ancak Covering Index tanımlarken, Indexler arası bölge kavramına (clustred index içerisindeki sütunu tekrar indexe dâhil etmek gerekmez) ve sorgulanan sütunlarla, indexlenen sütunların sırasına dikkat ederek tanımlanması gerekir.

Parçalı ındexler

Parçalı indexler, SQL Server 2005 ile birlikte gelen bir index türü olup farklı fiziksel dosya gruplarına dağıtılmış indexlere verilen addır. Fiziksel olarak farklı disklere dağılmış indexler, paralel I/O performansını artırır ve kaynak çekişmelerinin önüne geçer. Parçalı indexler de clustred veya non-clustred olabilir.

Eklenti sütunlu indexler

Eklenti sütunlu indexler, SQL Server 2005 ile gelen bir özellik olup index anahtar değerleri dışında, index yapısının en uç sayfalarında gerçek veriler de tutularak sorguların hızlandırılması sağlanabilir. Konu index oluşturma kısmında örneklendirilecektir.

Filtreli indexler

Filtreli indexler, bir tabloda yer alan belli sütunlar için bütün kayıtları indexlemek yerine sadece kurala uyan satırları indexlemek amacıyla kullanılabilen ve SQL Server 2008 ile birlikte kullanıma sunulan bir yapıdır. Konu index oluşturma bahsinde detaylandırılacaktır.

Indexler ile özellikle SELECT sorgusunun hızı önemli ölçüde artacaktır.

XML ındexler

SQL Server, XML Sütunlar üstünde de sorguların hızlanmasını sağlamak üzere index tanımlanmasına müsaade eder. Ancak XML indexler, veri tabanı indexlerine göre biraz farklılık içerir ve daha fazla ön şart gerektirir.

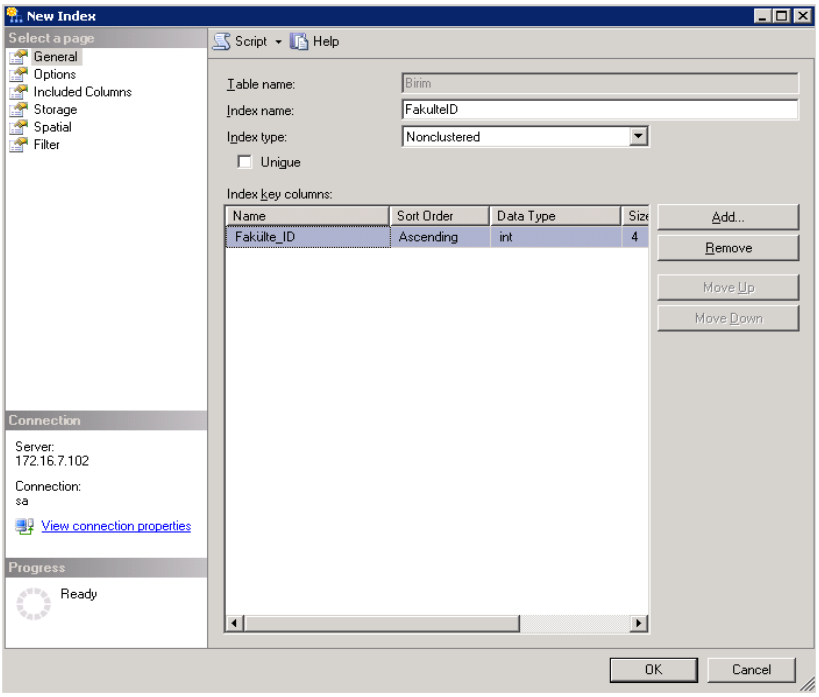
Full-Text ındexler

Full-text indexler, özellikle metin ifadelerin farklı formları ve dil kurallarından bağımsız olarak hızlı sorgulanmaları maksadıyla tercih edilir. SQL Server 2008 sürümünden itibaren Fulltext indexler veri tabanı motoru ile bütünleştirilmiş durumdadır. Yapıları diğer indexle aynı olmasına rağmen, birçok yönü ile diğer tür indexlerden farklılık gösterir.

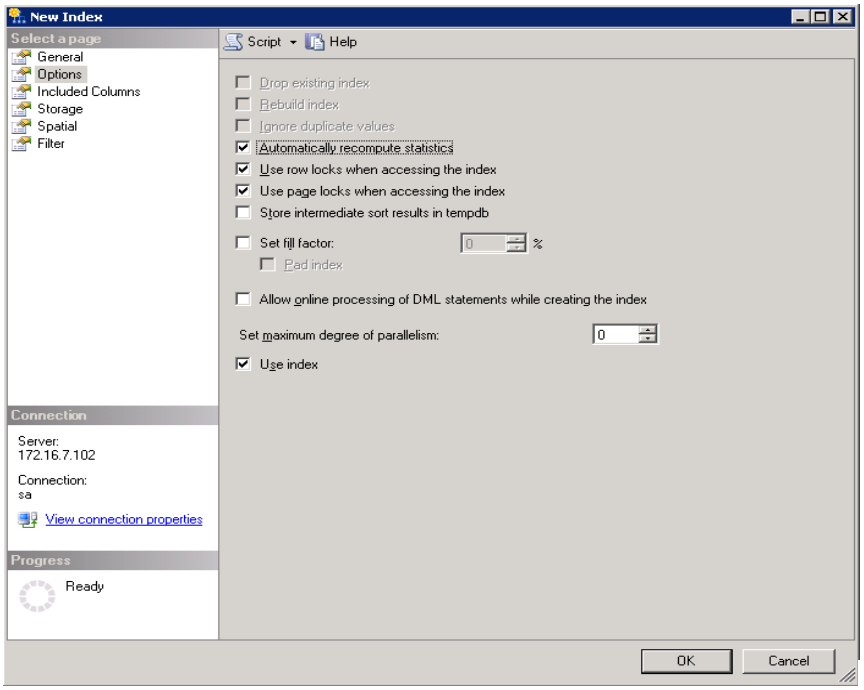
FILLFACTOR ve PAD INDEX Parametrelerine Karar Vermek

Bir OLTP sistemde, sürekli olarak INSERT, UPDATE ve DELETE işlemleri olduğundan, index sayfalarında boş alanlar bırakarak gitmek daha akıllıca olur. OLAP için ise Index sayfalarının daha dolu olması tercih edilir. Bu işlemler için, FILL FACTOR ve PAD_INDEX parametresi kullanılabilir. FILL FACTOR, uç seviye sayfaların ne kadar doluluk oranlarında olacağını tayin eder. PAD_INDEX ise bu oranın uç seviyenin dışındaki sayfalarda da geçerli kılınıp kılınmayacağını ifade etmek için kullanılır. PAD_INDEX denilmezse ara seviye sayfalar (intermediate

pages) için en az iki index sığacak kadar olmak üzere SQL Server tarafından uygun bulunan miktarda boş yer bırakılır.

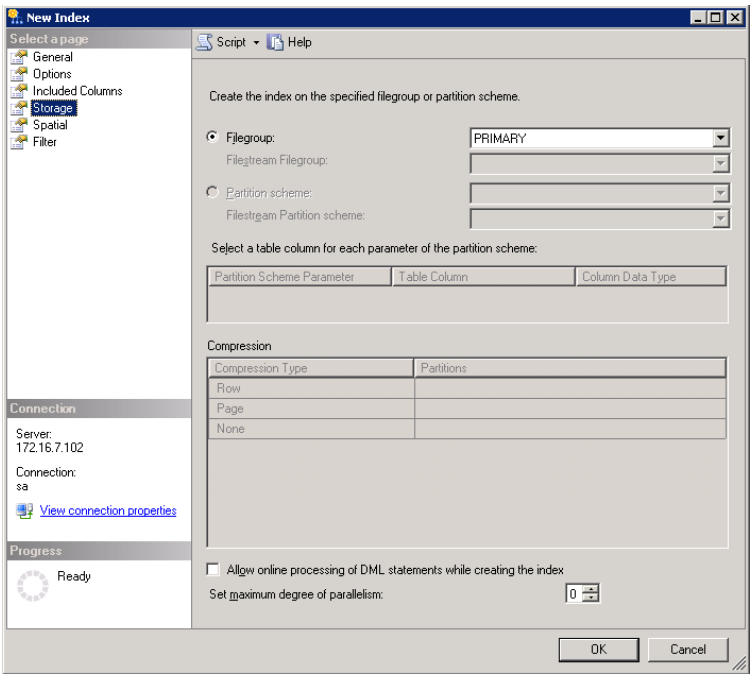


Şekil 4.19. Yeni Index oluşturma



Şekil 4.20. Indexin Options Penceresi

Yukarıda da görüldüğü gibi Options kısmından Fill factor'un ayarlanmasından Index'in çalışacağı maksimum paralellik derecesinin belirtilmesine kadar ya da Index kullanılırken Page lock'ları ya da Row lock'ların kullanılıp kullanılmadığının belirtildiği bir kısımdır. Şimdi diğer opsiyonlara bakalım. Sol taraftaki menüden aşağıya doğru bakıyoruz. Included Columns sekmesini Clustered Index oluştururken kullanamayacağımız için ona bakmıyoruz şu anda. Included column özelliği sadece Non Clustered Index'lerde kullanılan bir özelliktir. Storage kısmı aşağıdaki gibidir.



Şekil 4.21. Non Cluster Index'lerde Kullanılan Storage Kısmı

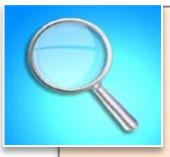
Yukarıda görülen storage kısmında oluşturulacak olan Index'in depolanacağı Filestream'in seçildiği belirlendiği bir kısımdır. Aynı şekilde eğer tablonuz Partition yapısına sahipse ya da tablo Compression olmuşsa bunlar da bu kısımda belirtilir. Bu belirtilen opsiyonlar en çok kullanılan opsiyonlardır. Index için opsiyonlar belirtildikten sonra bu opsiyonlarında içerdiği Index'in create script'ini üst taraftan script kısmından oluşturup execute ettikten sonra Index'imiz başarıyla oluşacaktır.

Bireysel Etkinlik

- Bir tablo üstünde hiç index yoksa kayıtlar hangi sıraya göre saklanır? Sorgulamada bunun bedeli nedir? Araştırınız.
- Clustred ve non-clustred indexe günlük hayattan örnekler veriniz.
- SQL Server, Clustred index tekil değil ise nasıl bir yol izleyerek tekil clustred index numarası oluşturur?
- Kapsam index ve karma index deyince aklmıza ne geliyor?
- Sizce çok az sayıda kayıt içeren bir tablo için index tanımlamalı mıdır? Araştırınız.

Özet

- **VERİ TABANI**
- *Veri tabanı düzenli veriler topluluğudur.* Kelimesinin anlamı; bilgisayar ortamında saklanan düzenli verilerle sınırlı olmamakla birlikte, daha çok bu anlamda kullanılmaktadır.
- Bilgisayar terminolojisinde, sistematik erişim imkânı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesidir.
- Bir başka tanımı da bir bilgisayarda sistematik şekilde saklanmış, programlarca istenebilecek veri yığınıdır.
- **Veri Tabanı oluşturma**
- Veri tabanı iki şekilde oluşturulabilir:
  - Management Studio kullanılarak,
  - T-SQL ifadesi olan CREATE DATABASE deyimi kullanılarak. (T-SQL üzerinde 5. Ünitede durulacaktır.)
- **Management Studio ile Veri Tabanı oluşturma**
- Management Studio ile veri tabanı oluşturmak için SQL Server Management Studio'yu açıyoruz. Object Explorer'daki Databases üzerinde fareyle sağ tıklayarak NewDatabase komutunu seçiyoruz. Database Name kısmına veri tabanına vermek istediğimiz ismi yazıp OK düğmesine basıyoruz.
- **Veri Tabanı Özelliklerini Değiştirmek**
- Veri tabanının erişim ayarları ve nesnelerin çalışma ayarlarının yapılması için veri tabanının özelliklerinin değiştirilmesi gerekebilir. Bunu yapabilmek içinözelliklerini değiştirmek istediğimiz veri tabanının ismini sağ tıklayıp *Properties* bağlantısını tıklıyoruz.
- **Veri Tabanı Üzerinde Tablo oluşturma**
- Tablo eklenmek istenen veri tabanına sağ tıklayıp New Table seçeneği ile açılan pencereye Kolon (Sütun) isimleri ve türleri belirlenerek tablo oluşturulabilir.
- **Index oluşturma**
- SQL Server açısından index kullanımının en önemli amacı, istenen bilginin daha az veri okuyarak daha kısa zamanda getirilmesini sağlamaktır. Index kullanarak bir tablonun tamamını okumaktansa index key vasıtasıyla okumak istediğimiz kayda daha hızlı ulaşmak mümkündür. Tamamlanması saatler süren bir sorgunun uygun index'ler kullanılarak saniyeler seviyesinde getirilmesi sağlanabilir.
- İki çeşit index vardır: "CLUSTERED ve NONCLUSTERED". Bir tabloda 1 tane CLUSTERED 249 adet de NONCLUSTERED index oluşturulabilir. Oluşturduğunuz bir tablo üzerinde eğer id kolonunu "SET PRİMARY KEY" yaparsanız, bu kolon otomatik olarak bir CLUSTERED INDEX oluşturur.
- **Sql Server Index Türleri**
  - **Unique Index: Index**'teki verilerin tekrarlayamaması maksadıyla kullanabiliriz. Bunun için, bir indexi UNIQUE deyimi ile tanımlamak yeterlidir.
  - **Karma Index:** Gerektiğinde 16 sütun veya toplam uzunluklar 900 Byte'i aşmamak üzere birden fazla alanı kapsayan bir index tanımlanabilir.



Özet (devamı)

- **Kapsam (Covering) Index:** Bir sorgunun WHERE kısmı da dahil olmak üzere, seçilen sütunları ile birlikte bir tek index olarak tanımlanmasına Covering Index denir.
- **Parçalı Index:** Parçalı indexler, SQL Server 2005 ile birlikte gelen bir index türü olup farklı fiziksel dosya gruplarına dağıtılmış indexlere verilen addır.
- **Filtreli Index:** Filtreli indexler, bir tabloda yer alan belli sütunlar için bütün kayıtları indexlemek yerine sadece kurala uyan satırları indexlemek amacıyla kullanılabilen ve SQL Server 2008 ile birlikte kullanıma sunulan bir yapıdır.
- **XML Index:** XML Sütunlar üzerinde tanımlanan indexlerdir.
- **Full-Text Index:** Metin ifadelerinin farklı formları ve dil kurallarından bağımsız indexlenmesini sağlar.

DEĞERLENDİRME SORULARI

1. Bir kullanıcının veri tabanındaki kayıtları yalnızca listeleyebilmesi için kullanıcıya hangi yetkinin verilmesi gerekir?
   a) db_datawriter
   b) db_datareader
   c) db_denydatawriter
   d) db_denydatareader
   e) db_denywriter

2. SQL Server açısından index kullanımının en önemli amacı nedir?
   a) Hızlı veri okuma
   b) Daha az veri okuyarak daha kısa sürede sonuç getirme
   c) Daha çok veriyi daha kısa zamanda okuma
   d) Daha çok veriyi daha uzun zamanda okuma
   e) Daha çok veriyi daha kapsamlı okuma

3. Aşağıdakilerden hangisi index tipidir?
   a) UNBOUNDED
   b) NONCLUSTERED
   c) DEVELOPED
   d) GROUPED
   e) UNGROUPED

4. Aşağıdakilerden hangisi bir tablo indexleme alanı olarak seçilen sütundaki verilerin tekrarlanmasını istemiyorsa index özelliği olarak seçmelidir?
   a) Karma
   b) Kapsam
   c) Full-Text
   d) Is Unique
   e) Null-Text

5. Fiziksel dosya gruplarına dağıtılmış olan indexlere verilen ad nedir?
   a) Karma index
   b) Kapsam index
   c) Parçalı index
   d) Sütunlu index
   e) Satırlı index

6. Full-Text indexler hangi amaçla kullanılır?
   a) Bir tabloda yer alan bazı kriterlere uyan verileri indexleme
   b) Farklı fiziksel dosya gruplarında index tutma
   c) Sorguların daha az işlemle daha hızlı çalışması
   d) XML Sütunlar üzerinde sorguların hızlanması
   e) Metin ifadelerinin indeklenmesi

7. FILL FACTOR parametresi ne için kullanılır?
   a) Sorguları doldurmak için
   b) Tabloları doldurmak için
   c) Uç seviye sayfaların doluluk oranını belirlemek için
   d) Veri tabanı doluluk oranını belirlemek için
   e) Veri tabanının boş alanlarını belirlemek için

8. Aşağıdaki yöntemlerden hangisi veri tabanı oluştururken kullanılabilir?
   a) Management Studio ile oluşturulabilir.
   b) Dosya kopyalanarak oluşturulabilir.
   c) Dosya ismi değiştirilerek oluşturulabilir.
   d) TSQL'de INSERT terimi kullanılarak oluşturulabilir.
   e) TSQL'de DROP terimi kullanılarak oluşturulabilir.

9. Management studio ile veri tabanı oluşturulurken aşağıdakilerden hangisini belirtmek gerekmez?
   a) Veri tabanı adı
   b) Tablo adı
   c) Autogrowth yüzdesi
   d) Veri tabanı veri dosyası fiziksel dosya yolu
   e) Veri tabanı Log dosyası fiziksel dosya yolu

10. Aşağıdakilerden hangi işlem db_owner yetkisine sahip bir kullanıcı tarafından yapılabilir?
   a) İşletim sistemi üzerinde kullanıcı oluşturmak
   b) Veri tabanı sunucusunu yeniden başlatmak
   c) Veri tabanı üzerinde tablo oluşturmak
   d) Veri tabanı sunucusuna kullanıcı eklemek
   e) İşletim sistemini yeniden başlatmak

Cevap Anahtarı

1.b, 2.b, 3.d, 4.b, 5.d, 6.c, 7.e, 8.c, 9.a, 10.b

YARARLANILAN KAYNAKLAR

Sql Server Books Online, (2012). 20 Temmuz 2019 tarihinde https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms130214(v=sql.105) adresinden erişildi.

TSQL İLE VERİ TABANI VE TABLOLARI OLUŞTURMAK, ÖZELLİKLERİNİ BELİRLEMEK

Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi
Serdar AYDIN

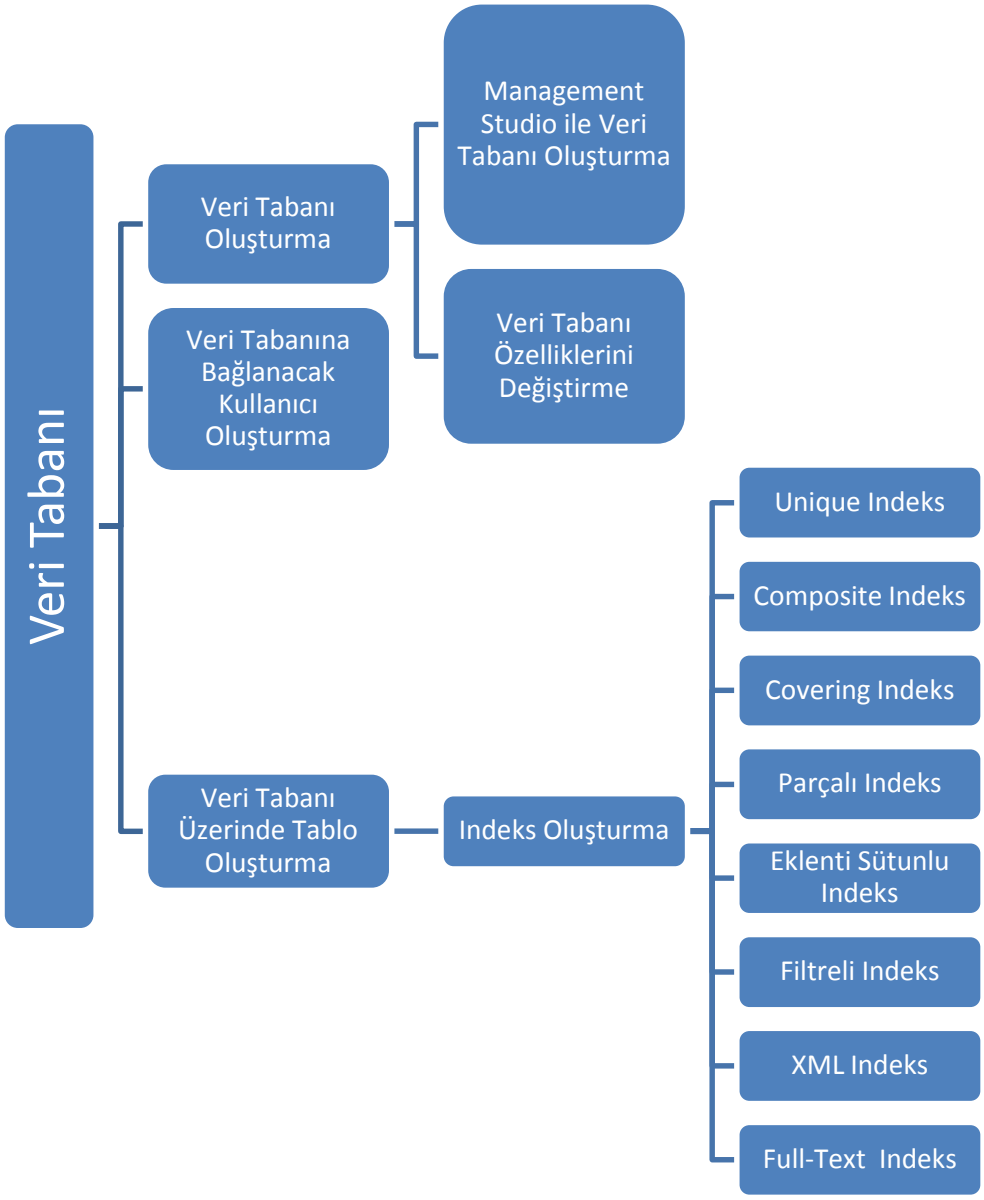
ÜNİTE
5

İÇİNDEKİLER

- SQL Server 2008 R2 Ortamında Tablo Oluşturmak
  - TSQL ile Veri tabanı oluşturmak
  - TSQL ile Tablo oluşturmak
  - TSQL ile Tablo düzenlemek
  - TSQL ile Tablo silmek

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - SQL Server 2008 R2 üzerinde TSQL ile tablo oluşturabilecek,
  - Tablolara çeşitli veri türlerinde alanlar tanımlayabilecek,
  - Arama sihirbazı kullanarak veri türü tanımlamanın nasıl gerçekleştirildiğini kavrayabileceksiniz.


Veri Tabanı
Veri Tabanı Oluşturma
Management Studio ile Veri Tabanı Oluşturma
Veri Tabanı Özelliklerini Değiştirme
Veri Tabanına Bağlanacak Kullanıcı Oluşturma
Veri Tabanı Üzerinde Tablo Oluşturma
Indeks Oluşturma
Unique Indeks
Composite Indeks
Covering Indeks
Parçalı Indeks
Eklenti Sütunlu Indeks
Filtreli Indeks
XML Indeks
Full-Text Indeks

GİRİŞ

Bu bölümde T-SQL ile Veri tabanı oluşturma, tablo oluşturma, tablo düzenleme silme gibi işlemlerin nasıl yapılabileceğine bakacağız. TSQL Microsoft'un veri tabanı sorgulama dilidir. Transact-SQL, SQL Server ve istemci (client) arasında iletişimi sağlayan SQL sorgulama dilinin gelişmiş bir versiyonudur. SQL yapılandırılmış sorgulama dili (Structured Query Language) anlamına gelen Transact Structured Query Language kelimelerinin kısaltmasıdır. T-SQL kullanarak veri tabanına kayıt eklenebilir, silinebilir, güncellenebilir; kullanıcı, eklenebilir, silinebilir, güncellenebilir ya da sorgulama ve raporlama yapılabilir. TSQL bir programlama dili değil sorgu dilidir.

T-SQL ile aşağıdaki işlemler yapılabilir:

- Veri Tabanı oluşturmak,
- Tablo oluşturmak,
- Veri tabanı ve tablolar üzerinde değişiklikler yapmak,
- Kayıt ekleme, silme, güncelleme,
- Verileri Filtrelemek.

T-SQL ile döngü veya mantıksal işlemler yapmak için bir derleyiciye gerek yoktur. Herhangi bir programlama dili öğrenmeden de T-SQL ile tüm amaçlarınıza hitap edecek projeler gerçekleştirebilirsiniz. T-SQL ifadelerini çalıştırabilmek için bir Management Studio ile SQL Server'a erişmeniz gerekir.

TSQL ile Management Studio'ya ihtiyaç duymaksızın herhangi bir SQL istemcisi ile SQL Server üzerinde bütün işlemleri yapabilirsiniz.

SQL deyimleri veri tabanları üzerinde çeşitli işlemleri yerine getirir. Veri tabanından sorgulama yapmak için SELECT, ekleme yapmak için INSERT güncelleme yapmak için UPDATE, silme yapmak için DELETE, yeni tablo oluşturmak için CREATE TABLE gibi komutlara sahiptir. Bu komutlar işlevlerine göre şu şekilde ayrılır.

TSQL ile Management Studio'ya ihtiyaç duymaksızın herhangi bir SQL istemcisi ile SQL Server üzerinde bütün işlemleri yapabilirsiniz.

T-SQL ile Veri Tabanı Oluşturmak

Veri tabanını Management Studio ile oluşturabildiğimiz gibi T-SQL ile de oluşturabiliriz.

Bunun için en basit şekilde,

```
CREATE DATABASE  veri_tabanı_ismi
```

komutunu kullanabiliriz. Bu komutla veri tabanı oluşturduğumuzda model veri tabanı kopyalanacak ve bu kopyanın ismi verdiğiniz isimle değiştirilerek veri tabanı oluşturulacaktır. Veri tabanımızı özelliklerini kendimiz belirterek oluşturmak istediğimizde ise;

```
CREATE DATABASE veri_tabanı_ismi
ON
PRIMARY (NAME=dosya_takma_ismi,
FILENAME=fiziki_dosya_ismi,
SIZE=dosya_boyutu,
MAXSIZE=maksimum_dosya_boyutu,
FILEGROWTH=dosya_artım_miktarı,
LOG ON
(NAME=dosya_takma_ismi,
FILENAME=fiziki_dosya_ismi,
SIZE=dosya_boyutu,
MAXSIZE=maksimum_dosya_boyutu,
FILEGROWTH=dosya_artım_miktarı)
```

T-SQL komutunu kullanmamız gerekir. Bu komutu inceleyelim:

Primary (Birincil) dosya veri tabanı için gerekli olan tüm nesnelerin saklandığı dosyadır. Veri tabanı için birincil dosya mutlaka olmalıdır.

Oluşturulan veri tabanında kullanılan ifadelerin anlamları şöyledir.

- *Veri_tabanı_ismi:* Veri tabanına verilecek isimdir.
- *Dosya_takma_ismi:* Veri tabanındaki dosyalar için belirlenen takma isimdir. İlgili dosyaya erişimde pratiklik sağlar. İşletim sistemince bilinen isimdir.
- *Fiziki_dosya_ismi:* İşletim sisteminde saklanacak dosyanın adını ve yolunu belirtir. Bu isimde işletim sistemince bilinen bir isimdir.
- *Dosya_boyutu:* Veri tabanı tanımlandığındaki boyutunu belirtir. Başlangıçta 1MB'tır. Management Studio'da bu 3MB olarak tanımlanır. Oluşturulacak log dosyası da bunun %10'u kadardır.
- *Maksimum_dosya_boyutu:* Dosyanın maksimum boyutunu belirtir. Belirtilmezse disk dolana kadar dosya artım miktarı kadar artmaya devam eder.
- *Dosya_artım_miktarı:* Dosyanın başlangıçta belirtilen boyutu dolduğunda sistem tarafından boyutu otomatik olarak artırılabilir. Bu aşamada veri tabanının boyunun ne kadar artırılacağı bu parametre ile belirtilir. % oran verilebileceği gibi sabit bir artım boyutu da girilebilir.

Örneğimizde çok fazla veri değişikliği olacağı için veri dosyası ile log dosyası arasındaki oran geleneklere göre üst sınır olan %25'te tutulmuştur.

Log dosyası farklı bir sürücüde olacak şekilde tanımlanmıştır. Bu özellikle performans artırıcı olsun diye yapılmış bir ayarlamadır. Transaction log dosyası ve tempdb'ye ait dosyalar için ayrı birer disk ayarlamak I/O işlemini artırdığı için veri değişikliği ekleme ve silme yapan veri tabanında performansı artıracaktır.

Management Studio'nun aksine TSQL ile veri tabanı oluştururken tüm parametreleri bilmeniz ve eksiksiz girmeniz gerekmektedir.

Veri Tabanına Erişecek Uygulama Kullanıcısını Ayarlamak

Tasarladığımız bu veri tabanına elbette üst katmandan erişilerek C#, VB.NET, Java, Delphi gibi dillerle programlar yazılacak. Peki bu diller içerisinden veri tabanımıza nasıl erişilecek? Elbette ki bir uygulama rolü veya kullanıcısı üstünden.

Bu aşamaya kadar veri tabanını oluşturdu iseniz hemen arkasından, veri tabanına erişecek login ve kullanıcıları oluşturmanız gerekir. Aşağıdaki ifade ile *webuser* adında bir veri tabanı uygulama erişimi için login ve veri tabanı kullanıcısı oluşturabilirsiniz:

```
USE master GO
CREATE LOGIN webuser WITH PASSWORD = 'sifre'
GO
USE ogrencibilgi GO
CREATE USER webuser FOR LOGIN webuser
```

Aynı ayarlamaları, SQL Management Studio kullanarak SQL Server'a bağlandıktan sonra Nesne gezgini (Object Explorer) üstünden de önce Security düğümüne gidip yeni bir login ardından da sırayla Databases, ogrencibilgi veri tabanı, Security düğümlerini tıklayıp yeni veri tabanı kullanıcısı oluşturabilirsiniz. (Ünite 4)

Bütün nesnelerin şemasını (sahibini) tek tek değiştirmek gerekiyordu. Bu probleme, "işten ayrılan eleman sorunu" denmektedir. SQL Server, herhangi bir kullanıcı ile herhangi bir şemanın ilişkilendirilebilmesine olanak sağlar. Böylece şemalar gerçek anlamda nesneleri gruplama işlevini yerine getirir.

Şema yapısı veri tabanı içerisindeki nesneleri erişim izni ve sahiplik atamaları gibi işlevleri kolaylaştırmak için gruplamaya yarar.

Şemaları Yönetmek

Bir veri tabanı içerisinde şema oluşturmak için şu genel ifade kullanılabilir:

```
CREATE SCHEMA schema ismi; (varsa şema içerisine eklenecek öğeler)
```

Örnek

- ogrencibilgi veri tabanı içerisinde katalog adında bir şema oluşturalım ve içerisine bir tablo ekleyelim:
- CREATE SCHEMA ogrencibilgisema
- CREATE TABLE [dbo].[Birey](
  - [ID] [int] IDENTITY(1,1) NOT NULL,
  - [TcNo] [varchar](11) NOT NULL,
  - [Adi] [varchar](50) NULL,
  - [Soyadi] [varchar](50) NULL,
  - [AnneAdi] [varchar](50) NULL,
  - [BabaAdi] [varchar](50) NULL,
  - [Cinsiyeti] [varchar](5) NULL,
- CONSTRAINT [PK_Ogren] PRIMARY KEY CLUSTERED
- [ID] ASC
  - WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY= OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
- ON [PRIMARY]
  - Daha sonra, bu şemayı bir kullanıcının default şeması olarak atamak istersek şu genel ifadeden yararlanabiliriz:
- ALTER USER kullanici ismi
- WITH DEFAULT_SCHEMA = sema ismi

Ardından bir önceki ünitede öğrendiğimiz şekilde webuser kullanıcısına ogrencibilgisema adlı şema içerisindeki veri tabanı öğelerinden seçme yapma yetkisi verelim:

```
GRANT SELECT ON SCHEMA::ogrencibilgisema TO webuser
```

Örnek

- Webuser isimli kullanıcının default şemasını katalog olarak değiştirelim:
- ALTER USER webuser
- WITH DEFAULT_SCHEMA = ogrencibilgisema

Şema tasarımı yapılırken şu kurallar göz önünde bulundurulmalıdır:

- Bir şemanın birden fazla kullanıcı sahibi olabilir. (Windows rolü, veri tabanı kullanıcısı, veri tabanı uygulama rolü, Ahmet, yerine işe başlayan Ali...)
- Birden fazla kullanıcı aynı default şema tanımı ile ilişkilendirilmiş olab...

- Ortak kullanılan nesnelerin sahibi dbo olmak zorunda değildir. Bütün kullanıcıların erişebileceği başka bir genel şema oluşturulup bu amaç için kullanılabilir.
- Bir kullanıcıyı sildiğinizde bu kullanıcıya ait bütün nesneleri silme veya sahibini değiştirme zorunluluğunu ortadan kaldırır.

Veri Tabanını Yönetmek

Bir veri tabanı oluşturulduktan sonra bazı özelliklerinin zaman içerisinde yeniden ayarlanması gerekebilir. Veri tabanı seçeneklerinden erişim ve nesnelerin çalışma seçenekleri gibi ayarlar yapılıdır. Bazı durumlarda veri tabanı dosyalarının boyutunu azaltmak veya artırmak ihtiyacı doğabilir. Aynı şekilde log dosyalarının boyutları ile ilgili ayarlamalar yapmak gerekebilir.

Veri tabanını düzenlemek için ALTER DATABSE komutu kullanılır.

Veri Tabanı Seçeneklerini Ayarlamak

Veri tabanı seçenekleri Management Studio'dan bir veri tabanı sağ tıklanıp özellikler menüsüne geçildikten sonra, options sekmesinden ayarlanabileceği gibi, ALTER DATABASE deyimi ile ayarlanabilir veya bir veri tabanı ilk oluşturulduğu anda burada verilen parametreler de tayin edilebilir.

Genel olarak ALTER DATABASE deyimi, veri tabanı seçeneklerini ayarlarken şu şekilde kullanılır:

```
ALTER DATABASE veritabani ismi
SET secenek durum
```

Hâlihazırdaki veri tabanı seçenek grubu için, parametrelerden hangilerinin geçerli olduğu şu şekilde de öğrenilebilir:

```
SELECT DATABASEPROPERTYEX ('veritabani ismi','ozellik')
```

Tablo 5.1. Veri Tabanı Şeçenekleri

Seçenek Grubu	Seçenek veya	Durum
Otomatik (automatic)	AUTO_SHRINK	Veri tabanının otomatik olarak küçültülmesini ayarlayan özelliktir. Bu özellik seçili iken veri tabanı dosyası ve log dosyası belli aralıklarla sistem tarafından otomatik olarak gereksiz alanları (%25'den fazlası boş olursa) dosyadan
Otomatik	AUTO_CLOSE	Bütün kullanıcılar veri tabanından bağlantıyı kestiğinde, veri tabanı sunucusu otomatik olarak dosyaları bırakır ve doğrudan işletim sistemi ortamında müdahele edilebilir hâle gelir. Canlı uygulamalarda bu seçenek kapalı olmalıdır.
Otomatik	AUTO_CREATE_STATISTICS ve AUTO_UPDATE_STATISTICS	SQL Server WHERE ve JOIN ifadeleri için sorgu süresi-maliyeti kestirirken kullanılmak üzere istatistikler tutar ve bu istatistikleri UPDATE eder. Bu seçeneklerin canlı bir sistem için açık olması performansı artırır.

Durum (state)	RESTRICTED USER	Bu özellik seçildiği anda sistemde login olan kullanıcılar ve db_owner sabit sistem rolüne haiz kullanıcılar dışında diğer kullanıcılar bu veri tabanı dosyasını kullanamaz. Hâlihazırda sisteme login olan kullanıcılar ise sistemden çıkmaları halinde db_owner rolüne sahip değil ise tekrardan veri tabanına erişemez. Yedek indirmek gibi tamir işlemleri esnasında, bu modda çalıştırmak bir zorunluluk olabilir.
Durum	SINGLE USER	Veri tabanı dosyasını aynı anda bir tek kişinin kullanmasını sağlar. Bu komut çalıştırıldığı anda veri tabanın-da birden fazla kişi bağlıysa bu özellik açıldıktan sonra kullanmaya devam ederler ancak bağlantı kesildiği andan itibaren bi r tek kullanıcı kuralı dikkate alınır. Yedek indirmek gibi tamir işlemleri esnasında,
Durum	MULTI USER	Normal olarak bir veri tabanı bu erişim modundadir. Herkes tarafından hakları dâhilinde erişilebilir.
Durum	READ ONLY | READ WRITE	Veritabanı dosyasının sadece okunabilir olmasını sağlayan seçenektir. Bu seçenek seçildiğinde kullanıcılar veri tabanından veri okuma (SELECT) komutlarını hakları doğrultusunda çalıştırabilir ancak DELETE, UPDATE, INSERT gibi DML komutları ile ALTER, CREATE ve DROP gibi DDL komutlarını çalıştıramaz. Tersi READ_WRITE modudur. Bu modda herkes yetkileri dahilinde okur ve yazar. Bir durum belirtilmeden kullanılırlar.
Kurtarma (Recovery)	RECOVERY FULL	Bu seçenekte veri tabanı üstünde yapılan işlemlerin ayrıntılı loglari tutulur. Bir veri kurtarma işleminde, log dosyaları ile birlikte veri tabanı kayıtları da elde edildikten sonra veri tabanı kurtarma işlemi gerçekleştirilebilir. Standart olan kurtarma seçeneği bu seçenektir. Böylece yarıda kalmış transactionlar bile
Kurtarma	BULK LOGGED	Full Recovery Model ile yaklaşık aynı seçenekleri kullanır. Farklı olarak toplu işlemler için daha az log tutar. Örneğin SELECT INTO vb. gibi komutlar için daha kısıtlı bir log tutulur.
ANSI	ANSI_NULL_DEFAULT	Bir tablonun bir sütunu için NULL olabilirlik konusunda bir tanımlama yapılmazsa server nasıl davranacak? ON: belirtilmedi ise NULL kabul edebilir OFF: belirtilmedi ise NULL olamaz. NULL olabilecek sütunlar ayrıca belirtilmeli
ANSI	ANSI_NULLS	NULL değerler içeren sütunların bir eşitlik sorgulamasında, eşittir mi yoksa eşit değildir mi olarak kabul edileceğine karar verir. NULL kıyas sonuçlar için OFF: doğru, ON:yanlış

Bu şekilde durumları kontrol edilebilen daha bir çok parametre vardır. Diğer seçenekler hakkında, bu şekilde bilgi almak için, Books Online'da *DATABASEPROPERTYEX* ifadesini aratarak bilgi edinebilirsiniz.

Öğrenci bilgi veri tabanında, gereksiz dosya boyutlarının otomatik olarak küçültülmesini istiyoruz.

Örnek

- ALTER DATABASE ogrencibilgi SET AUTO_SHRINK ON
- Daha sonra ilgili değişikliğin yapılıp yapılmadığını görüntülemek için
- SELECT DATABASEPROPERTYEX('dukkan','IsAutoShrink')

diyerek durumunu görebiliriz.
ogrencibilgi veri tabanında, bir süre için kimsenin değişiklik yapamamasını istiyoruz.

Örnek

- ALTER DATABASE ogrencibilgi
- SET READ_ONLY

Veri tabanını sadece okunabilir modda çalıştırabilirsiniz.

şeklinde kullanılır.

Veri Tabanı Seviyeli Collation Ayarı Yapmak

SQL Server, veri tabanı seviyeli colation ayarı yapmaya müsaade eder.

Özellikle İngilizce bir Windows sürümü üstünde çalışan ve default olarak İngilizce ayarlanmış bir SQL Server, uygulamalarınızda karakter problemine neden olabilir. Bu türden durumlarda, aşağıdaki şekilde collation ayarı yapabilirsiniz:

```
USE master
GO
ALTER DATABASE dukkan COLLATE Turkish_CI_AS
```

Bu türden bir ayarlamayı çalıştırabilmeniz için, ogrencibilgi veri tabanı içerisinde collation ayarlarına bağlı bulunan nesne olmaması gerekir.

Daha sonra ilgili değişikliği görmek için

```
sp_helpdb 'ogrencibilgi'
```

İfadesini çalıştırarak göz atabilirsiniz.

Yine SQL Server'in İngilizce sistem üstünde kurulu olduğu durumlarda, tarih formatı ile ilgili problemlerin önüne geçmek üzere (MDY yerine DMY), uygulama kullanıcınız için şu ifadeyi de çalıştırabilirsiniz:

```
sp_defaultlanguage 'webuser','Turkish'
```

Seviyeli Collation ayarları ile tarih formatı para birimi gibi formatların SQL tarafından otomatik düzenlenmesini sağlar.

Böylece, bu kullanıcı tarafından yapılan tarih değişimlerinde MDY formatı yerine DMY formatını kabul eder hâle gelecektir. Veritabanına erişen başka Türkçe dil ayarlı kullanıcılar bulunması gerekiyorsa aynı ayarları *webuser* yerine onların adlarını da vererek yapabilirsiniz.

Veri Tabanı Dosyalarının Boyutunu Değiştirmek

Veri tabanında yer alan veri dosyalarının veya Log dosyalarının boyutun'' zaman içerisinde artırmak gerekebilir veya yüksek bir alan verildiği için küçültn

gerekebilir. Bir veri tabanı üstünde bu türden ayarlamaları yaparken, *sysadmin* rolüne sahip olmak veya veri tabanı sahiplik rolüne sahip olmak gerekir.

Veri Tabanı Dosyalarının Boyut Artımını Ayarlamak

Bir veri tabanın boyutu küçük geliyorsa var olan veri dosyasının veya log dosyasının boyutunu artırarak çözüm olabileceği gibi, başka bir veri dosyası veya log dosyası eklemek de çözüm olarak kullanılabilir. Özellikle disk dolmuşsa başvurulacak çözüm, yeni bir dosya tanımlamaktır.

Otomatik artırmaya ayarlama

Bu seçenek, veri tabanının tanımında yapılması gereken FILEGROWTH parametresi ile belirtilir. Daha önceden oluşturulmuş bir veri tabanı için bu seçeneği şu genel ifadeden yola çıkarak düzenleyebiliriz:

```
ALTER DATABASE veritabani ismi
MODIFY FILE (NAME=dosya takma ismi,
SIZE = dosya boyutu,
MAXSIZE = maksimum dosya boyutu,
FILEGROWTH = dosya artim miktari)
```

Veri Tabanlarını Küçültmek (Shrinking)

Veri tabanlarını yönetirken daha önceden ayrılan yerin çok fazla olduğuna karar verdiğimizde, henüz veri tabanı tarafından kullanılmamakta olan alandan vazgeçebiliriz. Başka bir şekilde ifade etmek gerekirse veri tabanının veya dosyalarının boyutunu, veri tabanının asgari kullanımına kadar düşürebiliriz.

Örneğin veri tabanımızın boyutu 40MB olarak tanımlansın. İçerdiği veriler de 4MB ise bu veri tabanının boyutu 4 MB'a kadar düşürülebilir. Ancak daha fazla düşürülemez.

Veri tabanlarını yönetirken daha önceden ayrılan yerin çok fazla olduğuna karar verdiğimizde, henüz veri tabanı tarafından kullanılmamakta olan alandan vazgeçebiliriz.

Veri tabanının veya dosyalarının boyutunu değiştirmek, master veri tabanındaki bilgilerin değişmesini gerektirir. Bu nedenle herhangi bir hata ihtimaline karşılık, veri tabanının boyutu ile ilgili değişikliklerden önce ve sonra master veri tabanının yedeğini almak, herhangi bir riske karşı tavsiye edilir.

Bir veri tabanının boyutunu azaltmak için iki yöntem vardır, otomatik olarak veri tabanının boyutunun küçültülmek veya herhangi bir anda, elle küçültmek. Bu işi otomatik olarak SQL Server'e yaptırmak için, AUTO_SHRINK veri tabanı özelliği

```
ALTER DATABASE veritabani ismi
SET AUTO_SHRINK ON
```

ifadesi kullanılarak ayarlanabilir veya Management Studio'dan, veri tabanı özellikleri ekranından düzenlenebilir.

Herhangi bir anda, veri tabanın boyutu ile ilgili küçültmeye gidilecekse ya doğrudan veri tabanının boyutu küçültülebilir veya veri ya da log dosyalarında

birinin boyutunu küçültmek gerekebilir. Bu iki işlem için iki ayrı ifade kullanılır:

DBCC SHRINKFILE, veri tabanına ait bir dosyanın boyutunu küçültür. Bu ifade de dosyaların takma adlarını kullanarak erişir. Bir dosyanın boyutunun uygun olan en küçük değere indirilmesi için, genel ifade şu şekildedir:

```
DBCC SHRINKFILE(dosya adi, TRUNCATEONLY)
```

Boşalan alanın tamamının işletim sistemine devredilmesi işlemini yerine

getiren bu ifadede, TRUNCATEONLY deyimi isteğe bağlıdır. Bir dosyanın belli bir orana kadar boşaltılması ve boşalan alanın da veri tabanı bünyesinde tutulmaya devam edilmesi için şu genel yapı kullanılır:

```
DBCC SHRINKFILE(dosya adi, yuzde kaca inecek, NOTRUNCATE)
```

Bir dosya içerisinde yer alan verilerin, aynı dosya grubunda yer alan diğer dosyalara aktarılmak suretiyle içinin boşaltılması için, EMPTYFILE seçeneği ile kullanılabilir.

```
DBCC SHRINKFILE(dosya adi, EMPTYFILE)
```

DBCC SHRINKDATABASE: Bir veri tabanının boyutu küçültülür. Ancak bu işlem, bir job olarak ele alındığından, veri tabanının boyutunun küçüldüğünü görmek zaman alabilir. Genel kullanımı şu şekildedir:

Veri tabanlarının kullanmadığı alanları sisteme bırakmasını sağlayarak etkin disk alanı kullanımı sağlanabilir.

```
DBCC SHRINKDATABASE (veritabani ismi,TRUNCATEONLY)
```

Bu ifadede, bir veri tabanı adı verildiğinde kullanılmayan son extend'den başlamak üzere fazladan atanmış olan alan işletim sistemine devredilir. Burada TRUNCATE ONLY isteğe bağlı bir ifadedir. Hiç kullanılmaya da bilir.

Veri tabanından boşaltılan alanın, işletim sistemine devredilmeden, veri tabanı kapsamında tutulmaya devam edilmesi için de şu genel ifade kullanılabilir:

```
DBCC SHRINKDATABASE(veritabani ismi, yuzde kaca inecek, NOTRUNCATE)
```

Burada bir hedef yüzde oranı verilebildiğine dikkat edin.

Veri Tabanlarını Silmek

Pek gerekmese de bazen veri tabanlarını silmek de bir ihtiyaç olabilir. Bu durumda, veri tabanını Management Studio'dan veya DROP DATABASE deyimi ile silebiliriz. Ancak bir veri tabanını silebilmek için bazı şartların sağlanması gerekir:

- Veri tabanının, yedek yükleme, replication, kullanıcı erişimi gibi başka bir işlem tarafından kilitlenmemiş olması gerekir.
- Sizin veri tabanında db_owner veya sysadmin rollerinden birine sahip kullanıcı ile erişebiliyor olmanız gerekir.

Bir veri tabanı silindiği zaman, içerdiği bütün veriler kaybolur. Veri tabanı ile birlikte alınmış yedeklerini de silebilirsiniz veya alınmış yedeklerini bırakabilirsiniz. Bu size bir seçenek olarak sunulmuştur.

Bütün bunlardan sonra, veri tabanı silmek için genel ifade şu şekildedir:

```
DROP DATABASE veritabani ismi
```

Tabanı kapsamında tutulmaya devam edilmesi için de şu genel ifade kullanılabilir: şeklinde kullanılır.

Örnek

- ogrencibilgi adlı veritabanını sileceksek:
- DROP DATABASE ogrencibilgi

Tabloları Oluşturmak

Tablolar oluşturulurken sutünlarda taşınacak verinin türüne göre veri tipinin seçilmesi önemlidir.

SQL Server'da iki şekilde tablo oluşturulabilir: Management Studio'da veri tabanı sağ tıklanarak açılan menüden yeni tablo seçeneği işaretlenebilir veya sorgu ekranında CREATE TABLE ifadesi kullanılarak oluşturulabilir. SQL Server[1] bir tablo için binlerce sütuna müsaade eder. Aynca[1] SQL Server hafızayı page'ler

hâlinde yönettiğinden ve her page için 8060 byte kullanılabilir alan kaldığından, bir tablonun bir satır için sütun uzunlukları toplamının 8060byte'i geçmemesi gerekir. Bu uzunluğa ntext, text ve image alanlar dâhil değildir. Çünkü bu türden veriler BLOB olarak adlandırılır ve farklı yapılarda saklanır. MAX parametresi ile tanımlanan VARCHAR, CHAR, BINARY, VARBINARY tipler için SQL Server LOB veya tablo içerisinde saklama konusunda otomatik ayarlama yapabilir. SQL Server bir veri tabanında iki milyondan fazla tablo saklayabilir.

Bir tablo T-SQL ile şu format kullanılarak oluşturulur:

```
CREATE TABLE tabloAdi
(kolon adil veri tipi[NOT NULL],
 kolon adi2 veri tipi [NOT NULL],
…….                                  )
```

Tablolar Üstünde Değişiklik Yapmak

ALTER Table komutu ile tablo üzerinde düzenleme yapılabilir.

Genel olarak tablo üstünde değişiklik yapmak için ALER TABLE deyimi kullanılır.

Sütun eklemek

Bir tabloya sütun eklemek için şu genel yapıdan faydalanılır:

```
ALTER TABLE tablo ismi
ADD COLUMN sutun ismi sutun özellikleri
```

sutun_ozellikleri parametresi ile veri tipi, gerek varsa, uzunluğu ve sütuna dair daha sonradan öğreneceğimiz tanımlamalar yer alabilir.

Örnek

- Birey Tablosuna Doğum Yılı adında bir sütun ekleyelim:
  - ALTER TABLE [Birey]
  - ADD DogumYili AS varchar(2) NULL

Sütun değiştirmek

Bir tablonun sütunun türünü veya genişliğini değiştirmek için ALTER COLUMN ifadesi kullanılabilir.

```
ALTER TABLE tablo ismi
ALTER COLUMN sutun ismi sutun tanimlari
```

Sütun silmek

Artık işe yaramadığı düşünülen bir sütun silinebilir. Genel ifadesi şu şekildedir:

```
ALTER TABLE tablo ismi
DROP COLUMN sutun ismi sutun tanimlari
```

Tablo silmek

Bir tabloya artık ihtiyaç kalmadığını düşünüyorsanız, silebilirsiniz.

Sadece içerdiği verileri etkin olarak silmek için DELETE veya TRUNCATE deyimini kullanabilirisiniz. Bir tablo silmek için şu genel ifade kullanılır:

```
DROP  TABLE tablo ismi;
```

Geçici tablolar ile ana veri tabanı dosyasında düzenleme yapmaksızın geçici bir tablo üzerinde işlem yapılabilir böylelikle büyük dosyayı düzenleme yerine küçük dosyalar oluşturulup üzerinde çalışılıp silinebilir.

Geçici tablolarla çalışmak

Komplike sorguları adımlara ayırmak için, geçici bir süre kayıtları tutmak üzere ek tablolara ihtiyaç duyabiliriz. Bu tür durumlarda, T-SQL ile geçici tablolar oluşturup onları kullanabiliriz. Geçici tablolar, diğer bütün geçici nesneler gibi tempdb'de saklanır ve kullanıcı çıkış yaptığında veya SQLServer kapatıldığında otomatik olarak silinir. Global ve yerel geçici tablolar tanımlamak mümkündür. Global geçici tablolar, birden fazla bağlantı için erişilebilir durumda iken yerel geçici tablolar sadece oturumu açan kişi tarafından erişilebilir.

Geçici tablolar oluşturmanın iki yolu vardır:

1 . Yöntem: Oturum boyunca geçerli geçici tablolar oluşturmak için kullanılır.

```
CREATE TABLE #tablo adi(Alanl turl[(boyutl)] [[NOT]
NULL][,...])
```

Bu şekilde oluşturulan tablolar oturum kapandığında veya SQL Server durdurulduğunda silinir.

Örnek

- Geçici bir kitaplar tablosu oluşturalım:
- CREATE TABLE #ogrencibilgi(
- ogrenciNo INT,
- ogrenciAdi VARCHAR(55))
- GO
- Daha sonra tabloya, normal bir tabloymuş gibi kayıt ekleyebiliriz:
- INSERT INTO #ogrencibilgi VALUES (1, 'Ali Demir') ya da tabloda yer alan kayıtları
- SELECT * FROM #ogrencibilgi

ile görebiliriz. Kayıtları güncelleyebiliriz ya da silebiliriz. Ancak bu tablo, oturumu kapattığımız anda veya SQLServer kapatıldığı anda silinecektir. Bunun dışında biz de istediğimiz zaman bu tabloyu bildiğimiz yöntemle silebiliriz.

```
DROP TABLE #ogrencibilgi
```

Global bir tablo tanımlamak için tablonun isminin iki adet ## ile başlaması gerekir.

2. Yöntem: Geçici tablo oluşturmanın bir diğer yolu, tempdb adlı veri tabanı dosyasına bir tablo açmaktır. Bu veritabanındaki tablolar, sadece SQLServer kapatıldığında silinir.

Genel yapısı şöyledir:

```
CREATE TABLE tempdb..tablo_adi( Alanl turl[(boyutl)] [[NOT] NULL][,...])
```

Bu yöntemle oluşturulan geçici tablolar, SQL Server durdurulduğu anda kaybolur. Çünkü tempdb adlı veri tabanı SQL server açıldığı anda boşlatacaktır. Bazen, kullanıcı çıkış yaptığı halde geçici tablonun saklanması ihtiyacı olabilir. Bu tür durumlarda, geçici tablo SQL Server kapanmcaya kadar hafızada kalıp SQL

server kapatıldığında silinecekse tempdb'de tablo açma yöntemi kullanılabilir.

Örnek

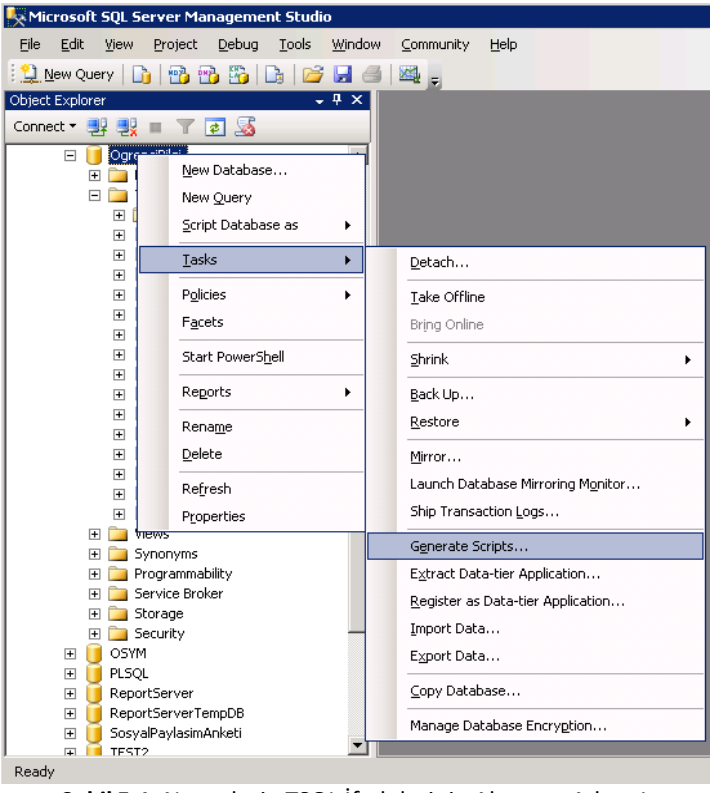
- tempdb..tablo_ismi şeklinde bir kitap adında geçici tablo oluşturalım:
- CREATE TABLE tempdb..ogrencibilgi (ogrenciNo INT, ogrenciAdi VARCHAR(55))

Nesnelerin T-SQL İfadelerini Almak

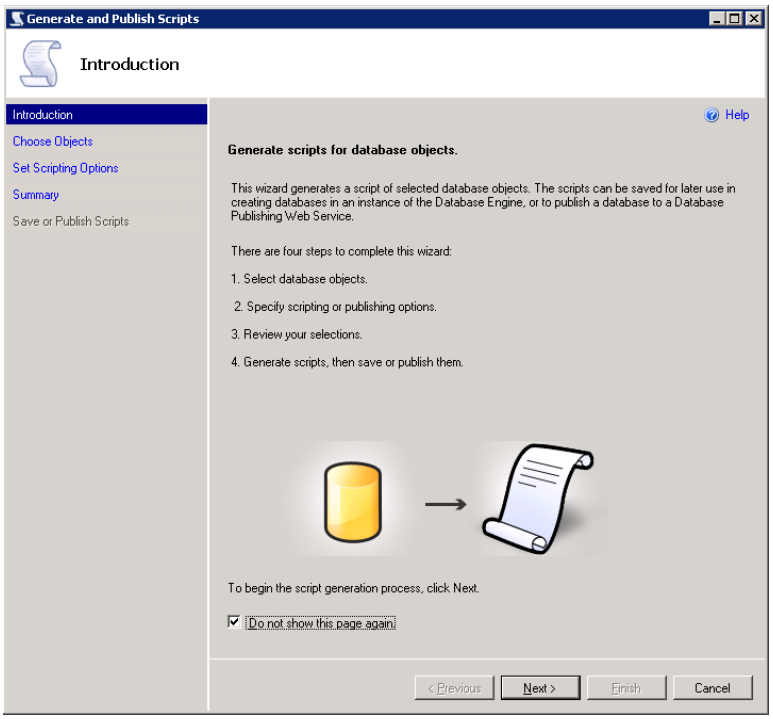
Management Studio ile gerçekleştirilen tüm işlemlerin TSQL karşılıkları alınabilir. Veri tabanı- tablo yapı ve verileri TSQL ifadesi olarak alınabilir.

Bir proje tamamlandıktan sonra, taşınması gerekiyor olabilir. Bu tür durumlarda genellikle Integration Services (eski adı ile DTS) birbiri ile aynı ağ üzerindeki sunucular arasında veri alışverişi için kullanılabilir tekniklerdir. Ancak durum her zaman böyle olmayabilir ya da geliştirilen uygulama, bir paket hâline getirilip dağıtılıyor olabilir. Bu türden durumlarda, veri tabanını oluşturan nesnelerin Transact-SQL ifadelerini bir düz metin dosyasında saklayıp daha sonra bu metin bir şekilde çalıştırılarak aynı şemanın başka bir sunuya aktarılması sağlanabilir.

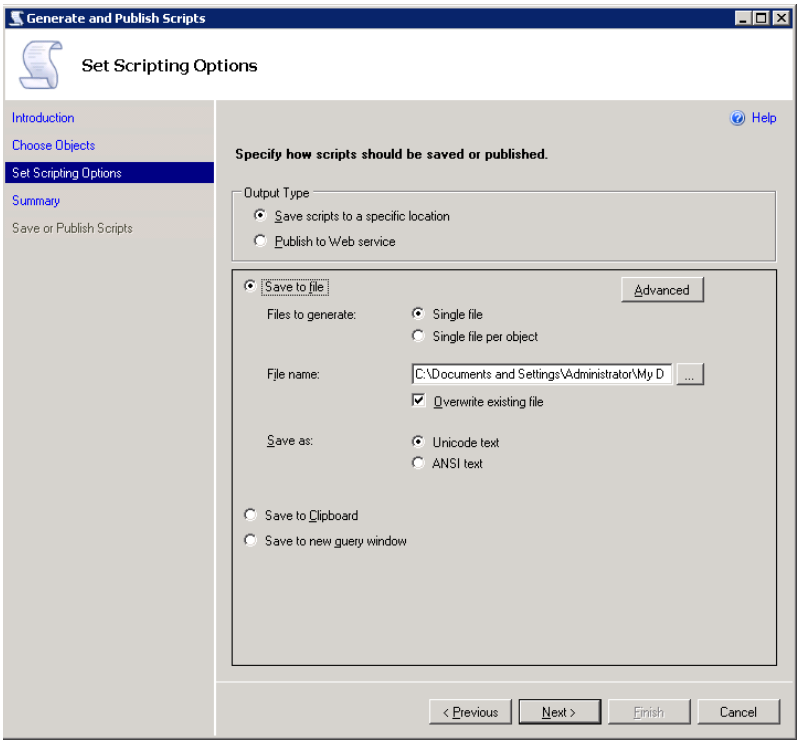
Management Studio ile bir veri tabanının Transact-SQL scriptini oluşturmak oldukça kolaydır. Veri tabanı üstüne sağ tıklandıktan sonra, Tasks \ Generate Scripts menüsünden “Script Wizard'ı başlatabilirsiniz.



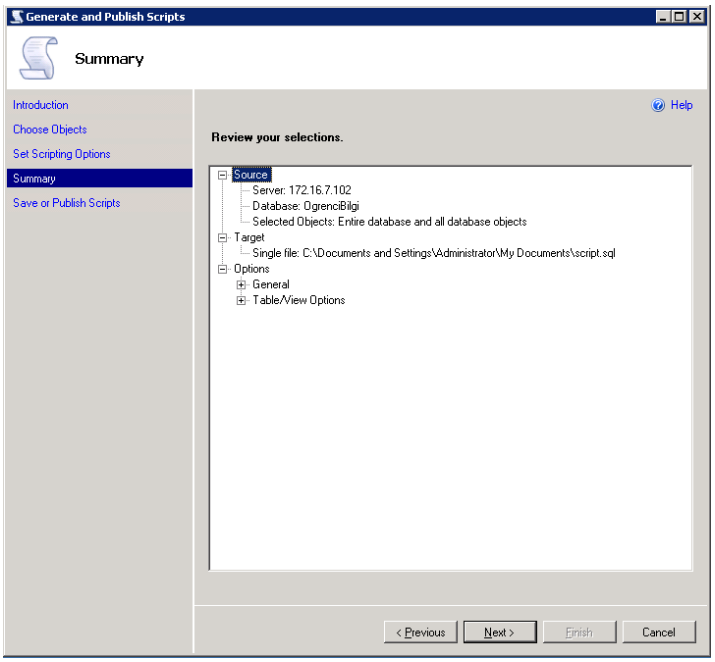
Şekil 5.1. Nesnelerin TSQL İfadelerinin Alınması Adım-1



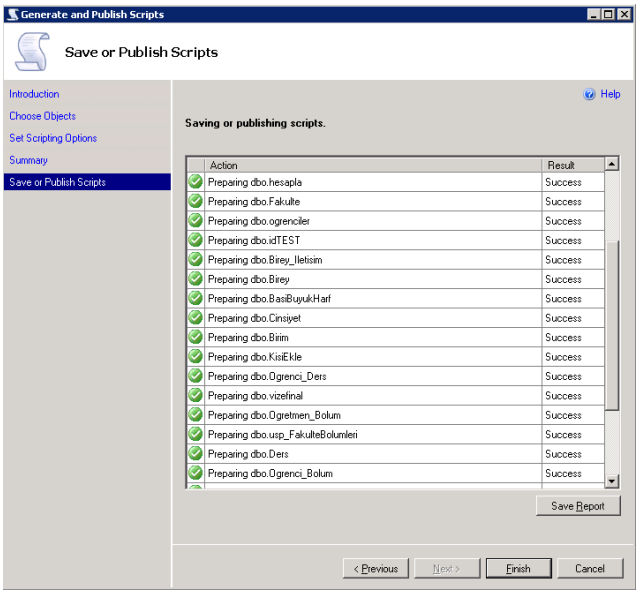
Şekil 5.2. Nesnelerin TSQL İfadelerinin Alınması Adım-2



Şekil 5.3. Nesnelerin TSQL ile İfadelerinin Alınması Adım-3



Şekil 5.4. Nesnelerin TSQL İfadelerinin Alınması Adım-4



Şekil 5.5. Nesnelerin TSQL İfadelerinin Alınması Adım-4

Özet

- **TSQL**
- Transact-SQL, SQL Server ve istemci (client) arasında iletişimi sağlayan SQL sorgulama dilinin gelişmiş bir versiyonudur. SQL Yapılandırılmış sorgulama dili (Structured Query Language) anlamına gelen Transact Structured Query Language kelimelerinin kısaltmasıdır. T-SQL kullanarak veri tabanına kayıt eklenebilir, silinebilir, güncellenebilir; kullanıcı, eklenebilir silinebilir, güncellenebilir ya da sorgulama ve raporlama yapılabilir. TSQL bir programlama dili değil sorgu dilidir.
  - T-SQL ile aşağıdaki işlemler yapılabilir:
  - Veri Tabanı oluşturmak
  - Tablo oluşturmak
  - Veri tabanı ve tablolar üzerinde değişiklikler yapmak
  - Kayıt ekleme, silme, güncelleme
  - Verileri Filtrelemek
- T-SQL ile döngü veya mantıksal işlemler yapmak için bir derleyiciye gerek yoktur. Herhangi bir programlama dili öğrenmeden de T-SQL ile tüm amaçlarınıza hitap edecek projeler gerçekleştirebilirsiniz.
- **TSQL ile Veri Tabanı Oluşturmak**
- "CREATE DATABASE veri_tabanı_ismi " komutu ile veri tabanı oluşturabilirsiniz.
- **Veri Tabanına Erişecek Uygulama Kullanıcısını Ayarlamak**
- "CREATE LOGIN webuser WITH PASSWORD = 'sifre' " komutu ile SQL SERVER üzerinde kullanıcı oluşturabilirsiniz.
- **Veri Tabanı Özelliklerini Değiştirmek**
- "ALTER DATABASE veritabani ismi SET secenek durum" komutu ile veri tabanı özellikleri değiştirilebilir.
- **Veri Tabanını Silmek**
- "DROP DATABASE ogrencibilgi" komutuyla veri tabanı silinebilir.
- **Tablo Oluşturmak**
- "CREATE TABLE tabloAdi (kolon adil veri tipi[NOT NULL], kolon adi2 veri tipi [NOT NULL], ……)" komutuyla veri tabanı üzerinde tablo oluşturulabilir.
- **Var olan tabloya sütun eklemek**
- "ALTER TABLE tablo ismi ADD COLUMN sutun ismi sutun özellikleri " komutuyla tabloya yeni sütun ekleneiblir.
- **Varolan sütunları değiştirmek**
- "ALTER TABLE tablo ismi ALTER COLUMN sutun ismi sutun tanimlari" komutuyla sütunlar değiştirilebilir.
- **Varolan sütunları silmek**
- "ALTER TABLE tablo ismi DROP COLUMN sutun ismi sutun tanimlari" komutuyla sütunlar değiştirilebilir.
- Geçici tablolarla çalışmak
- Komplike sorguları adımlara ayırmak için, geçici bir süre kayıtları tutmak üzere ek tablolara ihtiyaç duyabiliriz. Bu tür durumlarda, T-SQL ile geçici tablolar oluşturup onları kullanabiliriz.
  - 1.Yöntem: Oturum boyunca geçerli geçici tablolar oluşturmak için kullanılır.
    - CREATE TABLE #tablo adi(Alanl turl[(boyutl)] [[NOT] NULL][,...])
  - 2.Yöntem: Geçici tablo oluşturmanın bir diğer yolu, tempdb adlı veri tabanı dosyasına bir tablo açmaktır. Bu veritabanındaki tablolar, sadece SQLServer kapatıldığında silinir.
    - CREATE TABLE tempdb..tablo_adi(Alanl turl[(boyutl)] [[NOT] NULL][,...])

DEĞERLENDİRME SORULARI

1. TSQL ile tablo oluşturma komutu aşağıdakilerden hangidir?
   a) CREATE TABLE
   b) CREATE DATABASE
   c) ADD TABLE
   d) ADD DATABASE
   e) ALTER TABLE

2. TSQL ile tablo oluştururken aşağıdakilerden hangisi zorunludur?
   a) Veri tabanı ismi
   b) Tablo ismi
   c) Veri tabanı kullanıcı adı
   d) Tablo verisinin saklanacağı dosya
   e) Tablo verisinin saklanacağı dosya boyutu

3. Tablodan bir kayıt silmek için hangi komut kullanılır?
   a) ADD
   b) CREATE
   c) INSERT
   d) DROP
   e) DELETE

4. SQL Serverda oluşturulmuş bir tablonun bir sütununu değiştirmek için hangi komut kullanılır?
   a) TABLE <tablo ismi> DROP COLUMN <kolon ismi>
   b) ALTER TABLE <tablo ismi> DELETE COLUMN <kolon ismi>
   c) TABLE <tablo ismi> DELETE COLUMN <kolon ismi>
   d) ALTER TABLE <tablo ismi> ALTER COLUMN <kolon ismi>
   e) ALTER TABLE <tablo ismi> DROP COLUMN <kolon ismi>

5. Aşağıdakilerden hangisi T-SQL ile yapılabilir?
   a) Sunucu oluşturmak
   b) Web sitesi tasarlamak
   c) Veri tabanı oluşturmak
   d) Windows formu oluşturmak
   e) Program derlemek

6. Aşağıdakilerden hangisi T-SQL ile yapılamaz?
   a) Kayıt silinebilir.
   b) Kayıt eklenebilir.
   c) Form oluşturulabilir.
   d) Tablo üzerinde değişiklik yapılabilir.
   e) Tablo oluşturulabilir.

7. Aşağıdakilerden hangisi veri tabanında çeşitli işlemleri yerine getiren komutlardan biri değildir?
   a) DELETE
   b) INSERT
   c) UPDATE
   d) ERASE
   e) CREATE

8. Grant komutu ne için kullanılır?
   a) Veri tabanı kullanıcısını düzenler.
   b) Veri tabanını düzenler.
   c) Tabloya erişecek kullanıcı haklarını düzenler.
   d) Veri tabanına erişecek kullanıcı şifresini düzenler.
   e) Veri tabına erişecek kullanıcı haklarını düzenler.

9. Veri tabanına tablo silmek için hangi TSQL komutu kullanılır?
   a) DROP Database
   b) DROP User
   c) DROP Table
   d) DROP Login
   e) DROP tablo

10. Veri tabanı kullanıcı oluşturulurken hangi TSQL komutu kullanılır?
    a) Create Database
    b) Create File
    c) Alter Database
    d) Create User
    e) Create Login

,

Cevap Anahtarı

1.a, 2.b, 3.e, 4.d, 5.c, 6.c, 7.d, 8.e, 9.c

YARARLANILAN KAYNAKLAR

Sql Server Books Online, (2012). 20 Temmuz 2019 tarihinde https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms130214(v=sql.105) adresinden erişildi.

SORGU OLUŞTURMAK VE ÇEŞİTLERİNİ KULLANMAK-1

İÇİNDEKİLER

- Tabloyu güncellemek
- Select deyiminin yazım kuralları
- Sütunların sınırlandırılması
- Satırların sınırlandırılması
- Sıralama işlemleri

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Bir tabloya sorgu ile sütun ekleyip güncelleyebilecek,
  - Basit bir select cümlesinin yazımını ve bir tablodaki verilerin select deyimi kullanılarak listelenmesini kavrayabilecek,
  - Tablodan istenilen sütunların ve satırların görüntülenmesini sağlayabilecek,
  - Karşılaştırmada kullanılan bazı operatörleri ve fonksiyonları öğrenebilecek,
  - Null değerler üzerinde sorgulama yapabilecek,
  - Tablodaki verilerin sıralanarak listelenmesi hakkında bilgi sahibi

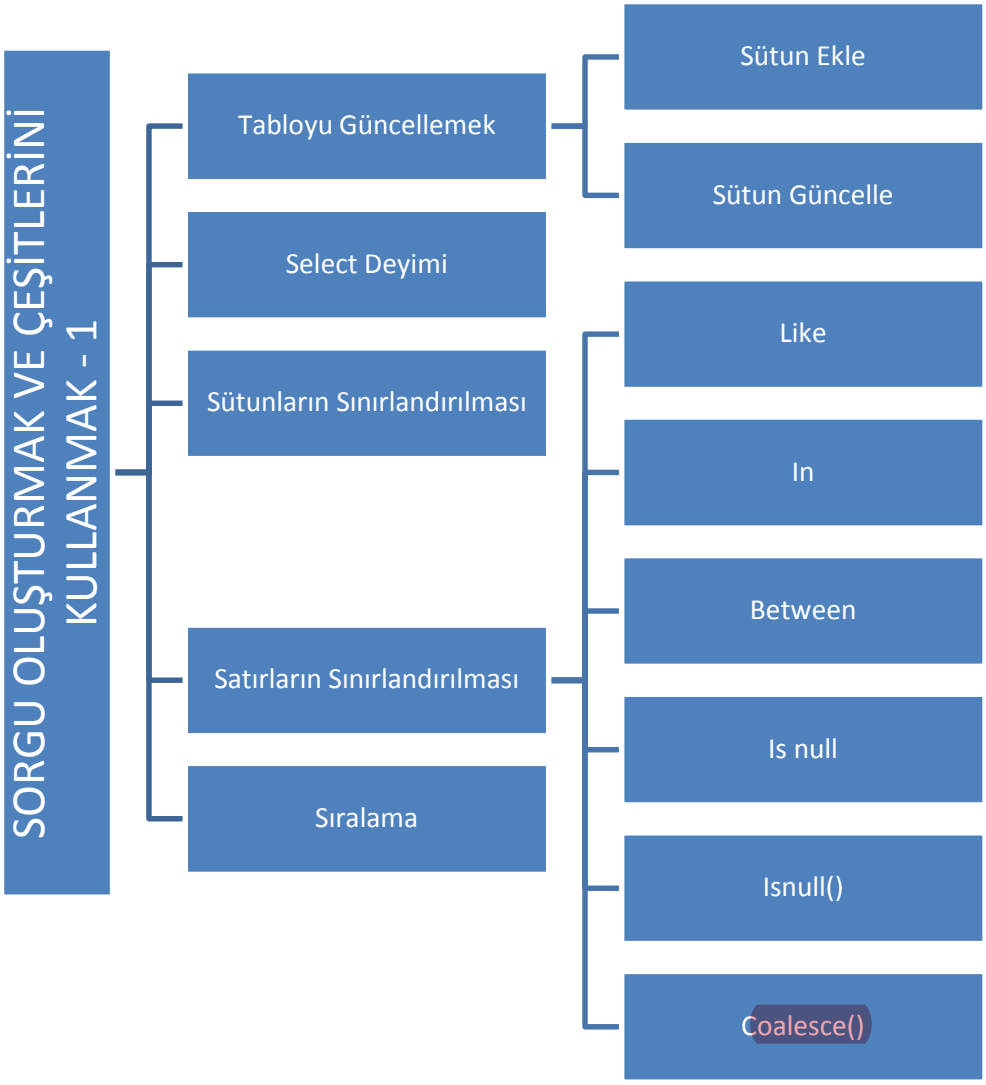
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi Sinan KUL

ÜNİTE

6


SORGU OLUŞTURMAK VE ÇEŞİTLERİNİ KULLANMAK - 1
Tabloyu Güncellemek
Sütun Ekle
Sütun Güncelle
Select Deyimi
Sütunların Sınırlandırılması
Satırların Sınırlandırılması
Like
In
Between
Is null
Isnull()
Coalesce()
Sıralama

GİRİŞ

Önceki ünitede SQL dilinin genel yapısından bahsedilmişti, T-SQL sorguları ile veri tabanının ve tabloların nasıl oluşturulacağı ve mevcut tabloların nasıl ortadan kaldırılacağı gösterilmişti.

Bu ünitede ise tablodaki sütun özelliklerinin değiştirilmesi ve tabloya yeni sütun eklenmesi için gerekli TSQL kodları gösterilecektir. Bu üniteyle birlikte ayrıca en sık kullanılan SQL deyimi olan "Select" deyiminin yazım kuralları gösterilerek SQL ile sorgu oluşturmanın eğlenceli dünyasına giriş yapılacaktır.

"Select" deyimi kullanılarak tablolardan satır ve sütunlar halinde verilerin çekilerek nasıl esnek bir biçimde görüntülenebileceği öğretilecektir. Bu bağlamda tablodaki bazı sütunların istenen sırada görüntülenebileceği sütunların sınırlandırılması konu başlığında incelenirken tablodaki belirli koşula veya koşullara uyan bazı kayıtların filtrelenmesi ise satırların sınırlandırılması konu başlığında incelenecektir. Satırların sınırlandırılmasında kullanılan koşullar, ilgili sütunlar ve değerlerin karşılaştırılmasını sağlayan operatörler ile oluşturulurken birden fazla koşul ise mantıksal operatörler ile birbirine bağlanmaktadır. Dolayısıyla bu ünitede karşılaştırma operatörleri ve mantıksal operatörler (And, Or ve Not) tanıtıldıktan sonra kullanımlarına örnekler verilecektir.

SQL'de diğer dillerde pek karşılığı bulunmayan karşılaştırma operatörleri ise ayrı alt başlıklar ile incelenecektir. Metinsel ifadeler içinde joker karakterler kullanarak arama yapılmasını sağlayan "like" operatörü, aralık sorgulamada kullanılan "between" operatörü, liste içinden arama yapmayı sağlayan "in" Operatörü ve herhangi bir hücreye değer girilip girilmediğini kontrol eden "is null" operatörü bu kapsamda incelenecektir. Null değerlerle çalışmayı kolaylaştıran Isnull ve Coalesce fonksiyonları ise kapsamın biraz dışında olmasına rağmen konuyla irtibatlı olması dolayısıyla üniteye dâhil edildi.

Bu ünitede son olarak verilerin belirli alanlara göre artan veya azalan olarak sıralanması ve sıralama sonrası en üstte kalan birkaç kaydın görüntülenebilmesi için sorgu kodlarının nasıl yazılacağı gösterilecektir.

Kısacası veri tabanındaki tablolardan değer okurken aranan değerin koordinatını tam olarak verebilmek ve verileri sıralı olarak listelemeyi öğrenmek için bu üniteyi dikkatlice okumanız gerekmektedir. Yeri gelmişken belirtmekte fayda görüyorum, SQL sorguları yazabilmek için SQL yazım kurallarını bilmek yetmez, sorgu yazabilme mantığının geliştirilebilmesi için bol uygulama yapılması gerekmektedir.

TABLOYU GÜNCELLEMEK

Tablo oluşturulduktan sonra tablonun yapısında (şemasında) değişiklik yapılabilmektedir. Böylelikle tabloya yeni sütunlar eklenebildiği gibi tablodaki mevcut sütunların isimleri, veri türleri ve kısıtları (constraint) güncellenebilmektedir. Bu bağlamda tablo yapısının güncellenebilmesi için SQL'de kullanılan komut "Alter Table" komutudur. Herhangi bir tabloya sütun eklemek için

mesela "Alter Table" (tabloyu değiştir) deyimi ile birlikte "Add" (ekle) deyimi kullanılmaktadır (Adar, 2012).

Sütun Eklemek

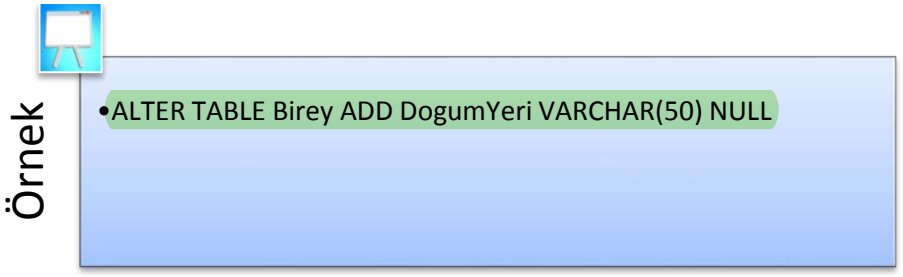
Herhangi bir tabloya sütun eklemek için "Alter Table" (tabloyu değiştir) deyimi ile birlikte "Add" (ekle) deyimi kullanılmaktadır. "Alter Table" deyiminden sonra yapısı güncellenecek tablo adı yazılır; "Add" deyiminden sonra ise eklenecek sütun adı ve veri türü belirtilir (Adar, 2012). Cümlenin sonunda ayrıca zorunlu olmamakla birlikte ilgili sütunun "Null" geçilebileceği veya geçilemeyeceğini belirten bir ifade yani bir kısıt (contraint) bulunabilir. Bu ifade "Null" ise sütun "Null" değer içerebilir demektir. "Not Null" ise ilgili sütunun "Null" geçilemeyeceğini ve mutlaka değer girilmesi gerekliliğini belirtmektedir. Bu ifadenin kullanılmadığı durumda ise kısıtlamanın bulunmadığı yani ilgili sütunun "Null" geçilebileceği anlaşılır.

Veri tabanı nesnesinin özellikleri değiştirilirken "Alter" deyimi kullanılır.

Söz Dizimi:

ALTER TABLE *Tablo_Adı* ADD *Alan_Adı* [*Alan_Türü*] [NULL|NOT NULL]

Örnek olarak "Birey" tablosuna bir alan ekleyelim:



Yukarıdaki SQL kodu ile "Birey" tablosuna "DogumYeri" adında ve "varchar" türünde ve 50 karakter uzunluğunda bir sütun eklenmektedir ve "null" (boş) değer içerebileceği belirtilmektedir. "Birey" tablo şemasının son hâli aşağıdaki gibidir:

	Column Name	Data Type	Allow Nulls
🔑	ID	int	☐
	TcNo	varchar(11)	☑
	Adi	varchar(50)	☑
	Soyadi	varchar(50)	☑
	AnneAdi	varchar(50)	☑
	BabaAdi	varchar(50)	☑
	Cinsiyeti	varchar(5)	☑
	DogumYeri	varchar(50)	☑

Şekil 6.1. Birey Tablosunun "Dogumyeri" Alanı Eklenmiş Hali

Şimdi de bir tabloya bir SQL cümlesi ile birden fazla sütun eklenmesini görelim. Örnek olması için "Birey" tablosuna "Varchar" türünde ve 50 karakter uzunluğunda "DoğumYeri" alanı ve "Date" türünde doğum tarihi sütunlarını ekleyelim.

Null olup olmayacağı belirtilmezse, null kabul eder.

Örnek

- ALTER TABLE Birey ADD (DogumYeri VARCHAR(50) NULL,
- DogumTarihi DATE)

Yukarıdaki SQL kodunda da görüleceği üzere "Add" deyiminden sonra virgüller ile ayrılmış kolon tanımlamaları parantezler içine alınmaktadır. Örnekte ayrıca "DogumTarihi" sütunu tanımlanırken "Null" olup olmayacağı belirtilmediği halde Şekil 6.2.de de görüleceği üzere varsayılan olarak "Null" değer içerebilir olarak yorumlanmıştır.

Column Name	Data Type	Allow Nulls
ID	int	☐
TcNo	varchar(11)	☑
Adi	varchar(50)	☑
Soyadi	varchar(50)	☑
AnneAdi	varchar(50)	☑
BabaAdi	varchar(50)	☑
Cinsiyeti	varchar(5)	☑
DogumYeri	varchar(50)	☑
DogumTarihi	date	☑

Şekil 6.2. Birey Tablosunun "Dogumyeri" Ve "Dogumtarihi" Alanları Eklenmiş Hâli

Sütun Güncellemek

Veri tabanı tablosu oluşturulduktan sonra tablodaki sütunların veri tiplerinin ve/veya kısıtların daha sonra güncellenmesi gerekebilmektedir. Tablodaki herhangi bir sütunun veri tipi ve constraint özellikleri değiştirilirken "Alter Column" deyimi kullanılmaktadır. Alter Column deyiminden sonra özelliği değiştirilecek olan sütun adı belirtilir ve sonrasında yeni veri türü ve "Null" olup olmama durumu belirtilir.

Söz Dizimi:

ALTER TABLE *Tablo_Adı* ALTER COLUMN *Alan_Adı* [*Alan_Türü*] [NULL|NOT NULL]

Tablonun sütunları değiştirilirken "Alter Column" deyimi kullanılır.

Örnek olarak "Birey" tablosundaki "TcNo" alanının veri tipini varchar (11) olarak değiştirelim ve "Null" değer içermeyeceğini belirtelim.

Örnek

- ALTER TABLE Birey ALTER COLUMN TcNo VARCHAR(11) NOT NULL

Birey tablosunun son hali Şekil 6.3.teki gibi olmaktadır.

	Column Name	Data Type	Allow Nulls
🔑	ID	int	☐
	TcNo	varchar(11)	☐
	Adi	varchar(50)	☑
	Soyadi	varchar(50)	☑
	Cinsiyet	int	☑
	AnneAdi	varchar(50)	☑
	BabaAdi	varchar(50)	☑
	DogumYeri	varchar(50)	☑
	DogumTarihi	date	☑

Şekil 6.3. Birey Tablosunun Son Hâli

SELECT DEYİMİNİN YAZIM KURALLARI

"Select" deyimi, veri tabanında bulunan veri kaynaklarından (tablo ve görünüm gibi) verileri sorgulayarak seçmek ve listelemek için kullanılmaktadır. "Select" deyimi ile ilgili tablodaki veriler sadece okunmaktadır; veriler üzerinde herhangi bir değişiklik (ekleme ve silme gibi) yapılmamaktadır.

En basit kullanımıyla "Select" deyimi şu şekilde yazılabilmektedir:

Söz Dizimi:

SELECT S*elect_Listesi* FROM T*ablo_Listesi*

Bütün sütun adları yerine sadece "*" karakteri kullanılır.

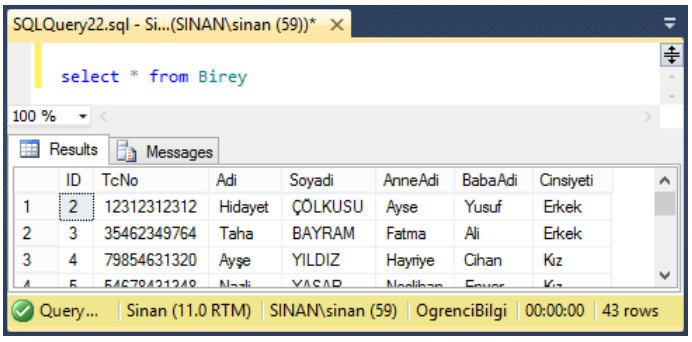
Select deyiminden hemen sonra kullanılan select listesi ile görüntülenmek istenen sütun adları yazılırken "From" ifadesinden sonra kullanılan tablo listesinde verilerin bulunduğu tablo adları yer almaktadır. Birden fazla sütun ve tablo adı birlikte kullanıldığında ise sütun adları ve tablo adları arasına virgül konulmaktadır.

Tabloda yer alan bütün sütun adlarının listelenmesi istenirse sütun adlarını tek tek yazmak yerine, "*" karakteri kullanılabilir. Bu kullanım ile tablo isimlerinin yazılış sırasına ve sütun adlarının tablolardaki yer alış sırasına göre listelendiğini belirtmek gerekir. Ayrıca ilgili tablodaki satırlar tablodaki kaydediliş sırasına göre listelenmektedir. Örnek olarak "Birey" tablosundaki kayıtları listeleyelim.

Örnek

```
SELECT * FROM Birey
```

Sorgu cümlesi çalıştırıldığında "Results" (sonuçlar) sekmesinde "Birey" tablosundaki kayıtlar listelenmektedir (Şekil 6.4). Ekranın sağ alt köşesinde sorgu ile kaç satır döndüğü bilgisi, onun hemen solunda sorgu süresi (mili saniye cinsinden) görüntülenmektedir.

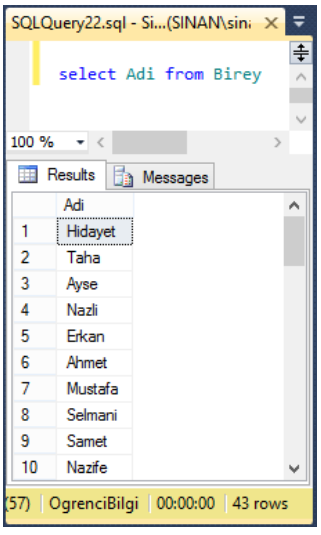


Şekil 6.4. Birey Tablosundaki Kayıtların Listelenmesi

SÜTUNLARIN SINIRLANDIRILMASI

Tablodaki kayıtlar listelenirken sadece belirli sütun adlarının görüntülenebilmesi için "Select" ifadesinden sonra ilgili sütun adlarının yazılması gerektiğini söylemiştik. Örnek olarak "Birey" tablosunda yer alan bireylerin adlarını listeleyelim. Bunun için "Select" deyiminden sonra ve "From" deyiminden önce sadece "Adi" yazılmalıdır.

Şekil 6.5.te de görüleceği üzere "Results" sekmesinde sadece "Adi" sütunu görüntülenmektedir.



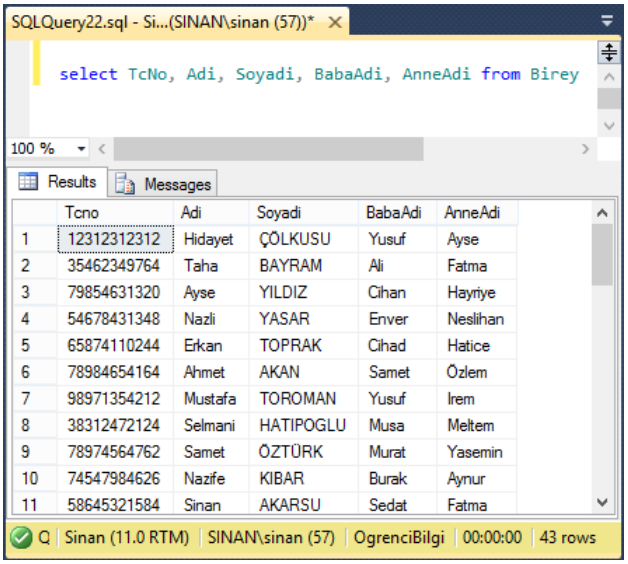
Şekil 6.5. Birey Adlarının Listelenmesi

Sütun adları yazıldığı sırada listelenmektedir.

Şimdi de "Birey" tablosundaki kayıtları listelerken TC kimlik numarası, adı, soyadı, baba adı ve anne adı alanları ile sorguyu sınırlayalım. "Select" deyiminden sonra ilgili sütun adlarının aralarına virgül konarak kullanılmaktadır.

Örnek

•SELECT TcNo, Adi, Soyadi, BabaAdi, AnneAdi FROM Birey



	Tcno	Adi	Soyadi	BabaAdi	AnneAdi
1	12312312312	Hidayet	ÇOLKUSU	Yusuf	Ayse
2	35462349764	Taha	BAYRAM	Ali	Fatma
3	79854631320	Ayse	YILDIZ	Cihan	Hayriye
4	54678431348	Nazli	YASAR	Enver	Neslihan
5	65874110244	Erkan	TOPRAK	Cihad	Hatice
6	78984654164	Ahmet	AKAN	Samet	Özlem
7	98971354212	Mustafa	TOROMAN	Yusuf	Irem
8	38312472124	Selmani	HATIPOGLU	Musa	Meltem
9	78974564762	Samet	ÖZTÜRK	Murat	Yasemin
10	74547984626	Nazife	KIBAR	Burak	Aynur
11	58645321584	Sinan	AKARSU	Sedat	Fatma

Şekil 6.6. Birey Tablosundan Bazı Bilgilerin Listelenmesi

Select ifadesindeki sütun isimlerinin yazıldığı sırada ve kayıtların tablodaki fiziksel konumlarına göre listelendiğini görebilirsiniz.

SATIRLARIN SINIRLANDIRILMASI

SQL sorgu sonuçlarının sınırlandırılarak ilgili bazı kayıtların listelenmesi için özel bir koşul belirtilebilmektedir. Böylece sadece koşula uyan kayıtlar listelenecektir. Bu bağlamda, satırları filtreleyen koşulları belirtmek için "WHERE" operatöründen faydalanılmaktadır (Gözüdeli, 2010).

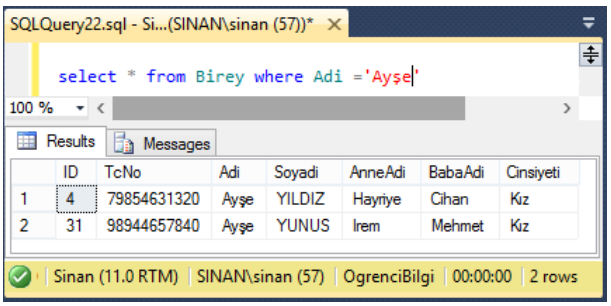
Veriler satır bazında filtrelenirken "Where" deyimi kullanılır.

Söz Dizimi:

SELECT *Select_Listesi* FROM *Tablo_Listesi* WHERE *Koşullar*

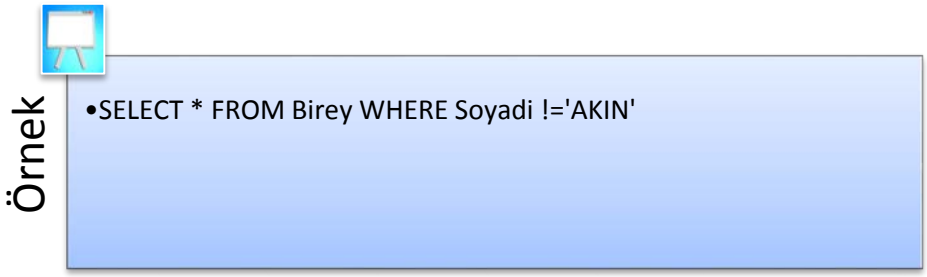
Örnek

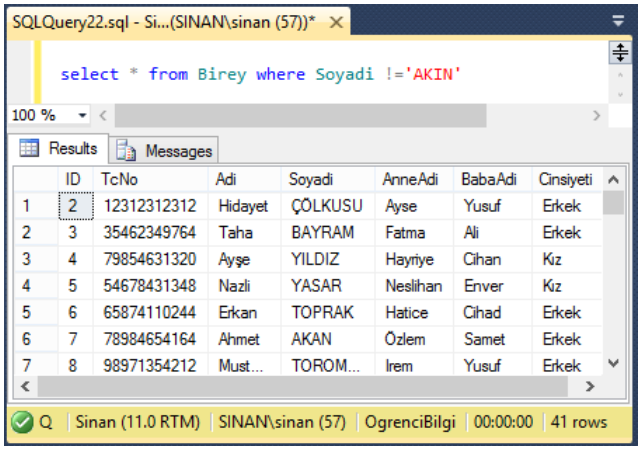
•SELECT * FROM Birey WHERE Adi ='Ayşe'



Şekil 6.7. Adı "Ayşe" Olanların Listelenmesi (2 Kayıt)

Şekil 6.7.deki örnek ile adı "Ayşe" olanlar listelendi. Sütun adı ile değer arasında "=" (eşittir) operatörünün kullanıldığını gördünüz. Şimdi de soyadı "AKIN" olmayan bireyleri listeleyelim:





Şekil 6.8. Soyadı "AKIN" Olmayanların Listelenmesi. (41 Kayıt)

Şekil 6.8.de eşit olmama durumunu ifade etmek için sütun adı ile değer arasında "!=" operatörü kullanıldı. Koşul tanımında kullanılan diğer tüm karşılaştırma operatörleri aşağıdaki tabloda (Tablo 6.1) sunulmuştur. Tabloda eşit olmama durumu için iki farklı operatör kullanılmaktadır. Uygulamada bunlar arasında bir fark yoktur. Benzer bir biçimde ">=" (büyük eşit) operatörü, "!<" (küçük değil) operatörüyle eşdeğer; "<=" (küçük eşit) operatörü ise "!>" (büyük değil) operatörüyle eşdeğer kullanıma sahiptir.

Tablo 6.1. Karşılaştırma operatörleri

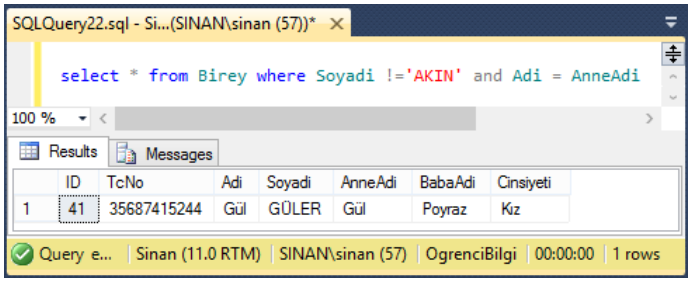
Operatör	Anlamı
=	Eşit
!=	Eşit değil
<>	Eşit değil
>	Büyük
!>	Büyük değil
<	Küçük
!<	Küçük değil
>=	Büyük eşit
<=	Küçük eşit
LIKE	Karakter türde karşılaştırma yapar

Operatörler performansları bakımından kıyaslandığında en performanslı kullanıma sahip olan operatör "=" (eşittir) operatörüdür. Kullanıldığı sorguyu en yavaş sonuçlandıran operatör ise "!=", "<>" (eşit değil) operatörleridir. Karşılaştırma operatörlerinin performanslarıyla ilgili değinilecek son husus bu operatörlerin sayısal değerlerle daha hızlı çalışmasıdır.

Satırları sınırlandırırken birden fazla koşul ifadesi kullanılacaksa bunlar arasına ya "And" (ve) veya "Or" (veya) operatörünün kullanılması beklenmektedir. "And" operatörü kullanıldığında iki koşulun aynı anda doğru olması durumu aranırken, "Or" operatöründe koşullardan sadece birinin doğru olması sonucun doğru olması için yeterlidir.

Örnek

```
•SELECT * FROM Birey WHERE Soyadi !='AKIN' AND Adi=AnneAdi
```



Şekil 6.9. Soyadı "AKIN" Olmayan Ve Adı Annesinin Adı İle Aynı Olanların Listelenmesi.

"Like" operatörü karakter türündeki değerler için kullanılır. Sayısal veya tarihsel veriler için kullanılamaz.

Şekil 6.9.daki örnekte iki şart ifadesi "AND" (ve) operatörüyle birleştirildi. Böylece "birey" tablosundaki kayıtlar içinde soyadı "AKIN" olmayan ve adı annesinin adına eşit olan kayıtlar listlenmektedir. "Birey" tablosunda bu koşullara uyan bir kayıt bulunmaktadır.

Like Operatörü

Karşılaştırma operatörlerinden biri olan “Like” operatörü, diğer

operatörlerden farklı olarak sadece karakter türdeki değerler içinde arama yapmak için geliştirilmiştir. Özellikle aranan karakter ifadenin ne olduğu tam olarak bilinmediği durumlarda “=” ifadesi yerine “Like” operatörü kullanmak gerekmektedir.

Tablo 6.2. Joker Karakterleri

Joker Karakter	Anlamı
%	Birden fazla harf ya da rakamın yerini tutar.
_	Bir tek harf veya rakamın yerini tutar.
[Harf]	Herhangi bir harf yerine gelebilecek harfleri belirtir.
[^Harf]	Herhangi bir harf yerine gelemeyecek harfleri belirtir.
[A-Z]	A ile Z arasındaki harfleri belirtir.

Örnek olarak ismi “K” harfi ile başlayan bireyleri listeleyelim:

% karakteri birden fazla karakterin yerini tutar.

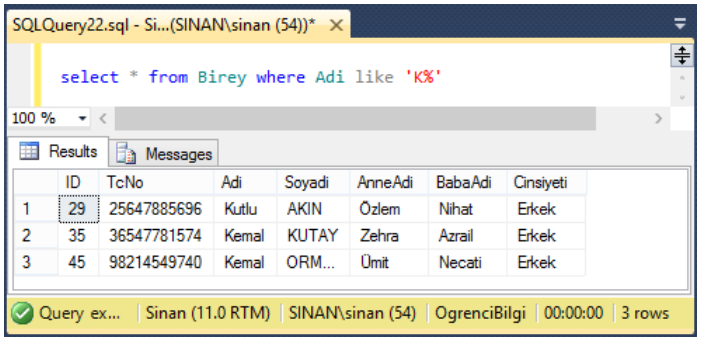
Örnek

```
•SELECT * FROM Birey WHERE Adi LIKE 'K%'
```

Örnek

```
•SELECT * FROM Birey WHERE Adi NOT LIKE '_e%'
```

“_” karakteri sadece bir karakterin yerini tutar.

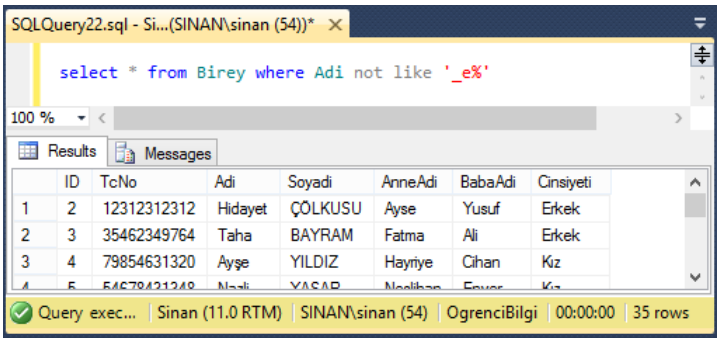


Şekil 6.10. İsmi “K” Harfi İle Başlayan Bireyler (3 Kayıt)

Şekil 6.10’da da görüldüğü gibi ismi “K” ile başlayanları ifade etmek için tek tırnaklar arasında “K” harfinden sonra “%” karakteri kullanılmıştır. Yüzde karakteri çok sayıda harfi veya rakamı temsil edebildiği gibi boşluğu da temsil edebilmektedir. Yani ilgili konumda herhangi bir karakter bulunmaya da bilir.

Verilen değer ile eşleşmeme durumunu kontrol etmek için de “Like” operatörüyle birlikte “Not” operatöründen de faydalanılmaktadır. Örnek olarak adının ikinci harfi “e” olmayan bireyleri listeleyelim:

[ ] köşeli parantezleri arasındaki karakter listesinden herhangi birinin gelebileceğini belirtir.



Şekil 6.11. İsminin ikinci harfi “e” olmayan bireyler (35 kayıt)

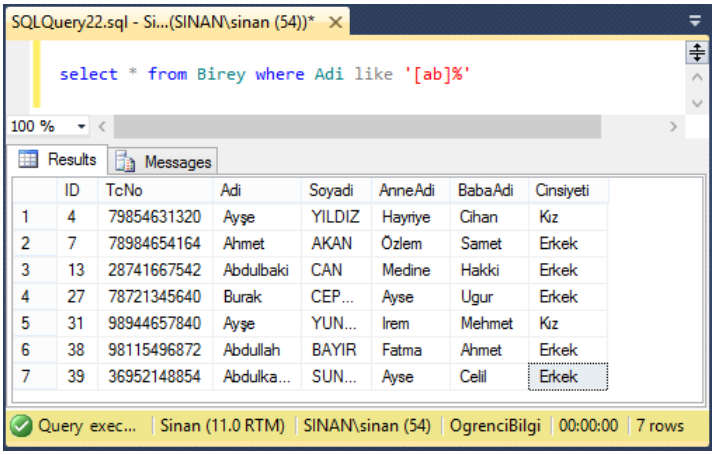
Şekil 6.11.deki örnekte de birinci harfi temsil için “_” (alttire) karakteri kullanıldı. Alttirenin, yüzde karakterinden ilk farkı, tek karakteri temsil etmesidir. İkinci farkı ise boşluk yerine kullanılmamasıdır. Yani alttire kullanılan konumda kesinlikle bir karakterin (harf, sayı vs.) bulunması beklenmektedir.

Şimdi de adının baş harfi “a” veya “b” olan bireyleri listeleyelim. Bunun için köşeli parantezler arasında “a” ve “b” harflerinin birlikte kullanıldığına dikkat edin.

Örnek

•SELECT * FROM Birey WHERE Adi LIKE '[ab]%'

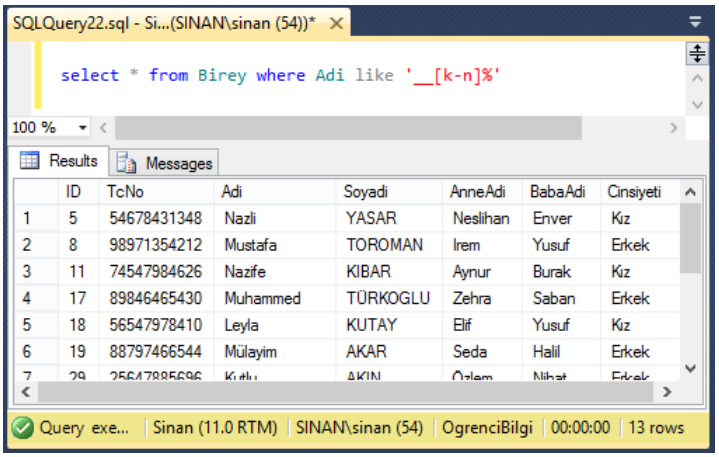
[ ] köşeli parantezleri arasında gelen iki karakter arasındaki karakterlerden herhangi birinin gelebileceğini belirtmek için – tire kullanılır.



Şekil 6.12. Adının Baş Harfi “A” Veya “B” Olan Bireyler (7 Kayıt)

Şimdi de adının üçüncü harfi “k”, “l”, “m” veya “n” olan bireyleri listeleyelim. Bunun için de köşeli parantezler içerisinde “k-n” kullanıldığına dikkat edin (şekil 6.13). Aralığın ilk harfi ile son harfi arasında “-” (tire) kullanıldı. Adının ilk iki harfini

temsil için iki adet "_" (alt tire) kullanıldı. Metin içindeki % karakteri ise ismin 3'ten fazla uzunlukta olabilmesi durumunu ifade etmektedir.

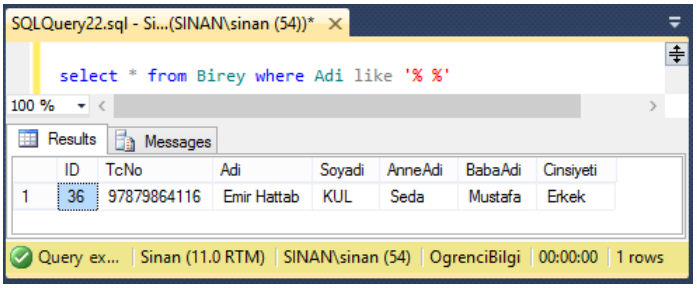


Şekil 6.13. Adının Üçüncü Harfi "K", "L", "M" Veya "N" Olan Bireyler (13 Kayıt)

Şimdi de çift isimli olan yani adı içinde boşluk karakteri bulunan bireyleri listeleyelim. Bunun için yüzde karakterleri arasında boşluk karakteri kullanıldı (Şekil 6.14).

Örnek

•SELECT * FROM Birey WHERE Adi LIKE'% %'



Şekil 6.14. Adı İçinde Boşluk Karakteri Bulunan Bireyler (1 Kayıt)

Between Operatörü

"Between" operatörü, bir aralık içinde yer alan kayıtları sorgulamak için kullanılır. Koşulun "true" (doğru) olabilmesi için kıyaslanan değerin, "Between" operatöründen sonra gelen birinci değerden büyük, ikinci değerden küçük veya bu değerlere eşit olması gerekmektedir.

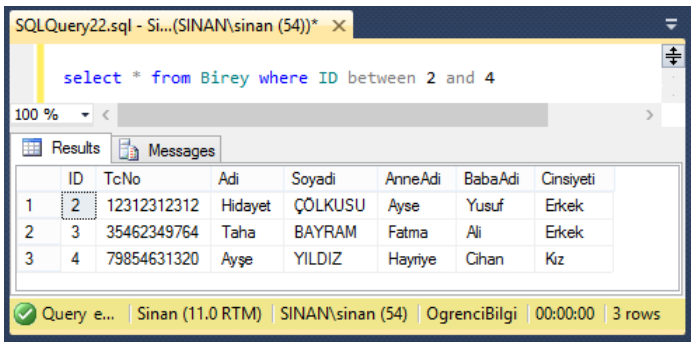
Between operatöründen sonra önce küçük değer kullanılmaktadır.

Söz Dizimi:

Sütun_Adı BETWEEN *Alt_Sınır* AND *Üst_Sınır*

Örnek

•SELECT * FROM Birey WHERE ID BETWEEN 2 AND 4



Şekil 6.15. ID Değeri 2 İle 4 Arasında Olan Bireyler (3 Kayıt)

Şekil 6.15.te "Between" operatörüne örnek verilirken "Birey" tablosundaki Id alanı 2 ve 4 arasında olan kayıtlar listelenmektedir.

"Between … and …" operatörünün eşleniği "And" operatörü ile yukarıdaki örnek sorgu aşağıdaki gibi de yazılabilir:

SELECT * FROM Birey WHERE ID >= 2 AND ID <= 4

In Operatörü

In operatörü, bir sütunun birden fazla değer ile aynı anda karşılaştırılabilmesi için kullanılmaktadır. Yani bir küme içinde arama yapılmaktadır. Karşılaştırmada kullanılan değerler parantez içinde virgüllerle ayrılmaktadır. "In", operatörü "Not" operatörü ile kullanıldığında ise sütun değerinin liste içindeki herhangi bir değere eşit olmama durumu sorgulanmaktadır (Elbahadır, 2012).

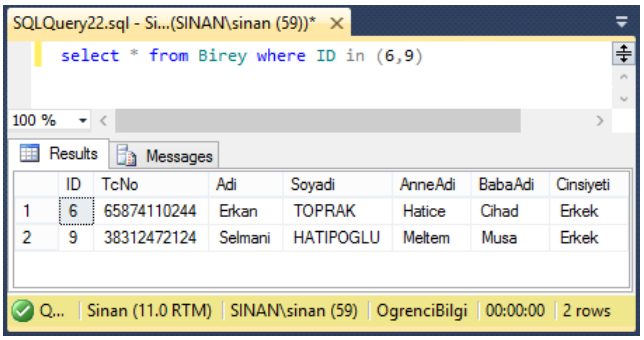
"In" operatörü liste içinde arama yapmak için kullanılır.

Söz Dizimi:

Sütun_Adı [NOT] IN (*Parametre_Listesi*)

Örnek

•SELECT * FROM Birey WHERE ID IN (6,9)



	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	6	65874110244	Erkan	TOPRAK	Hatice	Cihad	Erkek
2	9	38312472124	Selmani	HATIPOGLU	Meltem	Musa	Erkek

Şekil 6.16 ID Değeri 6 Olanların Ve ID Değeri 9 Olanların Listelenmesi (2 Kayıt)

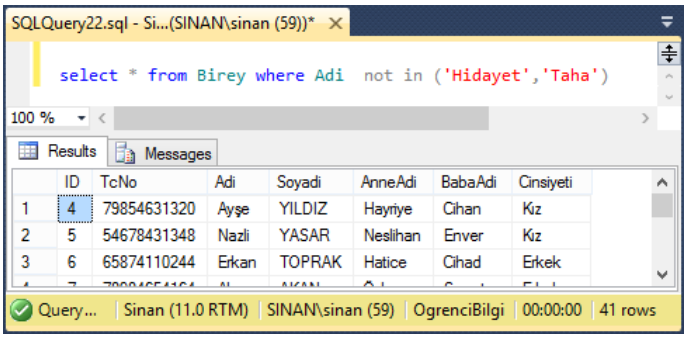
Şekil 6.16.daki örnekte "Birey" tablosunda Id'si 6 veya 9 olan kayıtlar listelenmektedir. Aynı sorgu "Or" operatörü marifetiyle aşağıdaki gibi de yazılabilir:

SELECT * FROM Birey WHERE ID =6 OR ID=9

Aşağıdaki örnekte ise "Not" operatörünün "In" operatörüyle kullanımı gösterilmektedir. "Adi" alanı "Hidayet" ve "Taha" olmayan bireyler listelenmektedir.

Örnek

- SELECT * FROM Birey WHERE Adi NOT IN ('Hidayet','Taha')



	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	4	79854631320	Ayşe	YILDIZ	Hayriye	Cihan	Kız
2	5	54678431348	Nazli	YASAR	Neslihan	Enver	Kız
3	6	65874110244	Erkan	TOPRAK	Hatice	Cihad	Erkek

Şekil6.17. Adı "Hidayet" Ve "Taha" Olmayan Kayıtların Listelenmesi (41 Kayıt)

Is Null Operatörü

Bu operatör, herhangi bir alana veri girilip girilmediğinin kontrolünü sağlamak maksadıyla kullanılmaktadır. Veri girilmemiş olan alanın değeri bildiğiniz gibi "Null"dur. Kısacası "is null" operatörüyle "Null" olan değer veya hücre tespit edilebilmektedir. "Null" olmayan hücrenin tespit edilebilmesi için de "Is Not Null" kullanılmaktadır.

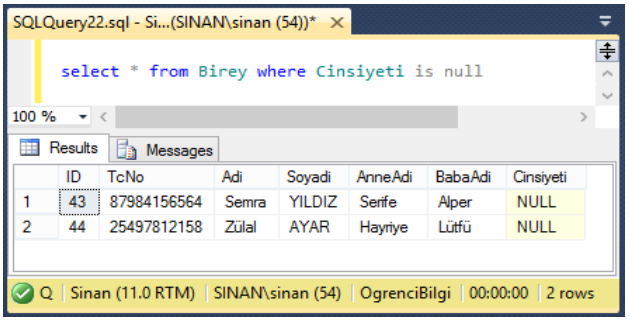
Söz Dizimi:

"Null" değerler için karşılaştırma operatörü kullanılamaz. Yani "adi=null" denemez.

Sütun_Adı IS [NOT] NULL

Örnek

•SELECT * FROM Birey WHERE Cinsiyeti IS NULL



Şekil 6.18. Cinsiyet Bilgisi Girilmeyen Bireyler (2 Kayıt)

Isnull() Fonksiyonu

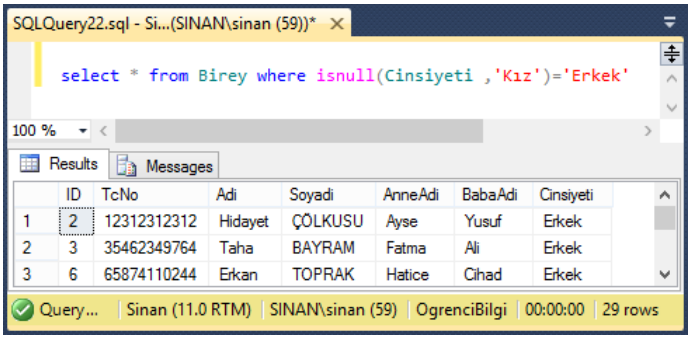
Isnull fonksiyonu, herhangi bir değerin null olması durumunda, sorgu sonucundan başka bir değer döndürmek için kullanılır. Bu fonksiyon, iki parametre almaktadır. Birinci parametre "null" olma durumu kontrol edilecek değeri, ikinci parametre ise "null" olma durumunda geri döndürülecek değeri temsil etmektedir.

Söz Dizimi:

ISNULL(*Sütun_Adı*, *Alternatif_Değer*)

Örnek

•SELECT * FROM Birey WHERE ISNULL(Cinsiyeti,'Kız')='Erkek'



Şekil 6.19. Erkek Bireylerin Listelenmesi (29 Kayıt)

Bu örnekte “Birey” tablosunda “Cinsiyeti” alanı “null” olan kayıtlar için değer, “Kız” olarak kabul ediliyor.

Coalesce() Fonksiyonu

Bu fonksiyon, “Isnull” fonksiyonuna benzemektedir. Farkı, bu fonksiyonun ikiden fazla parametre alabilmesidir. Fonksiyonun aldığı parametreler içinden, soldan sağa doğru “null” olmayan ilk parametre geriye döndürülmektedir. Bütün parametreler “null” değeri içeriyorsa da geriye “null” döndürülmektedir.

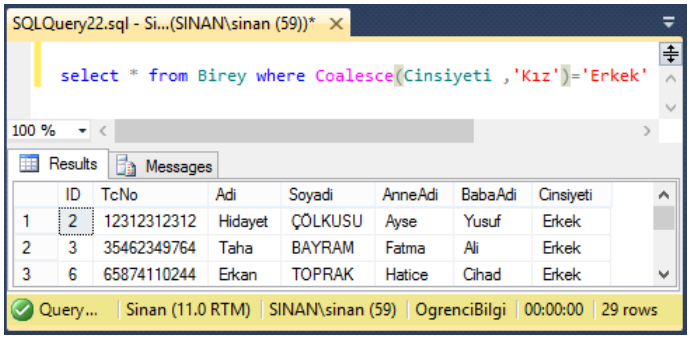
“Coalesce” fonksiyonu, “Isnull” fonksiyonunun geliştirilmişidir.

Söz Dizimi:

COALESCE(*Sütun_Adı*, *Alternatif_Değer1, Alternatif_Değer2 …*)

Örnek

- SELECT * FROM Birey WHERE COALESCE(Cinsiyeti,'Kız')='Erkek'



Şekil 6.20. Erkek Bireylerin Listelenmesi (29 Kayıt)

SIRALAMA İŞLEMLERİ

Veri tabanı tablolarından kayıtlar listelenirken tabloda kayıtlı olduğu sıraya göre görüntülendiğini belirtmiştik. Kayıtların bir veya birden fazla alana göre sıralı olarak listelenmesi için de “ORDER BY” deyimi kullanılabilmektedir. Kayıtların ilgili alana göre A’dan Z’ye veya küçükten büyüğe doğru sıralanmasında “ASC” (Ascending) deyimi kullanılırken, Z’den A’ya veya büyükten küçüğe sıralamada ise “DESC” (Descending) deyimi kullanılır. Sıralama tipi belirtilmediğinde ise varsayılan olarak “ASC” kabul edilmektedir.

Select ifadesinin yazımında en sonra sıralama işlemi gelmektedir.

Söz Dizimi:

SELECT *Select_Listesi* FROM *Tablo_Listesi* WHERE *Koşullar*

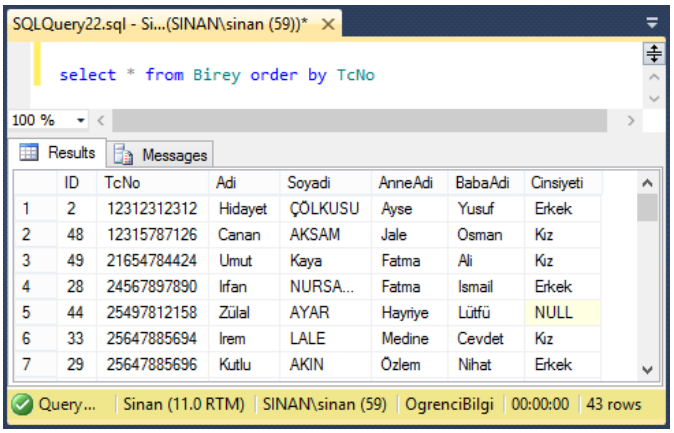
ORDER BY *sütun1* [DESC|ASC] [,*sütun2* [DESC|ASC], …]

Örnek olması için “Birey” tablosundaki kayıtları “TcNo” alanına göre sıralayarak listeleyen sorguyu yazalım. Sıralama tipi belirtilmediği için sıralamanın küçükten büyüğe doğru yapıldığını görebilirsiniz (Şekil 6.21).

Sıralama türü belirtilmezse “ASC” kabul eder ve küçükten büyüğe sıralar.

Örnek

•SELECT * FROM Birey ORER BY TcNo



	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	2	12312312312	Hidayet	ÇOLKUSU	Ayse	Yusuf	Erkek
2	48	12315787126	Canan	AKSAM	Jale	Osman	Kız
3	49	21654784424	Umut	Kaya	Fatma	Ali	Kız
4	28	24567897890	Irfan	NURSA...	Fatma	Ismail	Erkek
5	44	25497812158	Zülal	AYAR	Hayriye	Lütfü	NULL
6	33	25647885694	Irem	LALE	Medine	Cevdet	Kız
7	29	25647885696	Kutlu	AKIN	Özlem	Nihat	Erkek

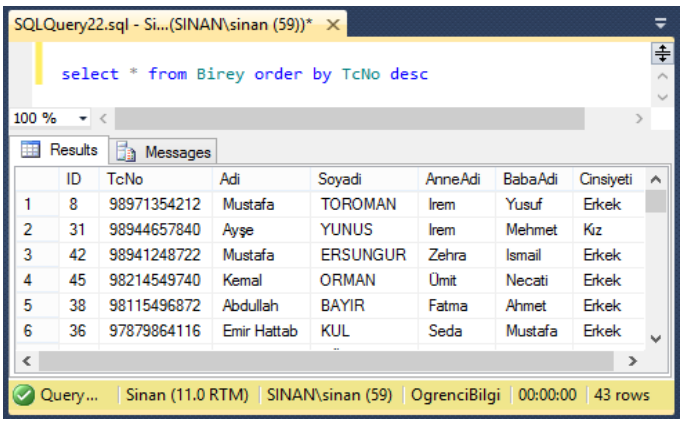
Şekil 6.21. Bireylerin TC Kimlik Numaralarına Göre Artan Şekilde Sıralanması (43kayıt)

Şimdi de “Birey” tablosundaki kayıtları “TcNo” alanına göre azalan şekilde sıralayalım.

“DESC” azalan sıralama, “ASC” artan sıralama için kullanılır.

Örnek

•SELECT * FROM Birey ORDER BY TcNo DESC



	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	8	98971354212	Mustafa	TOROMAN	Irem	Yusuf	Erkek
2	31	98944657840	Ayşe	YUNUS	Irem	Mehmet	Kız
3	42	98941248722	Mustafa	ERSUNGUR	Zehra	Ismail	Erkek
4	45	98214549740	Kemal	ORMAN	Ümit	Necati	Erkek
5	38	98115496872	Abdullah	BAYIR	Fatma	Ahmet	Erkek
6	36	97879864116	Emir Hattab	KUL	Seda	Mustafa	Erkek

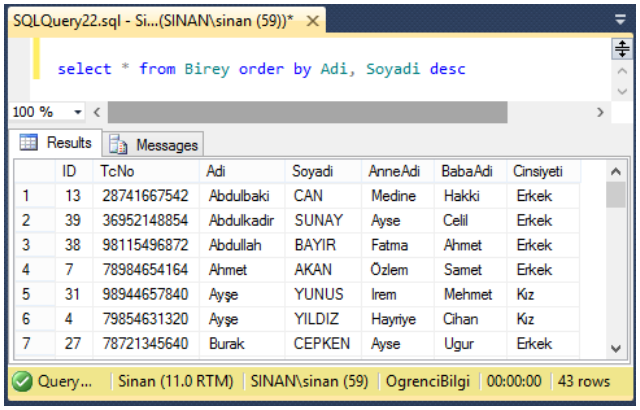
Şekil 6.22. Bireylerin TC Kimlik Numaralarına Göre Azalan Şekilde Sıralanması (43kayıt)

Bu defa da birden fazla alana göre sıralamaya örnek verelim. Bireyleri önce adlarına göre artan sonra soyadlarına göre azalan şekilde sıralayalım. "Order by" operatörü ile kayıtlar birden fazla sütuna göre sıralanırken sütun adlarının yazılış sırasına göre sıralanmaktadır. Yani örnekte (Şekil 6.23) "Soyadi, adı" şeklindeki kullanımda, öncelikle "Soyadi" alanına göre sıralamakta daha sonra soyadı aynı olanlar içinde "Adi" alanına göre kayıtları sıralayarak listelemektedir. Dikkat edin bu kod sadece listelemektedir; kayıtların tablodaki sırasını değiştirmemektedir.

"Order by" ve "Top" operatörleri birlikte kullanıldığında en yüksek veya en düşük değerlere sahip kayıtlar listelenebilir.

Örnek

•SELECT * FROM Birey ORDER BY Adi, Soyadi DESC



	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	13	28741667542	Abdulbaki	CAN	Medine	Hakki	Erkek
2	39	36952148854	Abdulkadir	SUNAY	Ayse	Celil	Erkek
3	38	98115496872	Abdullah	BAYIR	Fatma	Ahmet	Erkek
4	7	78984654164	Ahmet	AKAN	Özlem	Samet	Erkek
5	31	98944657840	Ayşe	YUNUS	Irem	Mehmet	Kız
6	4	79854631320	Ayşe	YILDIZ	Hayriye	Cihan	Kız
7	27	78721345640	Burak	CEPKEN	Ayse	Ugur	Erkek

Şekil 6.23. Bireylerin Adlarına Göre Artan, Soyadlarına Göre Azalan Şekilde Sıralanması

Top Operatörü:

"Order by" operatörü, "Top" operatörü ile birlikte kullanıldığında, sorgu sonucu dönen ilk N adet kaydın listelenmesi sağlanmaktadır.

Söz Dizimi:

SELECT TOP(*n*) [PERCENT] *select_listesi*

FROM *tablo_listesi*

WHERE *koşullar*

…

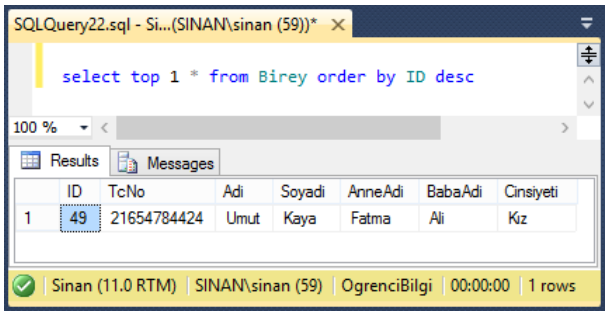
ORDER BY *sütun1* [DESC|ASC] [,*sütun2* [DESC|ASC], …]

Örnek

•SELECT TOP 10 PERCENT ID, Adi, Soyadi FROM Birey ORDER BY ID

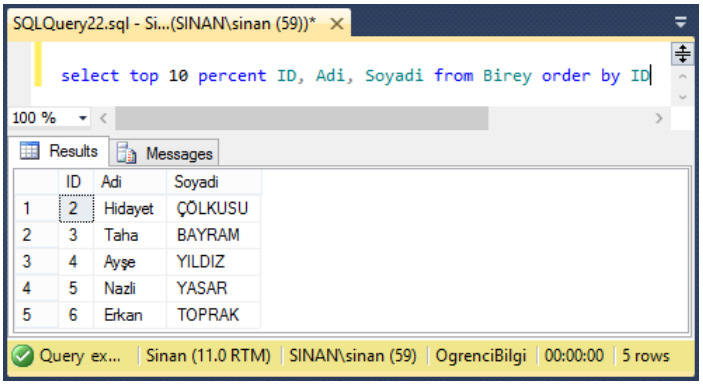
Örnek

•SELECT TOP 1 * FROM Birey ORDER BY id DESC



Şekil 6.24. ID Değeri En Büyük Olan Kaydın Gösterilmesi(1kayıt)

"Percent" (yüzde) deyiminin de bu iki deyime eklenmesiyle kayıtların yüzde kaçının listeleneceği belirtilebilmektedir.



Şekil 6.25. ID Değeri En Küçük Olanların %10'unun Listelenmesi (5 Kayıt)

Kayıtların %10'unu getirmek için "Top 10 Percent" ifadesi kullanılabilir.

Yukarıdaki örnekte "Birey" tablosundaki kayıtlar "ID" alanına göre sıralanarak kayıtların %10'u listelenmektedir.

Bireysel Etkinlik

- Bir tablodaki kayıtların sıralanmasında, tarihsel veri içeren kolona göre sıralamanın, tam sayı türünde kolona göre sıralamaya göre performans farkını test ediniz.
- Select deyiminin kullanımında bütün alanların getirilmesi ile yalnızca bir alanın getirilmesinin performansa etkisini test ediniz.
- Koşul tanımlarken "in" operatörü ile "=" operatörünün performans farkını test ediniz.
- "=" operatörünün "Like" operatöründen hızlı çalıştığını test ediniz.

Özet

- Herhangi bir tabloya SQL sorguları kullanılarak sütun eklemek için, “Alter Table” (tabloyu değiştir) deyimi ile birlikte “Add”(ekle) deyimi kullanılmaktadır. Tabloya birden fazla sütun birlikte eklenirken sütun tanımları virgüller ile ayrılmış olarak kullanılır. “Null” olup olmayacağı belirtilmeyen bir sütun ise varsayılan olarak “null” değer içerebilir olarak yorumlanmaktadır.
- Tablodaki bir sütunu güncellemek için "Alter Table" deyimi ile birlikte "Alter Column" deyimi kullanılmaktadır.
- "Select" deyimi, veri tabanında bulunan veri kaynaklarından verileri sorgulayarak seçmek için ve listelemek için kullanılmaktadır. Select deyimi ile tablodaki veriler üzerinde herhangi bir değişiklik yapılmamaktadır.
- "Select ... from..." cümleciğinde "Select" deyiminden sonra listelenmesi istenen sütun adları ve "From" deyiminden sonra ise tablo isimleri gelmektedir. Birden fazla sütun ve tablo adı birlikte kullanıldığında ise aralarına virgül konulmaktadır.
- Tablodaki bütün sütun adları listelenmek istendiğinde sütün adlarını tek tek yazmak yerine "*" karakteri kullanılmaktadır. Bu kullanım ile tablo isimlerinin yazılış sırasına ve sütun adlarının tablolardaki yer alış sırasına göre listelendiğini belirtmek gerekir.
- Select ifadesinden sonra yazılan sütun adları hangi sırada yazıldıysa o sırada ve sadece adı yazılı olan sütunlar listelenmektedir.
- SQL sorgu sonuçlarının sınırlandırılarak ilgili bazı kayıtların listelenmesi için özel bir koşul belirtilebilmektedir. Böylece sadece koşula uyan kayıtlar listelenecektir. Bu bağlamda, satırları filtreleyen koşulları belirtmek için “WHERE” operatöründen faydalanılmaktadır.
- İlgili kayıtların filtrelenebilmesi için kullanılan karşılaştırma operatörleri: eşit, eşit değil, büyük, büyük değil, küçük, küçük değil, büyük eşit, küçük eşit ve like operatörüdür. Birden fazla şartın birleştirilebilmesi için “And” ve “Or” mantıksal operatörlerinden faydalanılmaktadır. “AND” operatörü ile her iki şartın da sağlandığı kayıtlar listelenirken her iki şarttan birinin sağlanmasının yeterli olduğu durumlarda “veya” anlamına gelen “OR” operatörü kullanılmaktadır."Like" operatörü, karakter türdeki değerler için özelleştirilmiş, karşılaştırma operatörüdür.
- "Like" operatörü ile karşılaştırma yaparken "_" ve "%" joker karakterleri kullanılabilmektedir.
- "Between .. and.." karşılaştırma operatörü iki değer aralığını tanımlamak için kullanılır.
- "In" operatörü bir liste içindeki değerler ile topluca karşılaştırma yapmak için kullanılır.
- Sütun içeriğinin "Null" olup olmadığının kontrolü için “Is null” ve "Is not null" operatörleri kullanılmaktadır.
- "Isnull" ve "Coalesce" fonksiyonları bir sütun değerinin "null" olması durumunda yerine geçecek değeri tanımlamak için kullanılmaktadır.
- Sorgu sonucunda kayıtları sıralamak için "Order by" operatörü kullanılmaktadır.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi “Birey” tablosuna, tarihsel veri içeren ve “null” değer içermeyen “DogumTarihi” alanını eklemektedir?
   a) ALTER TABLE Birey ADD DogumTarihi DATETIME NOT NULL
   b) ALTER TABLE Birey ALTER COLUMN DogumTarihi NOT NULL
   c) ALTER TABLE Birey ADD DogumTarihi NULL
   d) ALTER TABLE Birey ALTER COLUMN DogumTarihi DATETIME NULL
   e) ALTER TABLE Birey ADD DogumTarihi DATETIME NULL

2. Aşağıdakilerden hangisi “Kitap” tablosundaki “YazarAdi” alanının uzunluğunu 50 karakter olarak güncellemektedir?
   a) ALTER TABLE Birey ALTER COLUMN TcNo VARCHAR(11)
   b) ALTER TABLE Kitap ALTER COLUMN YazarAdi VARCHAR(50)
   c) SELECT TOP 50 YazarAdi FROM Kitap
   d) ALTER TABLE Kitap ADD YazarAdi INT(50)
   e) ALTER TABLE YazarAdi ALTER COLUMN Kitap VARCHAR(11)

3. Aşağıdakilerden hangisi adının ikinci harfi “.” olan bireyleri listeler?
   a) SELECT * FROM Birey WHERE Adi LIKE ‘_.%’
   b) SELECT * FROM Birey WHERE Adi = ‘.2’
   c) SELECT * FROM Birey WHERE Adi LIKE ‘%.%’
   d) SELECT * FROM Birey WHERE LIKE Adi ‘.%’
   e) SELECT * FROM Birey WHERE Adi LIKE ‘%_.%’

4. Aşağıdakilerden hangisi adı içinde boşluk karakteri olan bireyleri listeler?
   a) SELECT * FROM Birey WHERE Adi LIKE ‘ ’
   b) SELECT * FROM Birey WHERE Adi = ‘% %’
   c) SELECT * FROM Birey WHERE Adi LIKE ‘% %’
   d) SELECT * FROM Birey WHERE LIKE Adi ‘ ’
   e) SELECT * FROM Birey WHERE Adi LIKE ‘_ _’

5. Aşağıdakilerden hangisi “SicilNo” bilgisi girilmemiş olan “Personel” kayıtlarını listeler?
   a) SELECT * FROM Personel WHERE SicilNo = ‘NULL’
   b) SELECT * FROM Personel WHERE ISNULL(SicilNo)
   c) SELECT * FROM Personel WHERE SicilNo LIKE NULL
   d) SELECT * FROM Personel WHERE SicilNo IS NULL
   e) SELECT * FROM Personel WHERE COALESCE (SicilNo, ‘’)

6. Aşağıdaki sorgulardan hangisi “Birey” tablosundaki kayıtları önce “Adi” sonra “Soyadi” alanına göre “a” dan “z” ye doğru sıralamamaktadır?
   a) SELECT * FROM Birey ORDER BY Adi, Soyadi
   b) SELECT * FROM Birey ORDER BY Adi ASC, Soyadi
   c) SELECT * FROM Birey ORDER BY Adi, Soyadi ASC
   d) SELECT * FROM Birey ORDER BY Adi ASC, Soyadi ASC
   e) SELECT * FROM Birey ORDER BY Adi DESC, Soyadi DESC

7. Aşağıdaki sorgulardan hangisi maaşı en yüksek olan 3 personeli listeler?
   a) SELECT * FROM Personel WHERE Maas = MAX(3)
   b) SELECT * FROM Personel GROUP BY MaaS, 3
   c) SELECT TOP 3 * FROM Personel ORDER BY Maas DESC
   d) SELECT TOP 3 PERCENT * FROM Personel ORDER BY Maas
   e) SELECT TOP 3 * FROM Personel ORDER BY Maas

8. Aşağıdaki sorgulardan hangisi “Birey” tablosundaki kayıtları listelerken sadece “Adi” ve “Soyadi” alanlarını görüntülemektedir?
   a) SELECT * FROM Birey WHERE Adi=Soyadi
   b) SELECT Adi, Soyadi FROM Personel
   c) SELECT * FROM Birey ORDER BY Adi, Soyadi
   d) SELECT Adi, Soyadi, * FROM Birey
   e) SELECT Adi, Soyadi FROM Birey

9. Aşağıdaki sorgulardan hangisi sayfa sayısı 400 ile 500 arasında olan kitapları yazar adlarına göre “z” den “a” ya doğru sıralamaktadır?
   a) SELECT * FROM Kitap ORDER BY YazarAdi DESC
   b) SELECT * FROM Kitap WHERE SayfaSayisi > 400 ORDER BY YazarAdi
   c) SELECT * FROM Kitap WHERE SayfaSayisi BETWEEN 400 AND 500
   d) SELECT * FROM Kitap WHERE SayfaSayisi BETWEEN 400 AND 500 ORDER BY YazarAdi DESC
   e) SELECT * FROM Kitap WHERE SayfaSayisi >400 AND SayfaSayisi<500

10. Aşağıdaki sorgulardan hangisi yanlış yazılmıştır?
    a) SELECT * FROM Birey ORDER BY Adi
    b) SELECT * FROM Birey ORDER BY Adi ASC, Soyadi DESC
    c) SELECT * FROM Birey ORDER BY Adi WHERE Adi=’Ali’
    d) SELECT * FROM Birey WHERE Adi= ‘Ali’ ORDER BY Adi
    e) SELECT * FROM Birey WHERE Adi LIKE ‘[az]%’

Cevap Anahtarı

1.a, 2.b, 3.a, 4.c, 5.d, 6.e, 7.c, 8.e, 9.d 10.c

YARARLANILAN KAYNAKLAR

Adar, İ., 2012. SQL Server 2012 Programlama ve Yönetim. İstanbul, 720

Gözüdeli, Y., 2010. Yazılımcılar için SQL SERVER 2008 R2 ve Veritabanı

Elbahadır, H., 2012. T-SQL SQL SERVER 2012. İstanbul, 294

Programlama. Ankara, 671

SORGU OLUŞTURMAK VE ÇEŞİTLERİNİ KULLANMAK-2

İÇİNDEKİLER

- Verileri gruplayarak analiz etme
- Grup fonksiyonları (max, min, count, avg, sum)
- Birden fazla sütuna göre gruplama
- Grup koşullarının kullanımı

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Gruplamanın ne olduğunu ve gruplamanın nasıl yapıldığını kavrayabilecek,
  - Gruplama fonksiyonlarının işlevlerini ve fonksiyonların kullanımlarını öğrenebilecek,
  - Birden fazla sütuna göre gruplama ve gruplanmış veriler ve grup fonksiyonları üzerinde koşul tanımlama izlemlerini gerçekleştirebilecek,
  - Null değerlerinin gruplamaya etkisi hakkında bilgi edeinebileceksiniz.

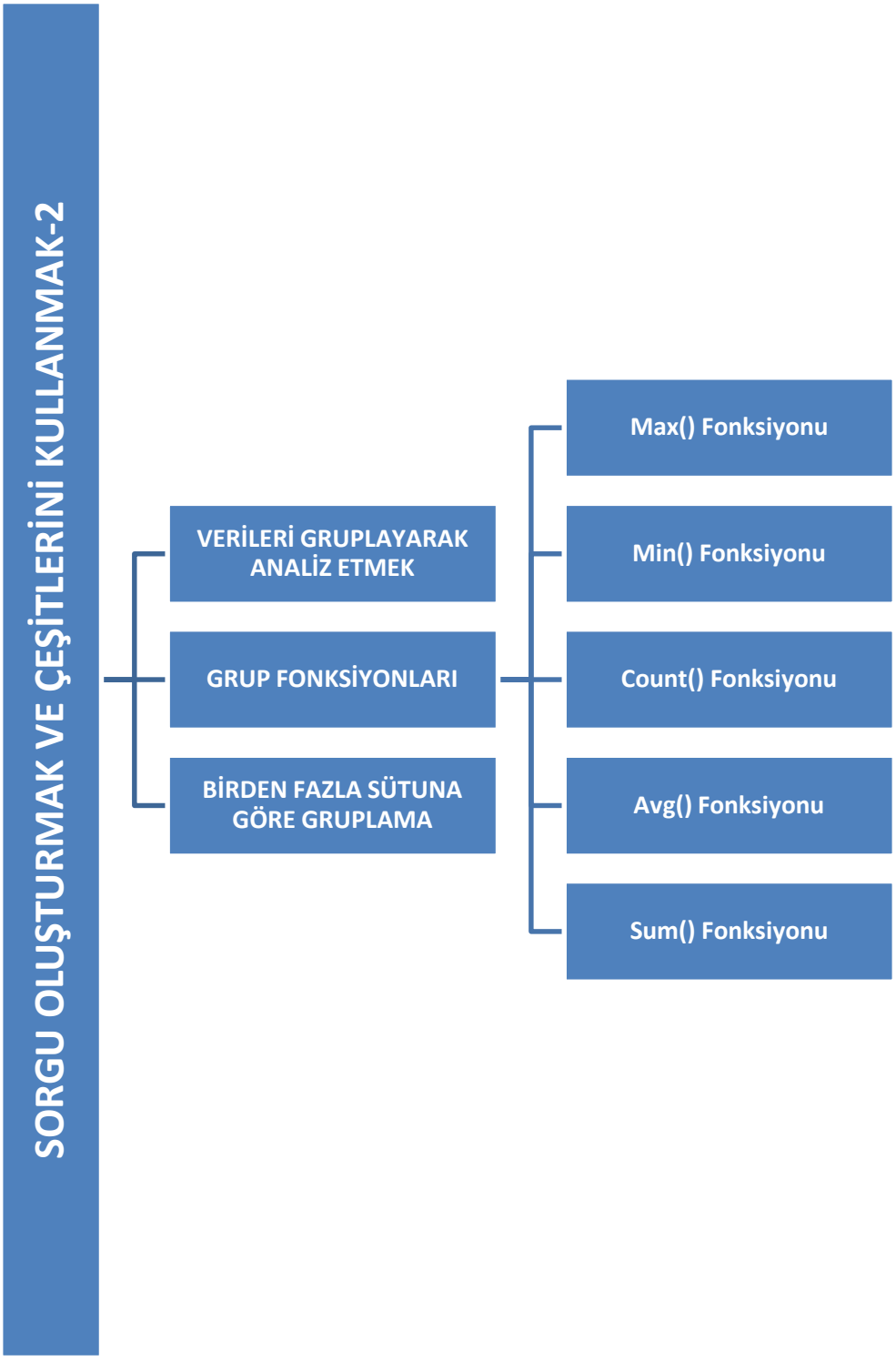
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi Sinan KUL

ÜNİTE

7


SORGU OLUŞTURMAK VE ÇEŞİTLERİNİ KULLANMAK-2
VERİLERİ GRUPLAYARAK ANALİZ ETMEK
GRUP FONKSİYONLARI
BİRDEN FAZLA SÜTUNA GÖRE GRUPLAMA
Max() Fonksiyonu
Min() Fonksiyonu
Count() Fonksiyonu
Avg() Fonksiyonu
Sum() Fonksiyonu

GİRİŞ

Önceki ünitede tablolara sütun eklenmesi ve var olan sütunlarının özelliklerinin güncellenmesi konularına değinildikten sonra SQL dilinde en çok kullanılan "Select" deyiminin kullanımına giriş yapılmıştı. "Select" deyiminin from deyimi ile kullanılarak veri tabanı tabloları veya görünümlerinden kayıtların nasıl çekildiği gösterilmişti. Yıldız operatörü ile kullanılarak ilgili tablodaki tüm sütunlar görüntülenebilirken sadece görüntülenmek istenen sütun verilerinin listelenebilmesi için de "Select" deyiminden sonra sadece ilgili sütunların yazılması gerektiği bilgisi verilmişti.

"Select" deyimi kullanılarak veri tabanı tablolarındaki tüm kayıtların veya özel şartlara uyan kayıtların ilgili sütunlarının nasıl listelenebildiği gösterilmişti. Kayıtların filtrelenmesinde kullanılan karşılaştırma operatörleri tanıtılarak her biri için açıklayıcı örnekler verilmişti. Önceki ünitede son olarak verilerin tablolardan sıralanarak listelenebilmesi için "order by" operatörünün nasıl kullanılabileceği ve sorgu sonucu dönen ilk N adet kaydın nasıl listelenebileceği gösterilmişti.

Bu ünitede ise verilerin istatistiksel analiz için nasıl gruplanacağı ve gruplanmış veriler üzerinde bazı hesaplama (gruplama) fonksiyonlarının nasıl işletileceği konuları işlenecektir. Gruplama fonksiyonlarının kullanım kuralları işlenirken her bir fonksiyonun parametre olarak alabileceği veri türüne değinilecektir. Bazı fonksiyonlar (Sum, Avg) sadece sayısal veriler için çalışabilirken diğerleri (Max, Min) metinsel ve tarihsel veriler için çalışabilmektedir. Count() fonksiyonu ise tüm veri türleriyle çalışabilmektedir. Count() fonksiyonunun diğer gruplama fonksiyonlarından farklı olduğu diğer bir konu ise parametre olarak "*" (yıldız) karakterini de alabilmesidir.

Gruplama fonksiyonlarının sütun adını parametre olarak kullandığı durumda "Distinct" ve "All" deyimlerinden biri sütun adından önce tercihen kullanılabilmektedir. Varsayılan olarak "All" deyimi kabul edilmektedir. Yani herhangi bir deyim kullanılmadığında tüm değerler hesaplamaya katılmaktadır. Distinct kullanıldığında ise aynı olan değerler sadece bir kere hesaplamaya katılmaktadır.

"Select" cümlesi içinde koşul belirtilirken, yani satırlar filtrelenirken, where operatöründen faydalanıldığı daha önceki ünitede öğretilmişti. Ancak koşul ifadesi içinde gruplama fonksiyonundan dönen değer kullanılacaksa, bu koşul ifadesinin having anahtar sözcüğünden sonra kullanılması gerekmektedir. Ünitede ayrıca, "select" cümlesi kurulurken yazım kurallarından bahsedilecek ve anahtar deyimlerin hangi sırayla cümle içinde yer alması gerektiği kurallarına değinilecektir.

Veriler gruplanmadan da gruplama fonksiyonlarının kullanılabilmektedir ki bu durum için ünitede çeşitli örnekler verilecektir. Ünitede, verilerin birden fazla alana göre nasıl gruplanacağı ve gruplu verilerin nasıl görüntülenebileceğine de değinildikten sonra son olarak gruplama fonksiyonlarından dönen değerler üzerinden nasıl koşul tanımlanacağı detaylı olarak incelenecektir.

VERİLERİ GRUPLAYARAK ANALİZ ETMEK

"Group By" deyimi "Where" operatöründen sonra ve "Order By" operatöründen önce gelmektedir.

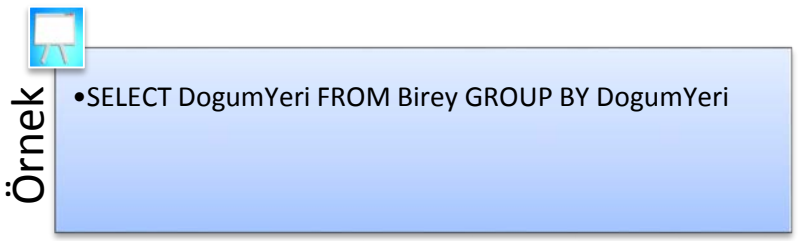
Veriler belirli özelliklerine göre istatistiksel olarak analiz edilebilmesi için kendi aralarında guruplanarak listelenebilmektedir. Gruplama ile genellikle grup toplamı, grup ortalaması, grup maksimumu, grup minimumu ve grup sayısı gibi değerlere ulaşılmak istenmektedir.

Verilerin gruplanmasında yani verilerin bir veya birden fazla tablodan gruplanarak sorgulanması için yazılan "Select" cümlesi içinde "Group By" deyimi kullanılmaktadır. "Group By" deyiminin "Select" yapısı içindeki tam konumu "Where" operatöründen sonra ve "Order By" operatöründen öncedir (Gözüdeli, 2010). Yani veriler gruplanmadan önce filtrelenecekse "Where" deyimi "Group by" deyiminden önce kullanılmalıdır. Aynı şekilde gruplanan veriler sıralanarak listelenmek istenirse de sıralama ifadesi olan "Order By", "Select" cümlesinin sonunda gelmelidir.

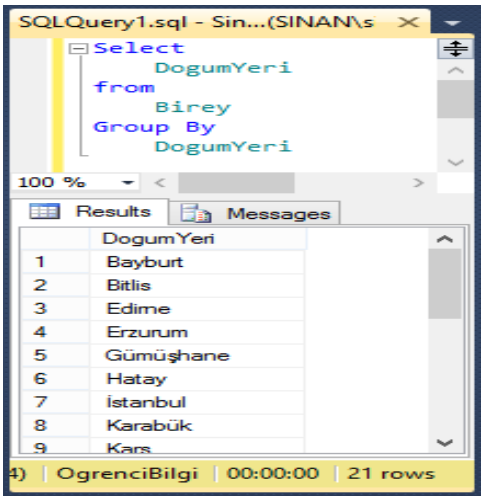
İçinde gruplamanın bulunduğu sorgu yapılarında sıklıkla yapılan hatalardan biri gruplama işlemine tabi tutulmayan alanların da görüntülenmek istenmesidir. Yani eğer gruplama operatörü kullanılmışsa sadece gruplanan alanlar ve gruplama fonksiyonlarından (Gruplama fonksiyonları bir sonraki konu başlığında işlenecektir.) dönen değerler görüntülenebilmektedir. Gruplanan alanlardaki veriler ise ilgili tablolardan benzersiz (eşsiz) olarak çekilmektedir.

Diğer bir dikkat edilecek konu, "Group By" ve "Order By" ifadelerinin birlikte kullanımları durumunda "Order By" ile birlikte kullanılan bütün sütun isimlerinin "Group By" ile gruplandırılmasının zorunluluğudur. Yani gruplama işlemi varsa görüntülenen ve sıralamaya sokulan bütün alanların gruplamaya alınması gerekmektedir.

Örnek olarak "Birey" tablosunda girilmiş olan "DogumYeri" verilerini benzersiz olarak listeleyelim (Şekil 7.1). "Group By" deyiminden sonra "DogumYeri" ifadesini yazarak sorgu sonucu dönen kayıtlar, "DogumYeri" alanına göre gruplanmaktadır.



Gruplanan veriler benzersiz olarak listelenmektedir.



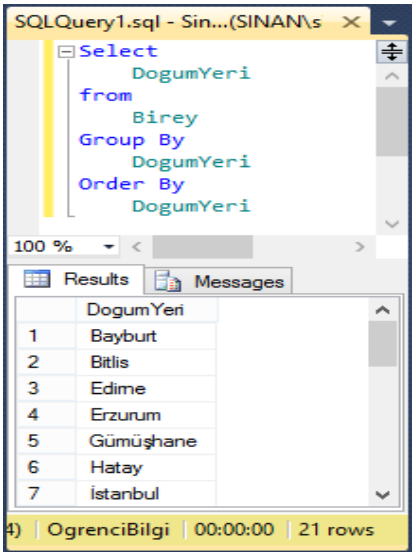
Şekil 7.1. Birey Tablosunun "Dogumyeri" Alanına Göre Gruplanarak Listelenmesi (21 Kayıt)

Gruplanan veriler ayrıca, sıralanmak istenirse sorgunun sonunda "Order By" deyimi yazılmalıdır. Şekil 7.2.deki örnek, sorgu sonucu dönen kayıtların "DogumYeri" alanına göre nasıl sıralanacağını göstermektedir.

Örnek

- SELECT DogumYeri FROM Birey GROUP BY DogumYeri ORDER BY DogumYeri

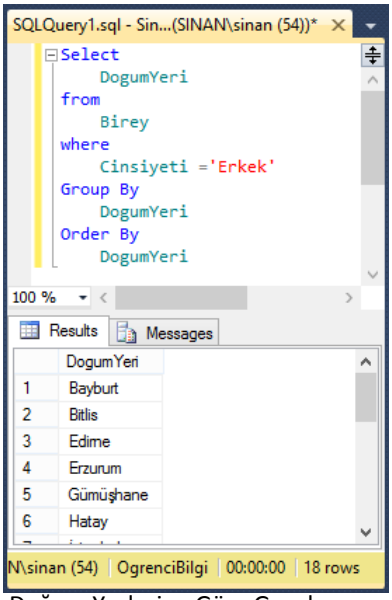
"Order By" deyimi sorgunun en altında yer almaktadır.



Şekil 7.2. "Dogumyeri" Alanına Göre Gruplandıktan Sonra Sıralanarak Listelenmesi (21 Kayıt)

Şimdi sorgumuzu biraz daha karmaşıklaştırarak kayıtları gruplamadan önce bir koşul ekleyelim ve filtrelenen veriler üzerinde gruplama yapalım. Sonrasında ise gruplu verileri sıralayarak listeleyelim.

Where" deyimi ile tanımlayacağımız koşulda "cinsiyeti" bilgisi "Erkek" olan bireyler çekildikten sonra verileri, "DogumYeri" alanına göre gruplayarak "DogumYeri" alanına göre alfabetik olarak (A'dan Z'ye) sıralayalım (Şekil 7.3).



Şekil 7.3. Erkek Bireylerin Doğum Yerlerine Göre Gruplanması Ve Sıralanması (18 Kayıt)

Bu örnek ile sorguya tanımlanan koşulun ("Where"), gruplama ifadesinden ("Group BY") önce geldiğini ve sıralama ifadesinin ("Order By") SQL sorgu cümlesinin sonunda geldiğini görmüş olduk.

GRUP FONKSİYONLARI

Gruplama fonksiyonları bir sütundaki verileri işleme tabi tutarak tek bir değer döndürür. Gruplama fonksiyonlarıyla mesela grup üyelerinin sayısı alınabileceği gibi bir birimde çalışan bireylerin maaş ortalamasını almak için de gruplama fonksiyonları kullanılabilmektedir. Bu bölüm altında detaylı olarak işlenecek olan gruplama fonksiyonlarından her bir grup için tek bir sonuç dönmektedir. Bu fonksiyonlar, "Max", "Min", "Avg", "Sum" ve "Count" olmak üzere beş tanedir.

Gruplama fonksiyonları, parametre olarak aldıkları sütundaki "null" olmayan değerler üzerinden işlem yapmaktadır. Diğer bir deyişle "null" değerleri görmezden gelmektedir (Adar, 2012). Örneğin "personel" tablosunda "maaş" alanının toplam bilgisi görüntülenmek istendiğinde, ilgili alanda "null" olmayan kayıtlar için işlem yapılmaktadır.

Veriler gruplanmadığında yani "Group By" ifadesi kullanılmadığında tüm veriler tek bir grup içinde kabul edilir. Dolayısıyla gruplanmayan veriler üzerinde de grup fonksiyonları kullanılabilir. Ancak gruplama fonksiyonları kullanılırken fonksiyon harici bir sütun değeri görüntülenmek istenirse bu sütuna göre verilerin gruplanması gerekmektedir. Aksi takdirde hata alınacaktır.

Grup fonksiyonlarının kullanım şekli aşağıdaki gibidir:

Söz Dizimi:

Fonksiyon([ALL | DISTINCT] ifade)

Tekrarlı kayıtları elemek için "DISTINCT" sözcüğünden faydalanılmaktadır. Böylece aynı veri fonksiyonda bir kere dikkate alınmaktadır.

"ALL" sözcüğü ise fonksiyonun bütün değerlere uygulanacağını söyler. "DISTINCT" ve "ALL" deyimleri kullanılmadığında SQL Server varsayılan olarak "ALL" seçeneğini kabul eder ve sorgudan dönen tüm değerler üzerinde işlem yapar. "DISTINCT" ve "ALL" deyimlerinin kullanımına ilişkin örnekler daha sonra verilecek, şimdilik burada keserek grup fonksiyonlarını tek tek açıklamaya geçelim.

"Sum" ve "Avg" fonksiyonları yalnızca sayısal değerler içeren sütunlar için kullanılır.

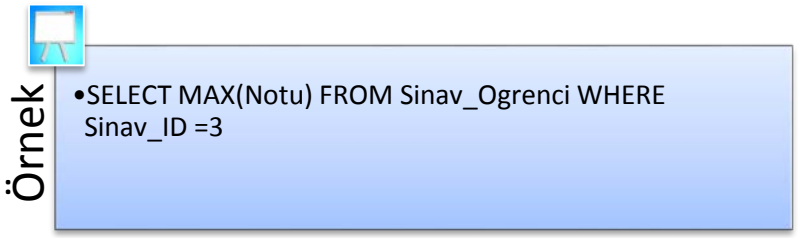
Tablo 7.1. Grup Fonksiyonları

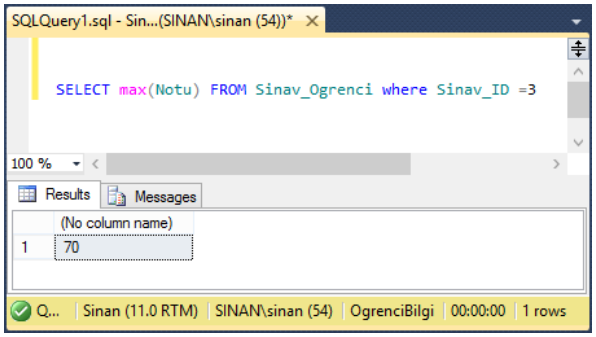
Fonksiyon	Desteklediği Veri Tipi	Açıklama
MAX	Sayısal, Metin, Tarih	Uygulandığı kolonun en büyük değerini verir.
MIN	Sayısal, Metin, Tarih	Uygulandığı kolonun en küçük değerini verir.
COUNT	Bütün veri tipleri	Koşula uyan kayıtların sayısını
AVG	Sayısal	Kolonun değerlerinin ortalamasını verir.
SUM	Sayısal	Kolonun değerlerinin toplamını verir.

Max() Fonksiyonu

Sorgu sonucu dönen kayıtlar içinde, "Max" fonksiyonu, parametre olarak aldığı sütun içindeki en büyük değeri bulur. Max fonksiyonu parametre olarak sayısal, metinsel ve tarihsel veri türlerindeki sütun adlarını veya değerleri almaktadır (Elbahadır, 2012).

Fonksiyonun daha iyi anlaşılabilmesi için bir örnek verelim. Örneğimizde (Şekil 7.4) öğrencilerin bir sınavdan (3 nolu sınav) aldıkları en yüksek notu görüntüleyelim. Hangi sınava ait kayıtlarla ilgilendiğimizi "Where" ifadesi ile belirtelim (Where Sınav_Id=3) . En yüksek sınav notu bilgisine erişmek ve görüntülemek içinse "MAX(Notu)" ifadesini "Select" deyiminden sonra yazalım.



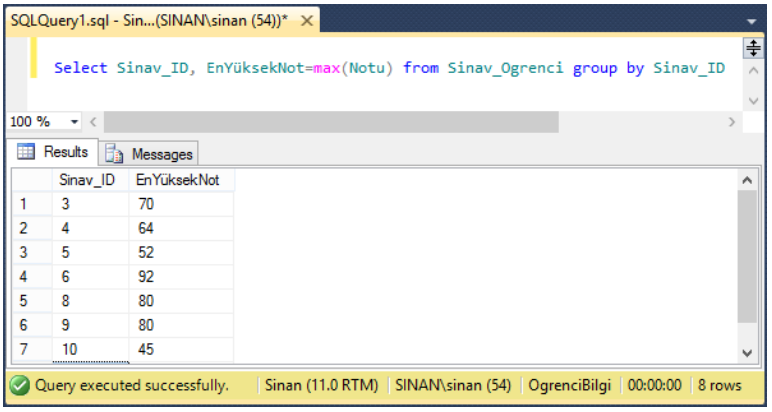


Şekil 7.4. 3 Nolu Sınavda Alınan En Yüksek Notun Görüntülenmesi

Sorgumuzda "Group By" deyimine yer vermedik, gruplama yapmadan da

gruplama fonksiyonlarının kullanılabileceğini göstermiş olalım istedik. “Group By” deyimi kullanılmadığı için veriler tek grup olarak kabul edilmektedir ve sorgu sonucu tek kayıt dönmektedir.

Şimdi de notları sınavlara göre gruplandırarak her bir sınavdan alınan en yüksek notları görüntüleyelim (Şekil 7.5). Gruplama fonksiyonu içinde geçmeyen her bir sütun adının “Group By” deyimi ile kullanılması gerektiğini daha önce söylemiştik. Örneğimizde “Group By Sinav_ID” ifadesi ile sorgu sonucu dönen kayıtların “Sinav_ID” alanına göre gruplanacağını belirtmektedir. “Sinav_ID” verisinin görüntülenmesi veya görüntülenme sırası tercihe bağlıdır. Yani “Select” deyiminden sonra hiç kullanılmayabilir veya “MAX(Notu)” ifadesinden önce veya sonra kullanılabilir.



Şekil 7.5. Sınavlarda Alınan En Yüksek Notların Görüntülenmesi (8 Kayıt)

Örnekte, “Max” fonksiyonuna “EnYüksekNot” takma adı verilmiş.

Örnek

••SELECT Sinav_ID, EnYüksekNot=MAX(Notu) FROM Sinav_Ogrenci GROUP BY Sinav_ID

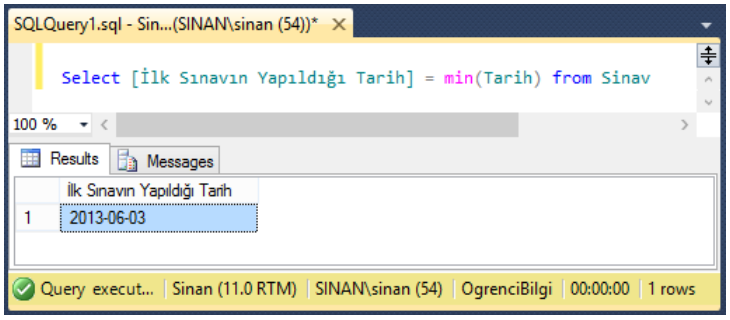
Min() Fonksiyonu

“Min” fonksiyonu, “Max” fonksiyonuyla aynı kullanım özellikleri göstermektedir. Farkı, parametre olarak aldığı sütun içindeki en küçük değeri bulmasıdır. “Max” fonksiyonu gibi “Min” fonksiyonu da sayısal, metinsel ve tarihsel veri türlerindeki sütun adlarını veya değerleri parametre olarak almaktadır. Yani sütundaki en küçük tarihi (en eski tarihi), sayıyı veya metni (alfabetik olarak sıralandığında en üstte kalan değeri) döndürmektedir.

Örnek olarak ilk yapılan sınavın tarihini bulalım. Bunun için “Sınav” tablosundaki “Tarih” alanından faydalanacağız. Fonksiyon sonucu dönen değere, “İlk Sınavın Yapıldığı Tarih” takma adını verelim. Takma ad içinde boşluk karakterleri bulunduğu için köşeli parantezler içinde kullanıldığına dikkat ediniz (Şekil 7.6).

Örnek

•SELECT [İlk Sınavın Yapıldığı Tarih] = MIN(Tarih) FROM Sinav



Şekil 7.6. En Önce Yapılan Sınavın Tarihinin Görüntülenmesi

"Min" fonksiyonunun, tarih veri türünde bir alan ile kullanımında en eski veriye ulaşıldığını görmüş olduk. Tarih veri türündeki alan, "Max" fonksiyonu ile kullanıldığında ise en güncel verinin yani en büyük tarih verisinin görüntüleneceğini söyleyebiliriz.

"Count" fonksiyonu "*" parametresiyle çalışabilen tek gruplama fonksiyonudur.

Count() Fonksiyonu

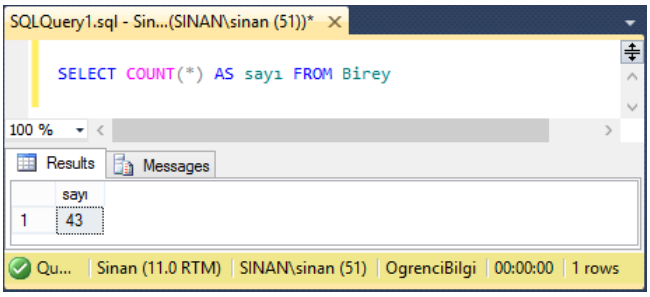
"Count" fonksiyonu en genel anlamıyla kayıt sayısını (tam sayı türünde) döndürmektedir. Parametre olarak "*" (yıldız) simgesi kullanıldığında sorgudan dönen kayıt sayısını verirken parametre olarak sütun adı kullanıldığında ise ilgili sütunda "null" olmayan kayıt sayısını döndürür. Herhangi bir sütunda farklı olan değer sayısını görüntülemek için de ilgili sütun adı "Distinct" deyimi ile birlikte kullanılabilmektedir.

"Count" fonksiyonu da diğer gruplama fonksiyonları gibi grupsuz veriler için yani group by deyimi olmayan bir select cümlesi ile listelenen kayıtlar için kullanılabilmektedir. "Count" fonksiyonunu diğer gruplama fonksiyonlarından ayıran özellik ise bu fonksiyonun, "*" parametresiyle çalışabilen tek gruplama fonksiyonu olmasıdır.

"Count" fonksiyonunun kullanımına örnek olması için "Birey" tablosundaki kayıt sayısını getiren sorguyu yazalım. Bu sorguda herhangi bir şart belirtmeyerek tablodaki kayıt sayısının tamamını alalım. Fonksiyon sonucu dönen değeri, "sayı" takma adı ile görüntüleyelim.

Örnek

•SELECT COUNT(*) AS sayı FROM Birey

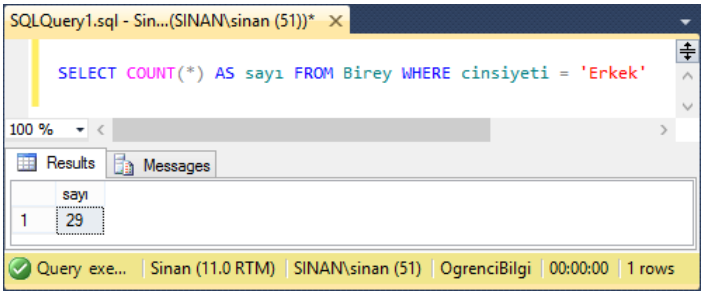


Şekil 7.7. Birey Tablosundaki Kayıt Sayısı

Şimdi de filtrelenmiş veriler üzerinde grup fonksiyonunu çağıralım ve Birey tablosunda "Cinsiyeti" bilgisi "Erkek" olan kayıtların sayısını veren sorguyu yazalım (Şekil 7.7). Bir önceki sorgudan (Şekil 7.6) tek farkı sorgu sonunda yer alan ve "Where" deyimi ile bir koşul tanımlanmasıdır.

Örnek

- SELECT COUNT(*) AS sayı FROM Birey WHERE cinsiyeti = 'Erkek'



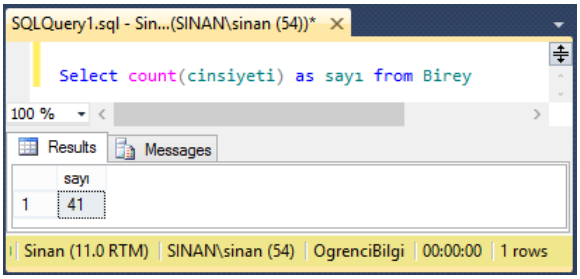
Şekil 7.8. Birey Tablosundaki "Cinsiyeti" "Erkek" Olan Kayıt Sayısı

"Group By" deyimi olmadan gruplama fonksiyonları kullanılırsa tek kayıt dönecektir.

Aşağıdaki SQL sorgusunda (Şekil 7.9) ise sütun isminin parametre olarak kullanılmasına örnek verilmiştir. Tanımlanan sorgu ile "Cinsiyeti" alanına değer girilmiş olan ("Null" olmayan) kayıt sayısı getirilmektedir.

Örnek

- SELECT COUNT(cinsiyeti) AS sayı FROM Birey

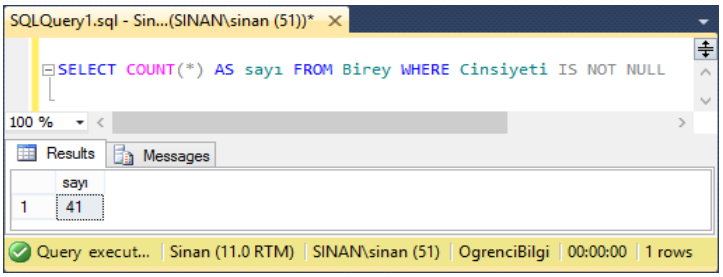


Şekil 7.9. Birey Tablosundaki "Cinsiyeti" Alanı Dolu Olan Kayıt Sayısı

"Cinsiyeti" bilgisi "Null" olmayan kayıt sayısına aşağıdaki sorgu (Şekil 7.10) ile de ulaşılabilir. "Where" deyiminden sonra "Cinsiyeti" alanının null olmaması koşulu tanımlanır ve sorgu sonucu dönen kayıt sayısı "Count" fonksiyonu ile alınır. "Count" fonksiyonunun * (yıldız) ile kullanıldığına dikkat ediniz.

Örnek

•SELECT COUNT(*) AS sayı FROM Birey WHERE Cinsiyeti IS NOT NULL



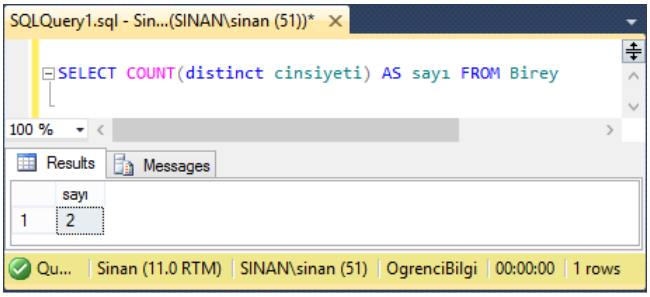
Şekil 7.10. Birey Tablosundaki "Cinsiyeti" Alanı Dolu Olan Kayıt Sayısı

Gruplama fonksiyonları yinelenen değerleri bir kere kullansın istenirse "Distinct" operatöründen faydalanılabilir.

Şimdi de "Distinct" deyiminin kullanımına da bir örnek olması için "Cinsiyeti" bilgisi aynı olan kayıtların bir kez sayılmasını sağlayan SQL kodunu yazalım (Şekil 7.11).

Örnek

•SELECT COUNT(DISTINCT cinsiyeti) AS sayı FROM Birey

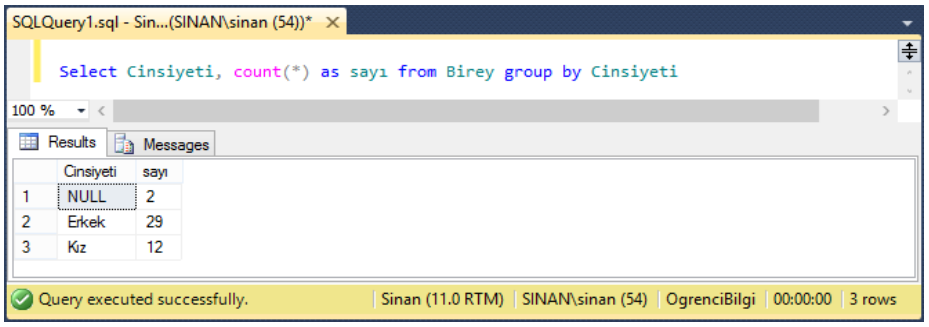


Şekil 7.11. Birey Tablosundaki "Cinsiyeti" Alanı Dolu Olan Kayıt Sayısı

"Count" fonksiyonunu bu kez de gruplanan veriler ile kullanalım. Yani "Birey" tablosunda bulunan kayıtlardan "Cinsiyeti" alanı "Erkek" olan "Kız" olan ve Cinsiyeti alanına değer girilmeyen (null olan) kayıt sayısını bulan sorguyu yazalım (Şekil 7.12). Bunun için gruplandırma deyiminden ("Group By") faydalanacağız. Bireyleri "Cinsiyeti" alanlarına göre grupladıktan sonra kayıt sayısını görüntüleyelim.

Örnek

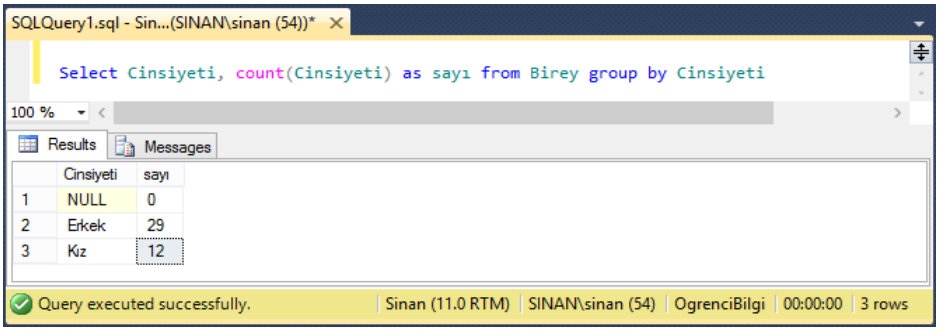
•SELECT Cinsiyeti, COUNT(*) AS sayı FROM Birey GROUP BY Cinsiyeti



Şekil 7.12. Cinsiyet Bilgilerinin Kaç Defa Görüntülendiğinin Listelenmesi. (3 Kayıt)

Gruplamada "Null" değerler de getirilmektedir.

Şimdi de yukarıdaki örnekte geçen "Count(*)" ifadesini "Count(Cinsiyeti)" olarak değiştirelim ve sorgu sonucundaki farklılığı görelim (Şekil 7.13).



Şekil 7.13. Cinsiyet Bilgilerinin Kaç Defa Geçtiğinin Listelenmesi.(Null Değerler Hariç)

Cinsiyeti bilgisi girilmemiş 2 adet birey kaydı bulunduğu hâlde örnekte de görüldüğü gibi "Count" fonksiyonu, parametre olarak sütun adı aldığı için Cinsiyeti null olanlar için kayıt sayısını 0 olarak görüntülemektedir.

"Count" fonksiyonu "null" olan sütun değerleri için 0 değerini döndürmektedir.

Avg() Fonksiyonu

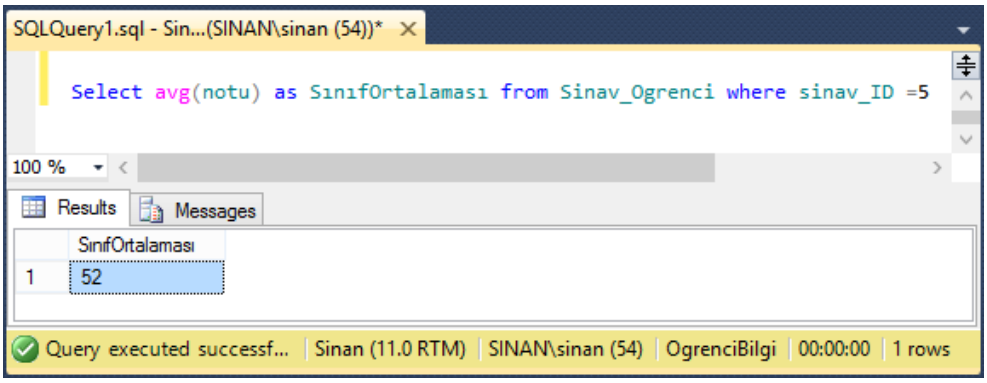
"Avg" fonksiyonu, parametre olarak aldığı değerlerin aritmetiksel ortalamasını almak için kullanılır. Bu fonksiyon sadece sayısal türdeki veriler ile çalışmaktadır. Diğer bir deyişle metinsel ve tarihsel veri türleri ile kullanıldığında hataya sebep olmaktadır.

Parametre olarak tam sayı veriler alırsa sonuç da tam sayı olur, parametre olarak noktalı sayı (float) veriler alırsa da sonuç noktalı sayı türünde bir sayı olacaktır. Metinsel ve tarihsel veri türündeki kolon adlarını ise parametre olarak almamaktadır.

"Avg" fonksiyonunun kullanımını anlamak için basit bir örnek verelim ve öğrencilerin belirli bir sınavdan (5 nolu sınavdan) aldıkları notların ortalamasını görüntüleyelim (Şekil 7.14).

Örnek

•SELECT AVG(notu) AS SınıfOrtalaması FROM Sinav_Ogrenci WHERE sinav_ID =5



Şekil 7.14. 5 Nolu Sınavdan Alınan Notların Ortalaması

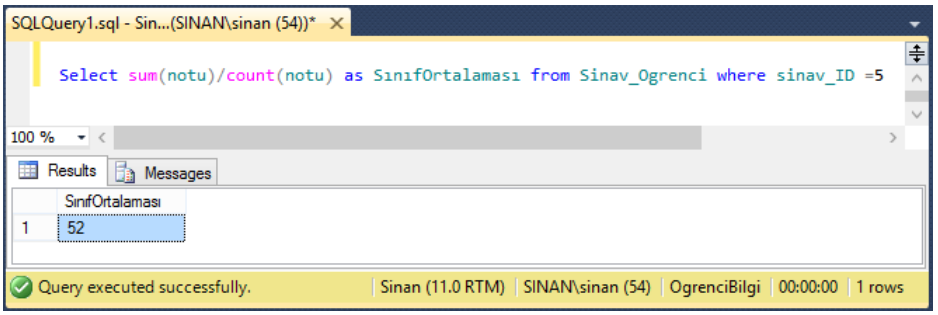
Sum() Fonksiyonu

"Sum" fonksiyonu, parametre olarak aldığı sütundaki değerleri toplayarak sonucu geri döner. Kullanımı "Avg" fonksiyonunun kullanımına benzerdir. Bu fonksiyon da sayısal türde veriler için geçerlidir. Yani parametre olarak aldığı sütunda tutulan verilerin sayısal türde olması gerekmektedir. Parametre olarak tam sayı veriler alırsa sonuç da tam sayı olur; parametre olarak noktalı sayı (float) veriler alırsa da sonuç da noktalı sayı türünde bir sayı olacaktır. Metinsel ve tarihsel veri türündeki kolon adlarını ise parametre olarak almamaktadır.

"Sum" fonksiyonunun kullanımına örnek olması için, öğrencilerin bir sınavdan aldıkları notların ortalamasını, "Sum" ve "Count" fonksiyonlarını kullanarak hesaplayalım (Şekil 7.15). Ortalamayı bulurken ilgili sınavdan alınan puanların toplamı, sınava giren öğrenci sayısına bölünmektedir. Sınav notu girilen öğrenci sayısını, "Count" fonksiyonu ile bulabileceğimizi daha önce görmüştük. "Sum" fonksiyonu ile hesaplanan toplam değeri, "Count" fonksiyonu ile hesaplanan kayıt sayısı değerine bölündüğünde ilgili sınavdan alınan notların ortalama değerine ulaşılmış olmaktadır.

Örnek

•SELECT SUM(notu)/COUNT(notu) AS SınıfOrtalaması FROM Sinav_Ogrenci WHERE sinav_ID =5



Şekil 7.15. 5 Nolu Sınavdan Alınan Not Ortalaması

"Count" fonksiyonu tam sayı ("int") türünde değer döndürdüğünü hatırlayalım. "Notu" alanı tam sayı ("int") türünde veri içerdiği için de "Sum" ve "Avg" fonksiyonları da sonuç olarak tam sayı türünde veri döndürmektedir. Dolayısıyla SQL'de iki tam sayının birbirine bölünmesinin de yine bir tam sayı olarak hesaplanacağı bilgisi akıldan çıkarılmamalıdır.

BİRDEN FAZLA SÜTUNA GÖRE GRUPLAMA

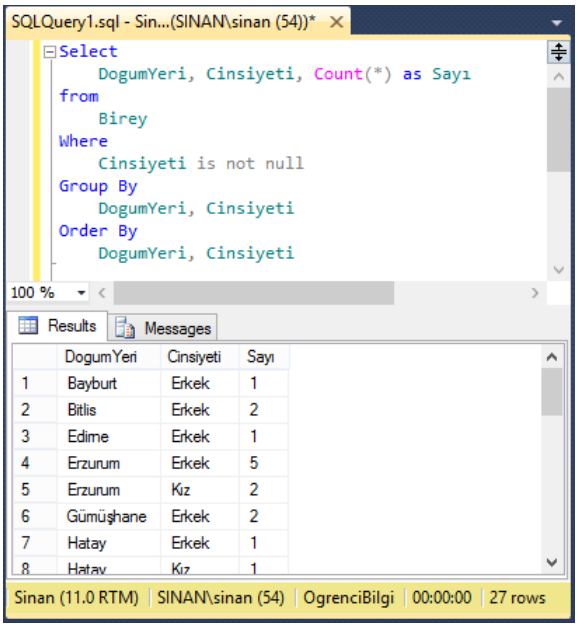
"Group By" deyimi ile gruplandırmaya katılmayan sütunlar "Select" deyimi ile kullanılamaz.

Birden fazla sütuna göre gruplarken "Group By" operatöründen sonra ilgili sütun adları yazılmaktadır.

Yeri gelmişken daha önce öğrendiğimiz bir kuralı hatırlatarak bir örnek verelim. Gruplama yapılırken "Select" ifadesi ile görüntülenen bütün sütun adları, "Group By" deyimi ile birlikte yazılmalıdır demiştik. Yani görüntülenen ve/veya sıralamada kullanılan sütun adları gruplamada kullanılmış olmalıdır. Aksi takdirde hata mesajı alınacaktır. Örnekte (Şekil 7.16) "Birey" tablosundaki kayıtlar "DogumYeri" ve "Cinsiyeti" alanlarına göre gruplandıktan sonra her bir doğum yerine göre aynı cinsiyette olan birey sayıları görüntülenmektedir. Cinsiyet bilgisi bulunmayanlar ise elenmektedir. Ayrıca gruplanan veriler "DogumYeri" ve "Cinsiyeti" alanlarına göre sıralanmaktadır.

Örnek

```
•SELECT DogumYeri, Cinsiyeti, COUNT(*) AS Sayı FROM
 Birey WHERE Cinsiyeti is not null
•GROUP BY DogumYeri, Cinsiyeti
•ORDER BY DogumYeri, Cinsiyeti
```



Şekil 7.16. Cinsiyet Bilgileri Girilmiş Olan Bireylerin Doğum Yerlerine Ve Cinsiyetlerine Göre Gruplanması Ve Sıralanarak Listelenmesi (27 Kayıt)

GRUP KOŞULLARININ KULLANIMI

Gruplama fonksiyonlarından dönen değerler için bir koşul tanımlamak gerektiğinde "Having" operatörü kullanılmalıdır.

Gruplama fonksiyonlarından dönen değerler üzerinden bir filtre tanımlanmak istendiğinde "Having" operatöründen faydalanılmaktadır. "Having" operatörünün "Select" cümlesindeki yeri, "Group By" operatöründen sonra ve "Order By" operatöründen ise öncedir. "Group By" operatörünün en genel kullanım şekli aşağıdaki gibidir:

Söz Dizimi:

SELECT Sütun_Adi1, Sütun_Adi2, … , GruplamaFonksiyonu(Sütun_Adi) FROM Tablo_Adi

WHERE Şartlar

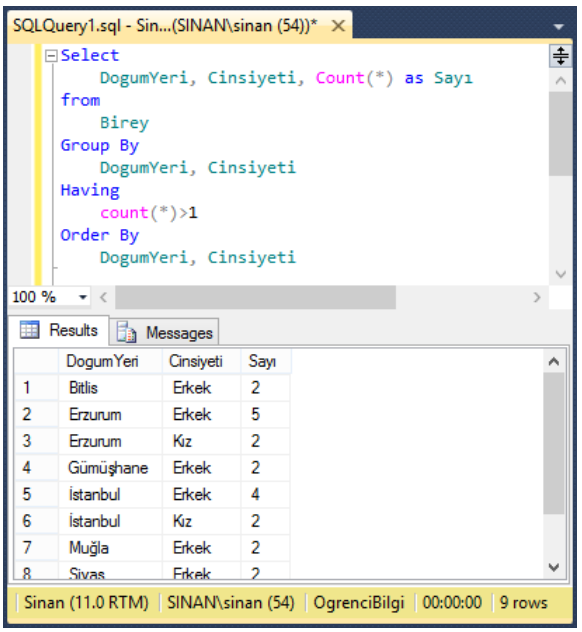
GROUP BY Sütun_Adi1, Sütun_Adi2, … HAVING Gruplama_Fonksiyonu_Şartları ORDER BY Sıralama_İfadesi

Operatörlerin sorguda yazılış sırasının değiştirilmesi sorgunun çalışmamasına sebep olacaktır. "Select … From …" ifadesi ile getirilen kayıtlar için,

"Where …" ifadesi ile şart tanımlanabildiğini görmüştük. "Group By … " ifadesi ile gruplamaya tabi tutulan kayıtlar için gruplama fonksiyonları ile ilgili koşullar da "Having …" ifadesi ile tanımlanmaktadır. Listelenecek kayıtlar ve kolonlar belirlendikten sonra istenirse kayıtlar üzerinde sıralama yapılabilmektedir. Dolayısıyla "Order By …" ifadesi sorgu yazımında en son gelmektedir.

Örnek

- SELECT DogumYeri, Cinsiyeti, COUNT(*) AS Sayı FROM Birey
- GROUP BY DogumYeri, Cinsiyeti
- HAVING COUNT(*)>1
- ORDER BY DogumYeri, Cinsiyeti



Şekil 7.17. Bireyler, Doğum Yerlerine Ve Cinsiyetlerine Göre Gruplandığında Gruptaki Kayıt Sayısı 1'den Büyük Olanların Sıralı Listelenmesi (9 Kayıt)

Şekil 7.17.deki örnekte bireyler, doğum yerlerine ve cinsiyetlerine göre gruplandıktan sonra gruptaki kayıt sayısı 1'den büyük olanlar listelenirken doğum yerlerine ve cinsiyetlerine göre artan şekilde sıralanmaktadır.

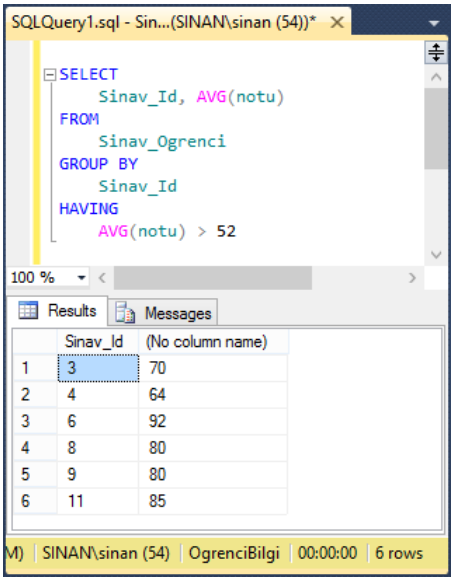
"Where" operatörüyle tanımlanan kısıtlar "Having" operatörü içinde de de yer alabilir.

"Having" operatörünün kullanımıyla ilgili dikkat edilecek hususlardan biri de "Where" operatörüyle tanımlanan kısıtlar "Having" operatörü içinde de yer alabilirken "Having" operatörü içinde tanımlanması gereken kısıtların "Where" operatörü ile kullanılmasının hataya sebep olmasıdır. Gruplama fonksiyonları sonucu dönen değerler ile ilgili kısıtlar sadece "Having" operatörü ile tanımlanabilmektedir. Diğer bir deyişle "Where" operatörüyle fiziksel veriler üzerinde filtre tanımlanabilirken "Having" operatörü, gruplama fonksiyonları ile hesaplanan sanal veriler üzerinde filtre tanımlanmasında kullanılmaktadır.

Şimdi de "Avg" fonksiyonundan dönen değerler ile filtre uygulamasına bir örnek verelim. Daha önce "Sınav_Id" bilgisi 5 olan sınavdan alınan notların ortalamasını alan SQL kod örneğini yazmıştık ve hesaplanan değerin 52 olduğunu görmüştük. Şimdi de sınav not ortalaması 52'den büyük olan sınavların ortalamalarını görüntüleyelim (Şekil 7.18).

Örnek

•SELECT Sinav_Id, AVG(notu) FROM Sinav_Ogrenci GROUP BY Sinav_Id HAVING AVG(notu) > 52



Fonksiyondan dönen değer için takma ad kullanılmazsa "No column name" olarak görünecektir.

Şekil 7.18. Sınav Not Ortalaması 52'den Büyük Olan Sınavların Not Ortalamaları

Bu örnekte gruplama fonksiyonuna takma ad verilmediği için SQL Server tarafından listeleme esnasında sütun adı, "No column name" (kolon adı yok) şeklinde görüntülenmektedir.

Bireysel Etkinlik

- Verilerin gruplanarak sorgulanmasının performans üzerindeki etkilerini araştırınız.
- Öğrencilerin sınavlardan aldıkları notları içeren bir tablo oluşturun ve her bir sınav için 50 ve üzeri not alan öğrenci sayılarını listeleyin.
- 3'ten fazla öğrencinin 50'nin altında not aldığı sınavları listeleyiniz.
- 3'ten fazla sınavdan 50'nin altında not alan öğrencileri listeleyiniz.

Özet

- Bir grup veri, belli bir özelliğe göre istatistiksel analiz için gruplandırılabilir. Bir veya birden fazla tablodan sorgulanan kayıtlar “Group By” deyimi kullanılarak gruplanabilmektedir.
- “Group By” deyimi “Select” yapısı içinde “Where” operatöründen sonra ve “Order By” operatöründen önce gelmektedir.
- Gruplama işlemi yapılırken görüntülenmek istenen alanlar, gruplama işlemine tabi tutulmalıdır. Yani “Group By” deyiminden sonra görüntülenmek istenen alanlar listelenmelidir.
- Gruplanan veriler üzerinde grup fonksiyonları kullanılabilmektdir.
- Grup fonksiyonu kullanılmadan gruplama yapıldığında kayıtlar benzersiz olarak listelenmektedir.
- Gruplama yapılırken select listesi içinde yer alan tüm sütun adları, gruplamaya dâhil edilmelidir.
- Gruplama fonksiyonları bir sütundaki verileri işleme tabi tutarak tek bir değer döndürmektedir.
- Başlıca grup fonksiyonları: MAX, MIN, AVG, COUNT ve SUM.
- Max fonksiyonu ilgili sütundaki en büyük değeri döndürür. Sayısal, metin ve tarih veri türlerindeki sütun adlarını parametre olarak almaktadır. Gruplama yapmadan da gruplama fonksiyonlarının kullanılabilmektedir ve sorgu sonucunda tek kayıt dönmektedir.
- Min fonksiyonu ilgili sütundaki en küçük değeri döndürür. Sayısal, metin ve tarih veri türlerindeki sütun adlarını parametre olarak almaktadır. Yani sütundaki en küçük tarihi, sayıyı veya metni döndürmektedir.
- Avg fonksiyonu ilgili sütun değerlerinin ortalamasını döndürür. Sayısal türde veriler için geçerlidir. Metin ve Tarihsel veri türleri için kullanımı hataya sebep olmaktadır.
- Sum fonksiyonu ilgili sütun değerlerinin toplamını döndürür. Yani parametre olarak aldığı sütunda tutulan verilerin sayısal türde veriler olması gerekmektedir. “Avg” fonksiyonu gibi bu fonksiyon da metin ve tarihsel veri türündeki kolon adlarını, parametre olarak almamaktadır.
- Count fonksiyonu herhangi bir sütunu parametre olarak aldığında, ilgili sütun değerlerinin "null" olmadığı kayıt sayısını döndürür.
- Count(*) şeklindeki kullanım ile sorgu sonucu dönen kayıt sayısını verir.
- Birden fazla sütuna göre gruplarken “Group By” operatöründen sonra bu sütun adlarının yazılması gerekmektedir. Diğer taraftan gruplama yapılırken “Select” ifadesi ile görüntülenen bütün sütun adları “Group By” deyimi ile birlikte yazılmalıdır.
- Grup fonksiyonlarının döndürdüğü değerler ile ilgili koşul tanımlamak için "Having" deyimi kullanılmaktadır.
- “Where” operatörüyle tanımlanan kısıtlar “Having” operatörü içinde de de yer alabilirken “Having” operatörü içinde tanımlanması gereken kısıtların “Where” operatörü ile kullanılması hataya sebep olmaktadır.
- "Having" deyimi "Group By" deyiminden sonra ve "Order By" deyiminden önce yazılmalıdır.

DEĞERLENDİRME SORULARI

1. Aşağıdaki sorgulardan hangisi personellerin maaşları toplamını getirir?
   a) SELECT COUNT(Maas) FROM Personel
   b) SELECT MAX(Maas) FROM Personel
   c) SELECT TOPLAM(Maas) FROM Personel
   d) SELECT SUM(Maas) FROM Personel
   e) SELECT SUM(*) FROM Personel

2. Aşağıdaki sorgulardan hangisi "Birey" tablosundaki "Adi" alanını tekrarsız olarak getirir?
   a) SELECT Adi FROM Birey GROUP BY Adi, Soyadi
   b) SELECT Adi FROM Birey GROUP BY Adi
   c) SELECT Adi FROM Birey GROUP BY Adi HAVING COUNT(*)>1
   d) SELECT Adi FROM Birey ORDER BY Adi
   e) SELECT DISTINCT Adi, * FROM Birey

3. Aşağıdaki sorgulardan hangisi "Birey" tablosundaki "Adi" ve "Soyadi" alanlarına göre tekrarlı olan yani adı ve soyadı aynı olan bireyleri listeler?
   a) SELECT Adi, Soyadi FROM Birey GROUP BY Adi, Soyadi
   b) SELECT * FROM Birey GROUP BY Adi, Soyadi HAVING COUNT(*)>1
   c) SELECT Adi, Soyadi FROM Birey GROUP BY Adi, Soyadi HAVING COUNT(*)>1
   d) SELECT TOP 2 Adi, Soyadi FROM Birey
   e) SELECT DISTINCT Adi, Soyadi FROM Birey

4. Aşağıdakilerden hangisi en yüksek maaşı alan personelin maaş bilgisini görüntüler?
   a) SELECT BirimId, MAX(Maas) FROM Personel
   b) SELECT MAX(Maas) FROM Personel
   c) SELECT AVG(Maas) FROM Personel
   d) SELECT Maas FROM Personel GROUP BY MAX(Maas)
   e) SELECT TOP1 Maas FROM Personel

```
SELECT DogumYeri, COUNT(*) FROM Birey GROUP BY
DogumYeri  HAVING COUNT(*)>11 ORDER BY DogumYeri
DESC
```

5. Yukarıdaki sorguya göre aşağıdakilerden hangisi yanlıştır?
   a) Bireyleri doğum yerlerine göre gruplar.
   b) Doğum yerlerine göre azalan Şekil 7.de sıralar.
   c) Doğum yeri anı olan birey sayısı 11'den büyük olan doğum yerlerini ve sayılarını görüntüler.
   d) Doğum yeri bilgisi girilmemiş olan bireyleri hariç tutmaktadır.
   e) Sorgu sonrası farklı girilmiş doğum yeri bilgisi sayısınca kayıt listeler

6. Aşağıdakilerden hangisi personelleri birimlerine göre gruplayarak her birimde en yüksek maaşı alan personelin maaş bilgisini görüntüler?
   a) SELECT BirimId, MAX(Maas) FROM Personel GROUP BY BirimId
   b) SELECT MAX(Maas) FROM Personel GROUP BY BirimId, Maas
   c) SELECT BirimId, AVG(Maas) FROM Personel ORDER BY BirimId
   d) SELECT BirimId, Maas FROM Personel GROUP BY MAX(Maas)
   e) SELECT BirimId, TOP(Maas) FROM Personel GROUP BY BirimId

7. Aşağıdakilerden hangisi cinsiyet bilgisi girilmeyen (null) birey sayısını verir?
   a) SELECT COUNT (Cinsiyeti) FROM Birey
   b) SELECT COUNT(*) FROM Birey
   c) SELECT SUM(*) FROM Birey HAVING Cinsiyeti IS NULL
   d) SELECT COUNT(Cinsiyeti) FROM Birey WHERE Cinsiyeti IS NULL
   e) SELECT COUNT(*) FROM Birey WHERE Cinsiyeti IS NULL

8. Aşağıdaki sorgulardan hangisi her durumda sıfır değerini döndürmektedir?
   a) SELECT COUNT(Cinsiyeti) FROM Birey WHERE Cinsiyeti IS NULL
   b) SELECT COUNT(*) FROM Birey WHERE Cinsiyeti IS NULL
   c) SELECT COUNT(*) FROM Birey
   d) SELECT SUM(Cinsiyeti) FROM Birey WHERE 1=2
   e) SELECT AVG(Cinsiyeti) FROM Birey

9. Aşağıdaki sorgulardan hangisi her durumda null değerini döndürmektedir?
   a) SELECT COUNT(Cinsiyeti) FROM Birey WHERE Cinsiyeti IS NULL
   b) SELECT COUNT(*) FROM Birey WHERE Cinsiyeti IS NULL
   c) SELECT COUNT(*) FROM Birey
   d) SELECT SUM(Cinsiyeti) FROM Birey WHERE 1=2
   e) SELECT AVG(Cinsiyeti) FROM Birey

"SELECT DogumYeri, Adi, SUM(*) FROM Birey GROUP BY DogumYeri WHERE AVG(Soyadi)>1 ORDER BY DogumYeri DESC"

10. Aşağıdakilerden hangisi verilen sorguda yapılan yanlışlardan biri değildir?
    a) "Order by" ifadesi cümlenin sonunda gelmiş.
    b) "*" karakteri "Sum" fonksiyonuna parametre olarak verilmiş.
    c) "Avg" fonksiyonu karakter türde bir değeri parametre olarak almış.
    d) Gruplama fonsiyonu "Where" ifadesi ile birlikte kullanılmış.
    e) "Adi" sütunu "Select" ifadesi içinde yer almış, "Group by" ifadesi içinde yer almamış.

Cevap Anahtarı

1.d, 2.b, 3.c, 4.b, 5d. 6.a, 7.e, 8.a, 9.d, 10.a

YARARLANILAN KAYNAKLAR

Adar, İ., 2012. SQL Server 2012 Programlama ve Yönetim. İstanbul, 720 Gözüdeli, Y., 2010. Yazılımcılar için SQL SERVER 2008 R2 ve Veritabanı

Elbahadır, H., 2012. T-SQL SQL SERVER 2012. İstanbul, 294
Programlama. Ankara, 671

SORGU OLUŞTURMAK VE ÇEŞİTLERİNİ KULLANMAK - 3

İÇİNDEKİLER

- Tabloların Birleştirilmesi
- Çoklu Tabloların Kullanılması
- Kartezyen Çarpımı
- Eşiti Olan Birleştirme
- Eşiti Olmayan Birleştirme

HEDEFLER

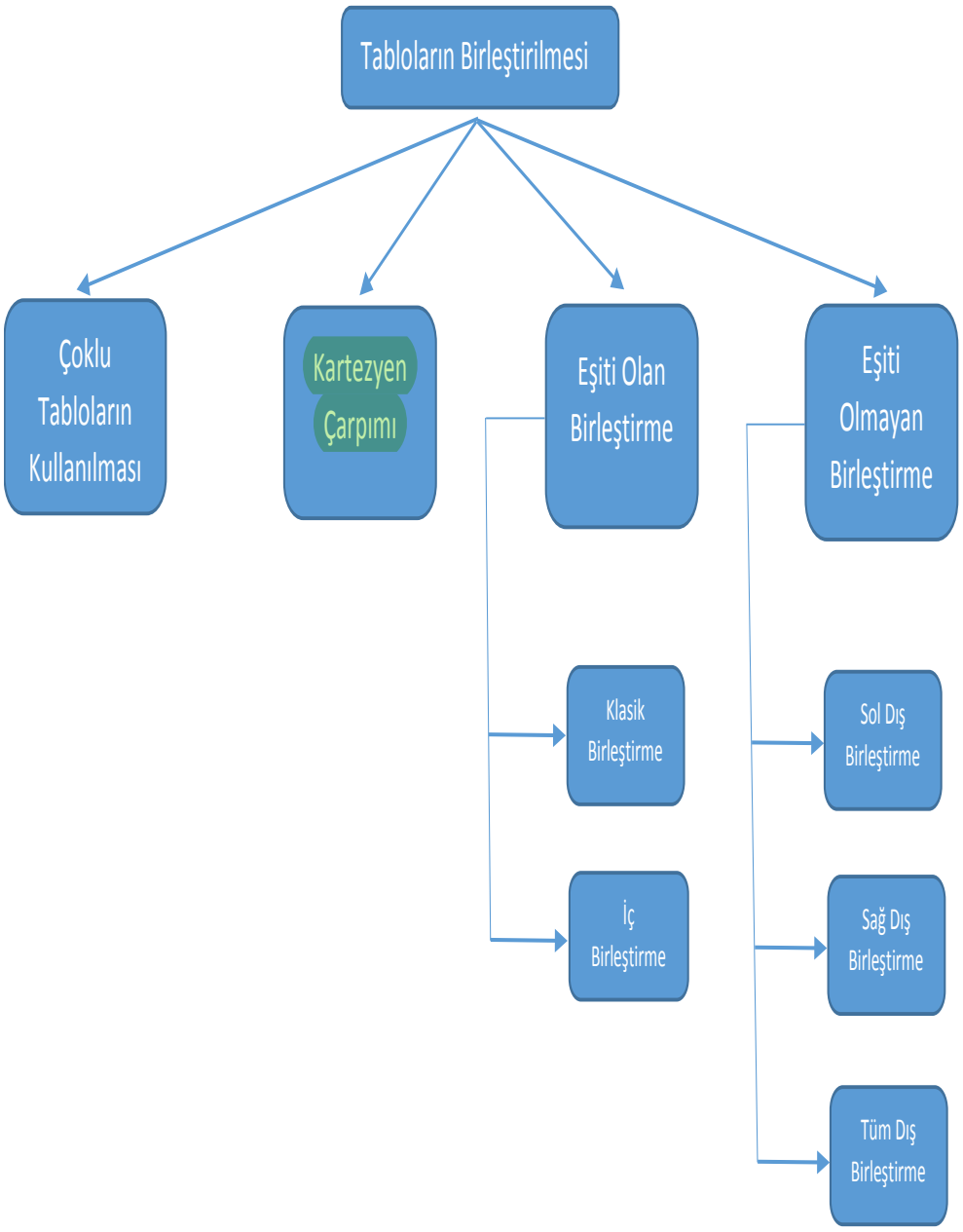
- Bu üniteyi çalıştıktan sonra;
  - Birden fazla tablodan veri sorgulanmasını kavrayabilecek,
  - Tabloya takma ad verilmesini ve tabloların koşul verilmeden birleştirilmesini öğrenebilecek,
  - Tabloların ortak sütunları üzerinden birleştirilmesini ve birleştirme koşulunu sağlayan kayıtların çekilmesini yapabilecek,
  - Birleştirme koşulunu sağlamayan kayıtların çekilmesi işlemi ve birleştirme koşulunu sağlamayan tabloya göre sorgu yapısı hakkında bilgi edinebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi
Sinan KUL

ÜNİTE
8


Tabloların Birleştirilmesi
Çoklu Tabloların Kullanılması
Kartezyen Çarpımı
Eşiti Olan Birleştirme
Eşiti Olmayan Birleştirme
Klasik Birleştirme
İç Birleştirme
Sol Dış Birleştirme
Sağ Dış Birleştirme
Tüm Dış Birleştirme

GİRİŞ

Önceki ünitede verilerin gruplanarak analiz edilmesi, birden fazla alana göre gruplama ve gruplanmış veriler üzerinde grup fonksiyonlarının işletilmesi ve grup fonksiyonlarından dönen değerler üzerinden koşul tanımlanması konularına değinilmişti. Gruplama fonksiyonlarının (Max, Min, Sum, Avg ve Count) kullanım kuralları işlenirken her bir fonksiyonun parametre olarak alabileceği veri türüne değinilmişti. Count() fonksiyonunun diğer gruplama fonksiyonlarından farklı olarak tüm veri türleriyle çalışabildiği ve parametre olarak "*" (yıldız) karakterini de alabildiği belirtilmişti. Grup fonksiyonlarından dönen değerler üzerinden belirli kayıtların filtrelenebilmesi için de gerekli koşulların tanımlanabilmesi için "Having" anahtar sözcüğünden faydalanılması gerektiği belirtilirken select cümlesinin söz dizim kuralları da ana hatlarıyla tanıtılmıştı.

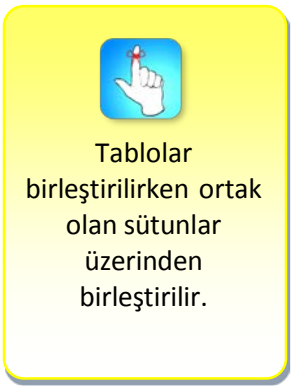
Özetle bundan önceki ünitelerde tek tablodaki kayıtların sorgulanması için gerekli T-SQL komutları işlenmişti. Bu ünitede ise T-SQL'de çoklu tabloların kullanılması konusu işlenecektir.

İlişkisel veri tabanı tasarımında, bilgiler farklı tablolara dağıtılır ve tablolar arası bağlantılar birincil ve ikincil anahtarlar üzerinden sağlanır. Yani bir tabloda sayı ile temsil edilen bir verinin gerçek değerine başka tablodan erişilebilmektedir. Bazen de bir varlığın farklı türde bilgileri farklı tablolarda tutulduğu için söz konusu varlığa ait diğer bilgilere erişmek için bir sorguda birden fazla tabloya erişmek gerekmektedir. Bu ünitede, farklı tablolarda farklı formatta tutulan kayıtların birleştirilerek tek liste oluşturulması gereğine binaen ve tabloların aralarında ilişkilendirilerek sorgulanmasında kullanılan SQL yapıları işlenecektir. Tabloların birleştirilmesinde yaygın olarak kullanılan yöntem tabloların, birincil ve ikincil anahtarlar vasıtasıyla eşleştirilmesidir. Tabloların ortak alanlarına yani birincil ve ikincil anahtarlarına göre birleştirilmesi, sorgu cümlesindeki koşullar ile sağlanmaktadır. Tablolar eşleştirildikten sonra her iki tabloda karşılıklı olarak eşleşen kayıtların listelenmesi (eşiti olan birleştirme), tablo birleştirme yapılarından "Inner Join" (iç birleştirme) ve klasik birleştirme ile mümkün olmaktadır. İç birleştirme, klasik birleştirmeyle aynı işleve sahip olmasına rağmen sorgunun yazılışı faklılık arz etmektedir. Klasik birleştirmede, koşul ifadesi için "Where" ifadesi yeterliyken iç birleştirmede "Inner Join" ve "On" ifadeleri kullanılarak sorgu oluşturulmaktadır.

Eşleştirilen tablolardan birinde bulunup diğer tabloda bulunmayan kayıtların da çekilebilmesi için eşiti olmayan birleştirme yapıları kullanılmaktadır. "Outer Join" (dış birleştirme) olarak da bilinen bu birleştirme "Left" (sol), "Right" ve "Full" (tüm) olmak üzere üç tip birleştirme bulunmaktadır. Bu birleştirme türleri dışında bir de herhangi bir koşul belirtilmeden iki tablonun birleştirildiği yapı bulunmaktadır ki bu birleştirmeye "Cross Join" (çapraz birleştirme) denilmektedir. Özetle bu ünitede her bir birleştirme türü ayrıntılı olarak işlenecek ve çeşitli örnekler ile konunun pekiştirilmesine çalışılacaktır.

TABLOLARIN BİRLEŞTİRİLMESİ

İlişkisel veri tabanı türünde, veri tekrarının önüne geçilebilmesi için bir varlığa iliklin veriler birden fazla tabloda tutulmaktadır. Örneğin öğrencinin kişisel bilgileri bir tabloda tutulurken iletişim bilgileri başka bir tabloda, aldığı dersler ve sınav notu bilgileri ise başka tablolarda tutulmaktadır. Herhangi bir öğrenciye ilişkin bilgiler sorgulanırken doğal olarak bu tabloların bazısı birlikte sorgulanmak durumundadır.



Diğer bir deyişle normalizasyon formları uygulanarak birden fazla tabloya dağılmış olan bilgi parçalarını tek bir listede birleştirerek gösterebilmek için tablolar arası birleştirme (join) işlemi uygulanır. Birden fazla tabloyu birleştirirken de tabloların ortak olan sütunları bir koşul ifadesiyle eşleştirilir. Daha sonra anlatılacak olan çapraz birleştirme ise bunun dışındadır.

Birleştirme şartındaki eşleştirme sonrası iki tablodan da sadece eşleşen kayıtların görüntülenmesinin sağlandığı, eşiti olan birleştirme, klasik birleştirme veya iç birleştirme ile yapılabilmektedir (Adar, 2012). Birleşme şartına uymayan kayıtların da listelendiği birleştirme türüne de eşiti olmayan birleştirme denilmektedir. Bütün bu birleştirme türleri ünite içinde ayrıntılı olarak incelenecektir. Aşağıdaki örnek kullanım, daha sonra anlatılacak olan klasik birleştirmeye aittir.

Söz Dizimi:

```
SELECT
  Select Listesi
FROM
  Tablo1, Tablo2
WHERE
  Tablo1.Sütun1=Tablo2.Sütun2
```

Takma Ad Verilmesi

Sorgulama esnasında veri tabanında yeralan bir tabloya veya kolona, takma ad verilmesi, sorgunun yazılmasını kolaylaştırır ve anlaşılır kılar. Tablo veya kolon adının uzun olması durumunda da kısa bir takma ad kullanılması yazımı kolaylaştırır. Hesaplanan değerler görüntülenirken de takma ad kullanılmadığında ilgili veriler "No column name" yani "Kolon adı yok" başlığı altında listelenmektedir (Gözüdeli, 2010). Anlaşılırlığı artırmak için de hesaplanan değerlerin bulunduğu sütuna takma ad verilmesi tavsiye edilmektedir.

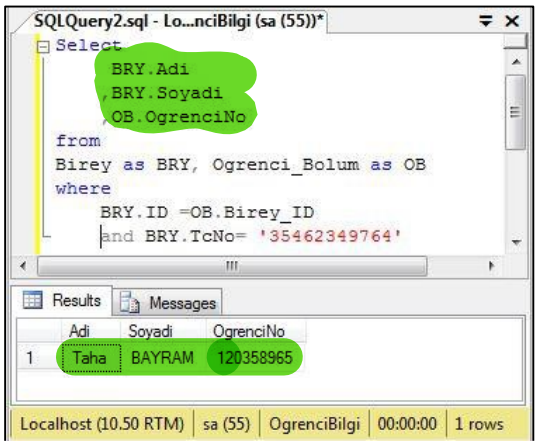
Birden fazla tablonun birleştirildiği durumda da olası karışıklıkların önüne geçmek için takma ad kullanımı faydalıdır. Özellike birleştirmede kullanılan tabloların aynı isme sahip sütunları bulunduğunda ilgili sütunlara erişilirken "TabloAdı.SütunAdı" yapısının kullanılması gerekmektedir. Bu yapı içinde de takma ad kullanılması yazımı kısaltacak ve anlaşılır kılacaktır. Ayrıca aynı isme fakat farklı

değerlere sahip sütunlar aynı sorgu sonucu görüntülenirken de takma ad kullanılması listeyi anlaşılır kılacaktır.

Tablolara takma ad verilmesine örnek olması için T.C. numarası bilinen bir öğrencinin, öğrenci numarasını bulan SQL kodunu yazalım (Şekil 8.1). Örnekte takma ad için tablo adlarının kısaltması kullanılmıştır. "Birey" tablosu için "Bry" ve "Ogrenci_Bolum" tablosu için de "OB" takma adı verilmiştir. Böylece "Birey" tablosu altındaki herhangi bir sütuna erişilirken bu takma ad kullanılmıştır. Örneğin "Adi" sütunu için "Bry.Adi" yazılmıştır. Bu yapının, özellikle iki tablonun da aynı isimde sütunlara sahip olması durumunda kullanılması gerektiğini söylemiştik. Dolayısıyla bu örnekte takma ad kullanımının zorunlu olmadığını tekrar ifade etmiş olalım.

Örnek

•SELECT Bry.Adi,Bry.Soyadi,Ob.OgrenciNo FROM Birey AS Bry, Ogrenci_Bolum AS Ob WHERE Bry.ID =Ob.Birey_ID AND Bry.TcNo= '35462349764'



Şekil 8.1. T.C.Nosuna Göre Öğrenci Numarası Getiren Sorgu

Takma ad kullanılırken nesnenin uzun adını temsilen birkaç harf kullanımı tercih edilebilir.

Birleştirilen tablolarda aynı isimde kolonlar bulunması durumunda bu kolonlardaki verilere erişilirken tablo adı uzun hâliyle veya takma adı ile kullanılmalıdır. Aksi takdirde hata verecektir. Örneğin, "Birey" ve "Ogrenci_Bolum" tablolarını birleştirdikten sonra "Birey" tablosundaki "ID" alanını görüntüleyelim

Örnek

•SELECT Bry.ID FROM Birey AS Bry, Ogrenci_Bolum AS Ob WHERE Bry.ID =Ob.Birey_ID

Takma ad uygulamasında "AS" ifadesinin kullanımı ise seçimliktir. Yani

kullanılması ve kullanılmaması arasında bir fark bulunmamaktadır.

ÇOKLU TABLOLARIN KULLANILMASI

Normalizasyon formları uygulanarak bir çok tabloya dağıtılmış olan verilere tek bir sorguda ihtiyaç duyulacak durumlar olacaktır. Bazı durumlarda aynı tabloyu bile kendisi ile ilişkilendirip birleştirmeye ihtiyaç duyulabilir. Bu durumda birden çok tabloyu bir araya getirip tablolar üzerindeki ilgili alanlarla tabloları ilişkilendirip birleştirmemiz gerekecektir.

Birleştirme yapısında tabloların yazılış sırası özellikle büyük tablolar için performansı etkileyebilmektedir. Ancak kullanımda ve görüntülenen kayıt sayısında herhangi bir farklılığa yol açmamaktadır.

Çoklu tabloların kullanımını basit bir örnek üzerinden izah etmeye çalışalım. Varsayalım ki veri tabanımızda öğrencilerin nüfus bilgilerinin tutulduğu "Birey" tablosu, adres ve telefon gibi iletişim bilgilerinin tutulduğu "Iletisim" tablosu olsun. Adresi Erzurum ilinde olan öğrencilerin TC Kimlik Numarası, ad soyad ve telefon bilgilerine ihtiyaç duyulduğunda "Birey" ve "Iletisim" tablolarını ortak alanlar üzerinden ilişkilendirerek sorgulamamız gerekecektir. Birden fazla tabloyu birleştirmek için kullanılan yöntemler ana hatlarıyla şöyle:

- Cross Join: Tabloların kartezyen çarpımını bulmak için kullanılır.
- Klasik Join : "Where" cümleciği kullanılarak yapılan birleştirmedir.
- Inner Join : Tablolar ilişkilendirip sorgulandığında sadece uyuşan kayıtlar geri döner.
- Outer join: Tabloların herhangi birinde yeralan kayıtları döndürür. "Left", "Right" ve "Full" olmak üzere 3 türü bulunmaktadır.

KARTEZYEN ÇARPIMI

Çapraz birleştirme (kartezyen çarpımı), üzerinde işlem yapılan iki tablodaki kayıtları çaprazlamak için kullanılır (Elbahadır, 2012). En nadir ihtiyaç duyulan bu birleştirme türünde ilişkili olsun veya olmasın birleştirilecek tabloların tüm satırları listelenmektedir. Birleştirme sonrası oluşan listede ilk satırda, birinci tablonun ilk kaydı ile ikinci tablonun ilk kaydı birlikte görüntülenirken listenin ikinci satırında birinci tablonun ikinci kaydı, ikinci tablonun ilk kaydı ile birlikte görüntülenmektedir. Birinci tablodaki tüm tüm kayıtların, ikinci tablonun ilk kaydı ile listelenmesinden sonra birinci tablodaki tüm kayıtlar tek tek bu defa da ikinci tablodaki ikinci kayıtlar ile görüntülenir. Ta ki ikinci tablodaki tüm kayıtlar da listelenene kadar çaprazlama bu şekilde devam eder.

Kartezyen çarpımda geri dönen kayıt sayısı, tablodaki kayıt sayılarının çarpımına eşittir.

Üzerinde birleştirme yapılan iki tablonun birincisinde x adet satır, diğerinde y adet satır varsa kartezyen çarpımı ile birlikte sonuç olarak x*y tane satırdan oluşan bir sonuç tablosu oluşmaktadır.

Çapraz birleştirmenin söz dizimi aşağıdaki gibidir. Çapraz birleştirmede "From" deyiminden sonra ilk tablonun adı ile ikinci tablonun adı arasına "Cross Join"

ifadesi yazılmaktadır. Klasik birleştirmede iki tablo arasına “,” konulduğunu ve tablo birleştirme koşulunun “Where” ifadesi sonrasında kullanıldığını hatırlatalım.

Söz Dizimi:

SELECT

Select_Listesi

FROM

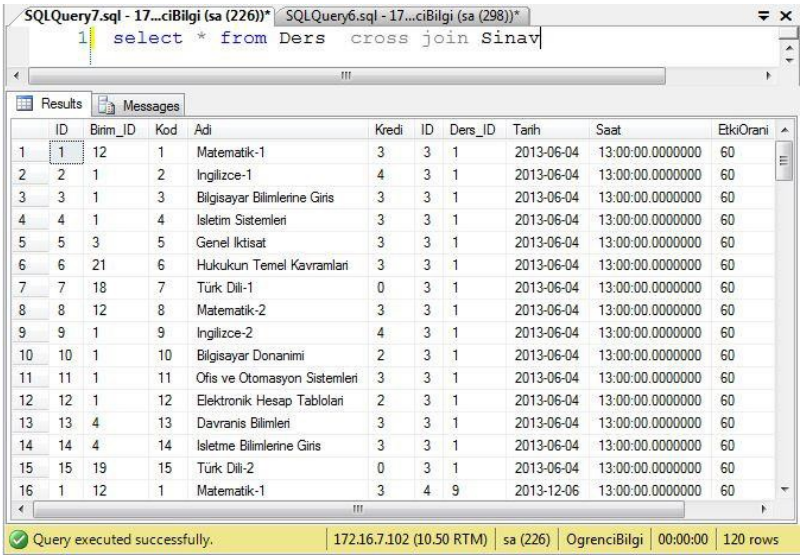
Tablo1 CROSS JOIN *Tablo2*

Örnek olarak “Ders” ve “Sinav” tablolarını “Cross Join” ile birleştirelim. Ders tablosunda 15 kayıt, sınav tablosunda ise 8 kayıt bulunmaktadır. Dolayısıyla çapraz birleştirme sonrası oluşan listede toplam 120 satır bulunması gerekmektedir (Şekil 8.2).

Çapraz birleştirmenin diğer adı kartezyen çarpımıdır.

Örnek

•SELECT * FROM Ders CROSS JOIN Sinav



	ID	Birim_ID	Kod	Adi	Kredi	ID	Ders_ID	Tarih	Saat	EtkiOrani
1	1	12	1	Matematik-1	3	3	1	2013-06-04	13:00:00.0000000	60
2	2	1	2	Ingilizce-1	4	3	1	2013-06-04	13:00:00.0000000	60
3	3	1	3	Bilgisayar Bilimlerine Giris	3	3	1	2013-06-04	13:00:00.0000000	60
4	4	1	4	Isletim Sistemleri	3	3	1	2013-06-04	13:00:00.0000000	60
5	5	3	5	Genel Iktisat	3	3	1	2013-06-04	13:00:00.0000000	60
6	6	21	6	Hukukun Temel Kavramlari	3	3	1	2013-06-04	13:00:00.0000000	60
7	7	18	7	Türk Dili-1	0	3	1	2013-06-04	13:00:00.0000000	60
8	8	12	8	Matematik-2	3	3	1	2013-06-04	13:00:00.0000000	60
9	9	1	9	Ingilizce-2	4	3	1	2013-06-04	13:00:00.0000000	60
10	10	1	10	Bilgisayar Donanimi	2	3	1	2013-06-04	13:00:00.0000000	60
11	11	1	11	Ofis ve Otomasyon Sistemleri	3	3	1	2013-06-04	13:00:00.0000000	60
12	12	1	12	Elektronik Hesap Tablolari	2	3	1	2013-06-04	13:00:00.0000000	60
13	13	4	13	Davranis Bilimleri	3	3	1	2013-06-04	13:00:00.0000000	60
14	14	4	14	Isletme Bilimlerine Giris	3	3	1	2013-06-04	13:00:00.0000000	60
15	15	19	15	Türk Dili-2	0	3	1	2013-06-04	13:00:00.0000000	60
16	1	12	1	Matematik-1	3	4	9	2013-12-06	13:00:00.0000000	60

Şekil 8.2. “Ders” Ve “Sinav” Tablolarını “Cross Join” İle Birleştirilme

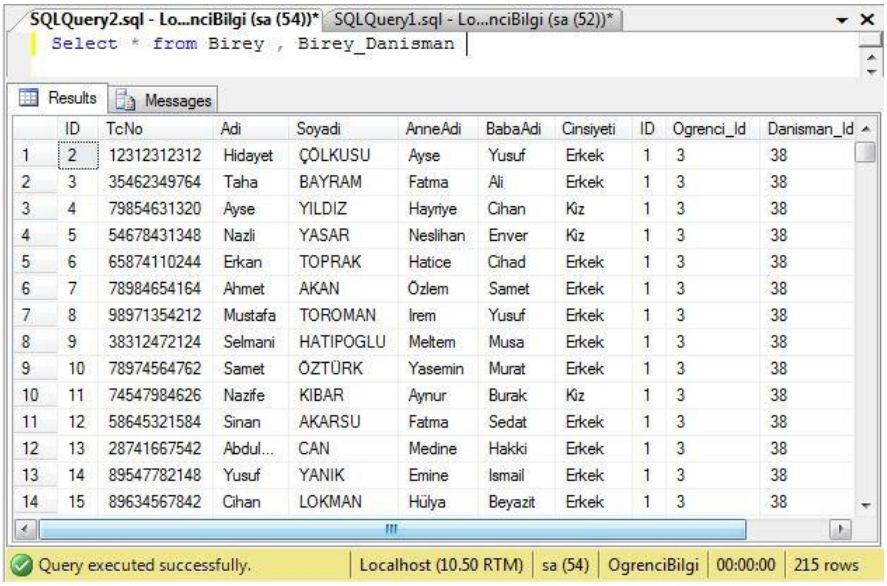
Şekil 8.2.deki örnekte öncelikle sınav tablosunun ilk kaydı, ders tablosundaki 15 kayıt ile birlikte görüntülenmektedir. Tablo alanlarının görüntülenme sırası ise ilgili tabloların kullanım sırasına göredir.

İki tablo arasında birleştirme koşulu tanımlanmamışsa da çapraz birleştirme elde edilmiş olur. Örnek olarak “Birey” ve “Birey_Danisman” tablolarını kullanarak bir kartezyen çarpımı elde edelim (Şekil 8.3). Bu kullanımda da “Cross Join” yapısına benzer şekilde öncelikle ikinci tablodaki (Birey_Danisman) tüm alanlar birinci

tablodaki (Birey) ilk kayıtla birlikte görüntülenmektedir. Görüntüleme sırasında ise öncelik Birey tablosunun alanlarındadır. Birleşme sonrası oluşan satır sayısı ise yine iki tablodaki kayıt sayılarının çarpımına eşittir.

Örnek

•SELECT * FROM Birey, Birey_Danisman



	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti	ID	Ogrenci_Id	Danisman_Id
1	2	12312312312	Hidayet	ÇÖLKUSU	Ayse	Yusuf	Erkek	1	3	38
2	3	35462349764	Taha	BAYRAM	Fatma	Ali	Erkek	1	3	38
3	4	79854631320	Ayse	YILDIZ	Hayriye	Cihan	Kiz	1	3	38
4	5	54678431348	Nazli	YASAR	Neslihan	Enver	Kiz	1	3	38
5	6	65874110244	Erkan	TOPRAK	Hatice	Cihad	Erkek	1	3	38
6	7	78984654164	Ahmet	AKAN	Özlem	Samet	Erkek	1	3	38
7	8	98971354212	Mustafa	TOROMAN	Irem	Yusuf	Erkek	1	3	38
8	9	38312472124	Selmani	HATIPOGLU	Meltem	Musa	Erkek	1	3	38
9	10	78974564762	Samet	ÖZTÜRK	Yasemin	Murat	Erkek	1	3	38
10	11	74547984626	Nazife	KIBAR	Aynur	Burak	Kiz	1	3	38
11	12	58645321584	Sinan	AKARSU	Fatma	Sedat	Erkek	1	3	38
12	13	28741667542	Abdul...	CAN	Medine	Hakki	Erkek	1	3	38
13	14	89547782148	Yusuf	YANIK	Emine	Ismail	Erkek	1	3	38
14	15	89634567842	Cihan	LOKMAN	Hülya	Beyazit	Erkek	1	3	38

Şekil 8.3. "Birey" ve "Birey_Danisman" tablolarının Kartezyen çarpımı

EŞİTİ OLAN BİRLEŞTİRME

Eşiti olan birleştirme sonucunda birleştirilen iki tablonun her ikisinde de bulunan ortak değere sahip olan satırların listelendiğinden daha önce bahsedildi. Bu birleştirmede bir tablo üzerinde bulunan değerler diğer tablonun ilgili alanı ile eşleştirilip sadece eşleşen değerler birleştirilir.

Klasik birleştirmede "Where" operatöründen sonra birleştirme şartı yazılır.

Eşiti olan birleştirmeyi "iç birleştirme" (Inner Join) ve klasik birleştirme ile gerçekleştirebiliriz. İlk olarak klasik birleştirmenin nasıl yapılabileceğini işleyelim.

Klasik Birleştirme

Bu birleştirme türünde tabloların eşleştirilmesi için gerekli koşul, "Where" ifadesinden sonra kullanılır. İki veya daha fazla tabloda yeralan aynı türde verileri içeren alanlar, "Where" ifadesinden sonra kullanılan koşul ile eşitlenerek klasik birleştirme işlemi gerçekleştirilir. Klasik birleştirmenin söz dizimi aşağıdaki gibidir:

Söz Dizimi:

```
SELECT
```

Select_Listesi

FROM

Tablo1, Tablo2

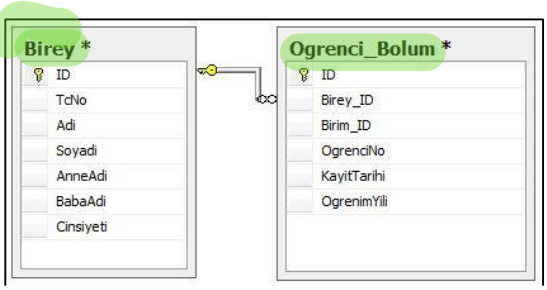
WHERE

Tablo1.Sütun1 = Tablo2.Sütun2

Örnek olarak "Birey" tablosu ile "Ogrenci_Bolum" tablolarını kullanarak öğrencilerin "Birey" tablosundaki ad, soyad, T.C. no, anne adı ve baba adı bilgilerini ve aynı öğrenciler için "Ogrenci_Bolum" tablosundaki numara (OgrenciNo) bilgilerine erişelim.

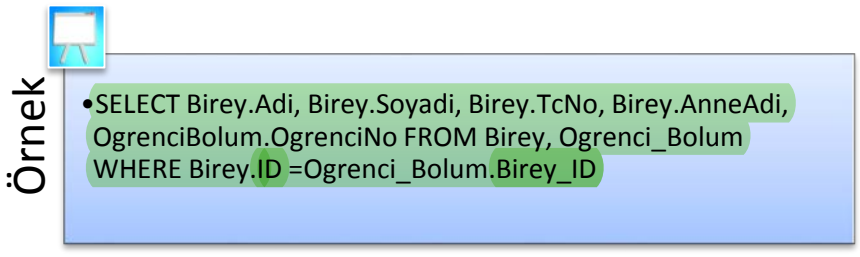
Klasik birleştirmede tablolar arasına virgül konur.

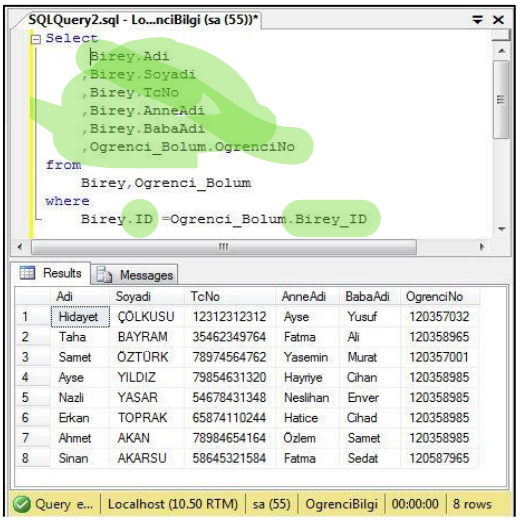
SQL koduna geçmeden önce tabloların yapısını diyagram üzerinde inceleyelim. İki tablo birbirine bağlanırken "Birey" tablosunun birincil anahtar alanı olan "ID" alanı ile "Ogrenci_Bolum" tablosundaki "Birey_Id" alanının eşleştirildiğini Şekil 8.4.te görebilirsiniz.



Şekil 8.4. "Birey" Ve "Ogrenci_Bolum" Tabloları Arasındaki İlişki

İki tabloda eşleşen alanlar "Where" deyimi sonrasında eşitlendiğinde iki tablonun nasıl birleştirilebildiğini Şekil 8.5.te görebilirsiniz. Örneğimizde 8 adet kayıt iki tablonun anahtar alanları üzerinden eşleştirilmiştir. Böylece "Ogrenci_Bolum" tablosunda karşılığı bulunan "Birey" bilgilerine erişilebilmiştir. Mesela "Hidayet" adlı öğreninin, adı, soyadı, TC No, anne adı ve baba adı bilgileri "Birey" tablosundan çekilirken öğrenci numarası bilgisi ("120357032") "Ogrenci_Bolum" tablosundan çekilerek aynı satırda listelenmiştir.





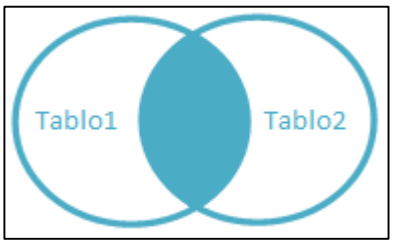
Şekil 8.5. "Birey" Ve "Ogrenci_Bolum" Tablolarının İlişkilendirilerek Sorgulanması

İç Birleştirme (Inner Join)

Birleştirme koşulunda =,<,>,!=,!<,!>,<> gibi Karşılaştırma operatörleri kullanılır.

Eşiti olan birleştirme türlerinden biri de iç birleştirme (Inner Join)'dir. İç birleştirme, klasik birleştirmeyle aynı işleve sahiptir yani sorgu sonucunda aynı sonuçları döndürür. En çok kullanılan tablo birleştirme yöntemi olan iç birleştirmede tablolar birleştirilirken, tablolarda bulunan ortak alanlar kullanılır.

İç birleştirmede bir seferde sadece iki tablo birleştirilebilmektedir. Ancak aynı sorgu içinde birden fazla iç birleştirme uygulanarak çok sayıda tablo tek bir sorguda birleştirilebilmektedir. En fazla kaç tablo birleştirilebileceği ise veri tabanının üzerinde bulunduğu programla alakalı bir durumdur. Örneğin MSSQL'de en fazla 256 tablo iç birleştirme işlemine sokulabilir. İç birleştirme Şekil 8.6.da görüldüğü gibi iki tablonun alanlarının ortak olduğu kayıtları getirmektedir.



Şekil 8.6. İç Birleştirme / Klasik Birleştirme

İç birleştirmenin klasik birleştirmeden tek farkı yazılış biçimidir. Klasik birleştirmede tablolar, "From" ifadesinden sonra aralarına virgül (",") konularak yazılarak birleştirme koşulları "Where" deyiminden sonra tanımlanırken iç birleştirmede sadece birinci tablo "From" ifadesinden sonra yazılır, diğer tablolar "Inner Join" deyimleriyle tesbih taneleri gibi sorguya eklenir ve eklenen her bir tablo için birleştirme koşulları "On" deyiminden sonra tanımlanır.

Tabloların birleştirme koşulunda için genellikle "=" operatörü kullanılır. En performanslı olan operatör de budur. Ancak diğer karşılaştırma operatörleri de kullanılabilmektedir. İç birleştirmenin iki tablo için uygulanması aşağıdaki gibidir:

Söz Dizimi:

SELECT

Select_Listesi

FROM

Tablo1

INNER JOIN

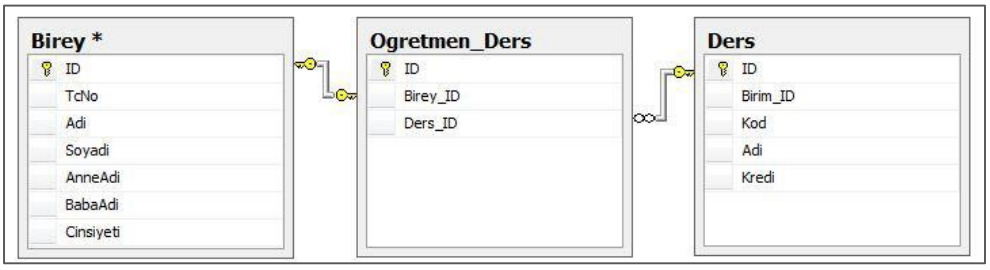
Tablo2

ON

Tablo1.Sütun1 Karşılaştırma_İfadesi Tablo2.Sütun2

Örnek olarak derslere ilişkin tüm bilgileri görüntüleyen SQL sorugusunu yazalım. Bunun için üç tabloyla bağlantı kurulması gerekmektedir. Hoca bilgisine erişmek için "Birey" tablosuna, hocaların hangi dersi verdiği bilgisi için "Ogretmen_Ders" tablosuna ve son olarak derslerin "Adı", "Kodu" ve "Kredisi" bilgilerine erişmek "Ders" tablosuna bağlantı kurulmaktadır.

Tabloların yapısı Şekil 8.7.deki gibidir. "Ogretmen_Ders" tablosunun iki tabloyu birleştirici rolü burada açıkça görülmektedir. "Ogretmen_Ders" tablosundaki "Birey_ID" alanı, "Birey" tablosunda birincil anahtar olan "ID" alanine karşılık gelirken "Ogretmen_Ders" tablosundaki "Ders_ID" alanı, "Ders" tablosunda birincil anahtar olan "ID" alanına karşılık gelmektedir.

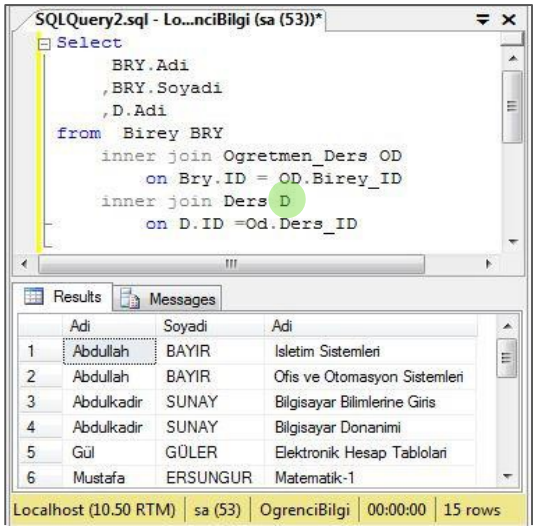


Şekil 8.7. "Birey","Ogretmen_Ders" Ve "Ders" Tabloları Arasındaki İlişki

Şimdi de iç birleştirme yapısı kullanarak bu üç tabloyu birleştiren SQL kodunu yazalım. Öncelikle "Birey" tablosunu "Ogretmen_Ders" tablosu ile birleştirelim. İki tablo arasına "Inner Join" deyimini yazdıktan sonra birleştirme koşulunu (Bry.Id = OD.Birey_ID) "on" deyiminden sonra yazalım. Sonrasında "Ders" tablosunu da "ID" alanı üzerinden "Ogretmen_Ders" tablosunun "Ders_ID" alanı üzerinden ilişkilendirelim (D.ID = OD. Ders_ID). Birleştirme yapılırken de "Birey" tablosuna "BRY", "Ogretmen_Ders" tablosuna "OD" ve "Ders" tablosuna "D" takma adını kullanalım.

Örnek

•SELECT Bry.Adi, Bry.Soyadi, D.Adi FROM Birey Bry INNER JOIN Ogretmen_Ders Od on Bry.ID = Od.Birey_ID INNER JOIN Ders ON D.ID =Od.Ders_ID

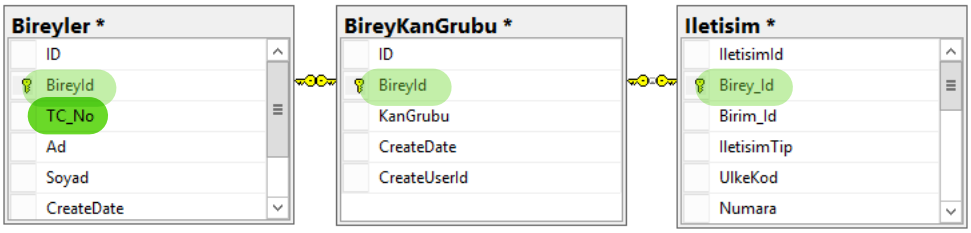


Şekil 8.8. "Birey","Ogretmen_Ders" Ve "Ders" Tablolarının İlişkilendirilmesi

Her iki tablonun birleştirme koşulu "On" deyiminden sonra tanımlanmaktadır.

İç birleştirmeyi bir örnekle daha pekiştirelim. Kan grubu "Arh-" olan öğrencilerin T.C. numarası, ad, soyad ve telefon numaraları bilgilerini görüntüleyen SQL sorgusunu yazalım. Bunun için Şekil 8.9.da ilişkileri gösterilen üç tablonun birleştirilmesi gerekmektedir. "Bireyler" tablosunda öğrencilerin nufüs bilgileri, "BireyKanGrubu" tablosunda öğrencilerin kan grupları ve "Iletisim" tablosunda da öğrencilerin telefon numaraları bilgileri tutulmaktadır.

"Bireyler" tablosu ile "BireyKanGrubu" tablosunu birleştirirken her iki tabloda da yer alan "BireyId" alanları eşitlenecektir. Daha sonraki "Inner Join" işlemi için "Iletisim" tablosu "BireyKanGrubu" tablosuyla birleştirilecektir. Bunun için de "BireyKanGrubu" tablosundaki "BireyId" ve "Iletisim" tablosundaki "Birey_Id" alanları eşitlenecektir.



Şekil 8.9. "Bireyler","Bireykangrubu" Ve "Iletisim" Tabloları Arasındaki İlişki

Örnek

•SELECT Bry.TC_No, Bry.Ad, Bry.Soyad, I.numara FROM Bireyler Bry INNER JOIN BireyKanGrubu Kg on Bry.BireyID = Kg.Birey_ID and Kg.KanGrubu='Arh-' INNER JOIN Iletisim I ON Kg.BireyID =I.Birey_Id

```
SELECT Bry.TC_No
      ,Bry.Ad
      ,Bry.Soyad
      ,I.numara
FROM Bireyler Bry
    INNER JOIN BireyKanGrubu Kg
        ON Bry.BireyID = Kg.BireyID
           and Kg.KanGrubu='Arh-'
      INNER JOIN Iletisim I
        ON Kg.BireyID =I.Birey_Id
```

100 %

Results | Messages

	TC_No	Ad	Soyad	numara
1	12332134512	Erkan	Bayramoğlu	593444111

Şekil 8.10. "Bireyler","Bireykangrubu" Ve "Iletisim" Tablolarının İlişkilendirilmesi

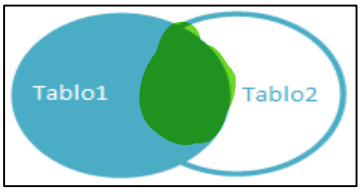
EŞİTİ OLMAYAN BİRLEŞTİRME

Birleştirme şartına uymayan kayıtların da sonuç olarak dönmesi için dış birleştirme kullanılır.

Eşiti olmayan birleştirmeye dış birleştirme (Outer Join) de denilmektedir. Eşiti olan birleştirmede birinci ve ikinci tablodan birleştirme ölçütüne uyan kayıtlar görüntülenirken birleştirme şartına uymayan kayıtlar sonuç olarak dönmez. Fakat eşiti olmayan birleştirmede birleştirme şartına uyan kayıtlarla birlikte şartı sağlamayan diğer kayıtlar da listelenmektedir. Eşiti olmayan birleştirme "Left Outer Join", "Right Outer Join" ve "Full Outer Join" yapıları ile gerçekleştirilmektedir. Takip eden başlıklar altında bu yapılar ayrıntılı olarak işlenecektir.

Sol Dış Birleştirme (Left Outer Join)

Eşiti olmayan birleştirmede iki tablodan birincisi sol (left), ikincisi ise sağ (right) tablo olarak isimlendirildiğini düşünebilirsiniz. İki tablo birleştirilirken birinci tablodaki (sol tablo) tüm kayıtların getirilmesi ikinci tablodan (sağ) ise birleştirme şartına uyan kayıtların getirilmesi istenirse "Left Outer Join" kullanılır. Böylece ikinci tabloda karşılığı olmayan kayıtlar için ikinci tablodaki sütun sayısı kadar "Null" değeri döner. Birleştirme sonucunda birinci tablodaki veriler sonuç listesinin sol tarafına ikinci tablodan gelen veriler ise sağ tarafına yerleşmektedir.



Şekil 8.11. Sol Dış Birleştirme

Sol dış birleştirme Şekil 8.11.da görüldüğü gibi birinci tablonun tüm kayıtlarını ve ikinci tablonun ise koşula uyan kayıtlarını getirmektedir. Dış birleştirme de iç birleştirme ile benzer yazım kurallarına sahiptir. Ancak takdir edeceğiniz gibi sol dış birleştirmede tabloların yazılış sırası önemlidir. "Left Join" deyiminden önce yazılan tablo birinci (left), "Left Join" deyiminden sonra yazılan tablo ise ikinci tablo (right) kabul edilmektedir.

Sol dış birleştirmede "Outer" kullanımı isteğe bağlıdır.

Söz Dizimi:

SELECT

Select Listesi

FROM

Tablo1

LEFT [OUTER] JOIN *Tablo2*

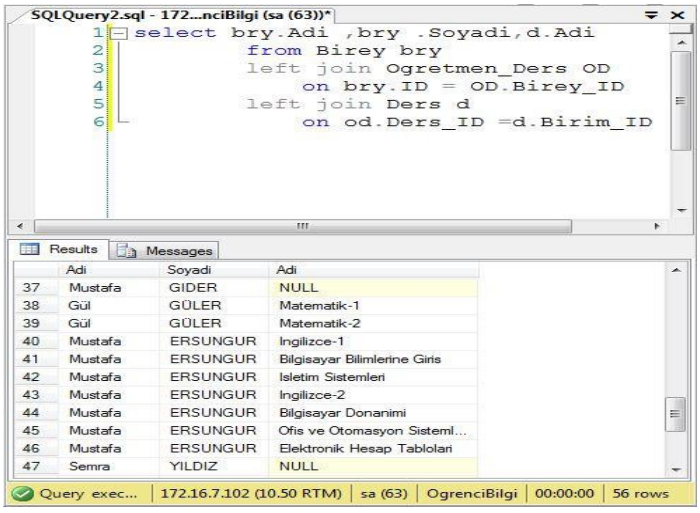
ON Tablo1.Sütun1=Tablo2.Sütun2

Sol dış birleştirmeye örnek olması için tüm hoca bilgilerini görüntülerken ders veren hocalar için ders bilgilerini de görüntüleyen SQL sorgusunu yazalım. "Birey" tablosundaki tüm kayıtların görüntülenmesini istediğimiz için "From" ifadesinden sonra "Birey" tablosuna bağlantı kuralım. Diğer tablolarını (Ogretmen_Ders ve Ders tabloları) ise "Left Join" yapıları içine yerleştirelim. Tablo yapılarını hatırlamak için Şekil 8.7.yi inceleyebilirsiniz.

Sol dış birleştirmede birinci tablodaki tüm kayıtlar, ikinci tabloda ise şarta uyan kayıtlar gelir.

Örnek

•SELECT Bry.Adi, Bry.Soyadi, D.Adi FROM BİREY Bry LEFT JOIN Ogretmen_Ders Od ON BRY.ID=Od.Birey_ID LEFT JOIN Ders D ON Od.Ders_ID =D.Birim_ID



Şekil 8.12. "Birey", "Ogretmen_Ders" Ve "Ders" Tablolarının İlişkilendirilmesi

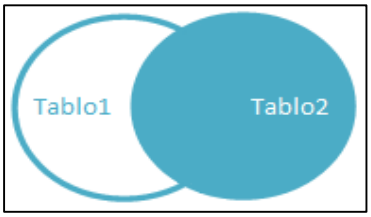
Sağ Dış Birleştirme (Right Outer Join)

Sağ dış birleştirmede ikinci tablodaki tüm kayıtlar, birinci tabloda ise şarta uyan kayıtlar gelir.

Sağ dış birleştirme, sol dış birleştirmeye büyük ölçüde benzemektedir. Sol dış birleştirmede ilk tablodaki tüm kayıtlar ve ikinci tabloda koşula uyan kayıtlar listelenmekteydi. İki tablo birleştirilirken ikinci tablodaki tüm kayıtların getirilmesi ve birinci tablodan ise birleştirme şartına uygun olan kayıtların getirilmesi istenirse sağ dış birleştirme ("Right Outer Join") kullanılır. Sağ birleştirmede birinci tablodan birleştirme şartına uygun olmayan satırlar için birinci tablonun sütun sayısı adedince "Null" değeri döner. Birleştirme sonucunda ise tabloların kullanım sırası dikkate alınır. Yani birinci tablodaki veriler sol tarafa, ikinci tablodan gelen veriler ise sağ tarafa yerleşmiş şekilde gelir.

Söz Dizimi:

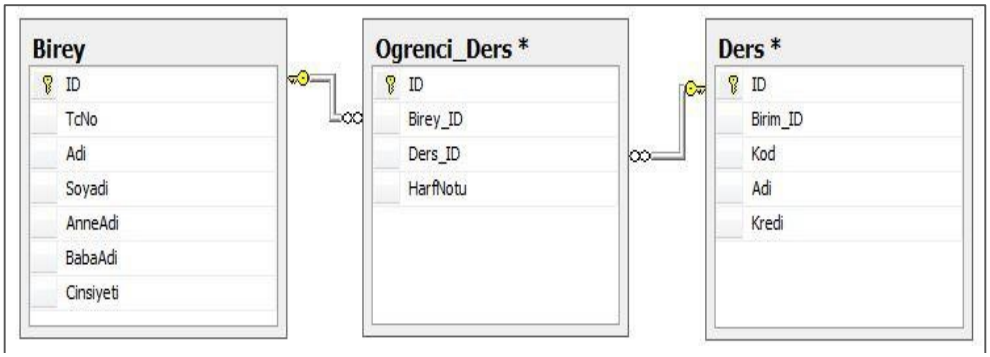
```
SELECT
Select_Listesi
FROM
        Tablo1
RIGHT [OUTER] JOIN Tablo2
ON
Tablo1.Sütun1=Tablo2.Sütun2
```



Şekil 8.13. Sağ Dış Birleştirme

Sağ dış birleştirme yukarıdaki şekilde görüldüğü gibi ikinci (sağdaki) tablonun tüm kayıtlarını ve birinci (soldaki) tablonun ise koşula uyan kayıtlarını getirmektedir.

Sağ ve sol dış birleştirmenin birlikte kullanılmasına örnek olması için tüm bireylerin ad ve soyad bilgilerini görüntülerken ders alması varsa aldığı derslerin adını ve ilgili dersten alınan harf notunu görüntüleyen SQL sorgusunu yazalım. Tablo yapıları, Şekil 8.14.deki gibidir.



Şekil 8.14. "Birey", "Ogretmen_Ders" Ve "Ders" Tabloları Arasındaki İlişki

"Birey" tablosu Şekil 8.14.de görüldüğü gibi "Ogrenci_Ders" tablosuyla ilişkilidir. Dolayısıyla bu iki tablonun ardışık olarak çağırılması gerekmektedir. Birey tablosundaki tüm kayıtların görüntülenmesini istediğimiz için de "Right Join" yapısından sonra "Birey" tablosunu, öncesinde ise "Ögrenci_Ders" tablosunu bağlamalıyız. "Ders" tablosunu bağlarken de "Right Join" yapısı yerine "Left Join" yapısı kullanmalıyız.

Örnek

•SELECT Bry.Adi, Bry.Soyadi, Od.HarfNotu, D.Adi FROM Ogrenci_Ders Od RIGHT JOIN Birey Bry ON Od.Birey_ID=BRY.Id LEFT JOIN Ders D ON D.Id=Od.Ders_ID

SQLQuery6.sql - 17...ciBilgi (sa (298))*

```
1  select
2         BRY.Adi
3        ,BRY.Soyadi
4        ,OD.HarfNotu
5        ,D.Adi
6         from Ogrenci_Ders  OD
7           right join Birey BRY
8               on Od.Birey_ID = bry.ID
9           left join Ders D
10              on D.ID =Od.Ders_ID
```

Results | Messages

	Adi	Soyadi	HarfNotu	Adi
12	Hidayet	ÇÖLKUSU	Z	Türk Dili-2
13	Taha	BAYRAM	AA	Bilgisayar Bilimlerine Giris
14	Ayse	YILDIZ	BA	Isletim Sistemleri
15	Nazli	YASAR	BB	Genel Iktisat
16	Erkan	TOPRAK	NULL	NULL
17	Ahmet	AKAN	NULL	NULL
18	Mustafa	TOROMAN	NULL	NULL
19	Selmani	HATIPOGLU	NULL	NULL
20	Samet	ÖZTÜRK	NULL	NULL

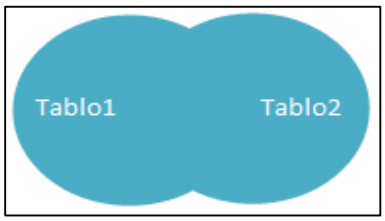
Query... | 172.16.7.102 (10.50 RTM) | sa (298) | OgrenciBilgi | 00:00:00 | 54 rows

Şekil 8.15. "Birey", "Ogretmen_Ders" Ve "Ders" Tablolarının İlişkilendirilmesi

Tüm Dış Birleştirme (Full Outer Join)

Dış birleştirme kapsamında göreceğimiz son yapı tüm dış birleştirme ("Full Outer Join"). Tüm dış birleştirme sonucu oluşan tablo için "Left Join" ve "Right Join" in birleşmesinden oluşan bir tablodur diyebiliriz. "Full Outer Join" ile iki tablo birleştirildiğinde "Left Join" ya da "Right Join" ayrımı yapılmaz ve birleştirmeye dâhil olan tablolardaki tüm kayıtlar getirilir (Şekil 8.16).

Tüm dış birleştirmede tablolardaki şarta uyan kayıtlara ek olarak şarta uymayan kayıtlar da gelir.



Şekil 8.16. Tüm Dış Birleştirme

Söz Dizimi:

SELECT

Select_Listesi

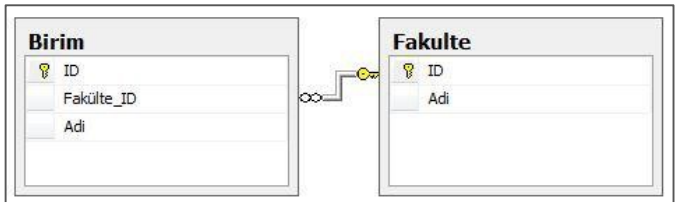
FROM

Tablo1

FULL [OUTER] JOIN *Tablo2*

ON Tablo1.Sütun1=Tablo2.Sütun2

Tüm dış birleştirmeye örnek olması için fakülteler ve onların altında yer alan bölümleri (birim) getiren sorguyu yazalım. İlgili tabloların şema gösterimi Şekil 8.16.teki gibidir.



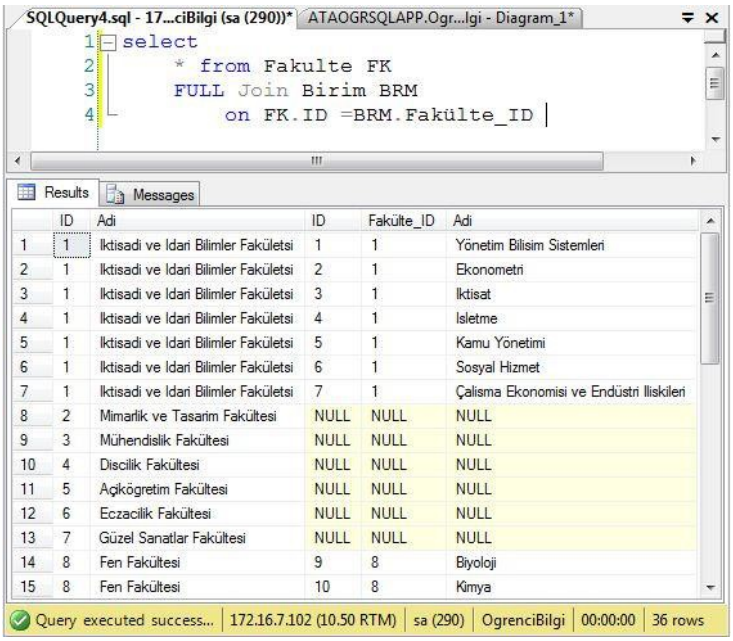
Şekil 8.17. "Birim" Ve "Fakulte" Tabloları Arasındaki İlişki

Tüm dış birleştirmede geri dönen kayıt sayısı, tablolardaki kayıt sayıları toplamından ortak kayıtların çıkarılmasıyla bulunur.

"Fakulte" tablosu, "Birim" tablosuna bağlanırken "Birim" tablosundaki "Fakulte_ID", "Fakulte" tablosunda birincil anahtar olan "ID" alanı ile ilişkilidir. Birleştirmeyi tüm dış birleştirme ile yaptığımız için sorgudan dönen listede tüm fakültelerin ve tüm birimlerin görüntülenmesi sağlanmaktadır (Şekil 8.18).

Örnek

```
•SELECT * FROM Fakulte Fk FULL JOIN Birim Brm ON
 Fk.ID=BRM.Fakülte_ID
```



	ID	Adi	ID	Fakülte_ID	Adi
1	1	Iktisadi ve Idari Bilimler Fakületesi	1	1	Yönetim Bilisim Sistemleri
2	1	Iktisadi ve Idari Bilimler Fakületesi	2	1	Ekonometri
3	1	Iktisadi ve Idari Bilimler Fakületesi	3	1	Iktisat
4	1	Iktisadi ve Idari Bilimler Fakületesi	4	1	Isletme
5	1	Iktisadi ve Idari Bilimler Fakületesi	5	1	Kamu Yönetimi
6	1	Iktisadi ve Idari Bilimler Fakületesi	6	1	Sosyal Hizmet
7	1	Iktisadi ve Idari Bilimler Fakületesi	7	1	Çalisma Ekonomisi ve Endüstri Iliskileri
8	2	Mimarlik ve Tasarim Fakültesi	NULL	NULL	NULL
9	3	Mühendislik Fakültesi	NULL	NULL	NULL
10	4	Discilik Fakültesi	NULL	NULL	NULL
11	5	Açiköğretim Fakültesi	NULL	NULL	NULL
12	6	Eczacilik Fakültesi	NULL	NULL	NULL
13	7	Güzel Sanatlar Fakültesi	NULL	NULL	NULL
14	8	Fen Fakültesi	9	8	Biyoloji
15	8	Fen Fakültesi	10	8	Kimya

Şekil 8.18. "Birim" Ve "Fakulte" Tablolarının İlişkilendirilmesi

Bireysel Etkinlik

- Klasik birleştirme ile dâhili birleştirmenin performans yönünden farkının olup olmadığını testler yaparak açıklayınız.
- Sol dış birleştirme ile sağ dış birleştirmenin performans yönünden farkının olup olmadığını testler yaparak açıklayınız.
- Tüm dış birleştirme ile kartezyen birleştirmenin performans farkının olup olmadığını testler yaparak açıklayınız.
- Sayısal türde alanlar ile yapılan birleştirmenin, sayısal olmayan alanlar ile yapılan birleştirmeden daha performanslı çalıştığını test ediniz.

Özet

- Normalizasyon formları uygulanarak birden fazla tabloda yer alan verileri tek bir sonuçta gösterebilmek için birleştirme (join) işlemi uygulanır. İki tabloyu birleştirirken bu tabloların ortak olan sütunları birleştirme şartında eşitlenir. Birleştirilecek ortak alanların türlerinin aynı olması gerekmektedir.
- Sorgulama esnasında bir tabloya veya kolona takma ad verilmesi sorgunun yazılmasını kolaylaştırmaktadır ve sorguyu anlaşılır kılmaktadır. Sorgular içerisinde hesaplanan değerler görüntülenirken de takma ad kullanılmadığında "No column name" yani "Kolon adı yok" başlığı altında listelenmektedir. Birleştirilen tablolarda aynı isimde kolonlar bulunması durumunda bu kolonlara erişilirken tablo adı uzun hâliyle veya takma adı ile kullanılmalıdır. Aksi takdirde hata sorgu sonucunda hata alınacaktır.
- Tabloları birbirleriyle ilişkilendirmek için "Cross Join", "Klasik Join","Inner Join","Outer Join" yöntemleri kullanılmaktadır. "Outer Join", "Left Join", "Right Join" ve "Full Join" olmak üzere 3 türden oluşmaktadır.
- Çapraz birleştirme, üzerinde işlem yapılan iki tablodaki kayıtları çaprazlamak için kullanılır. İlişkili olsun olmasın birleştirilecek tabloların tüm satırlarını listemektedir. Üzerinde işlem yapılacak iki tablo x ve y tane satırdan oluşuyorsa kartezyen çarpımı sonucu x*y tane satırdan oluşan bir tablo ortaya çıkar.
- Eşiti olan birleştirme sonucunda birleştirilen tablolarda ortak değer sahip olan satırlar sonuç olarak döner. Eşiti olan birleştirme "İç Birleştirme (Inner Join)" ya da klasik "Klasik Birleştirme" ile gerçekleştirilebilir.
- Dış birleştirme (Outer Join) olarak da adlandırılan Eşiti Olmayan birleştirmede (Outer Join) birleştirme şartına uyan kayıtlarla birlikte şartı sağlamayan diğer kayıtlar da sonuç olarak döner. Eşiti olmayan birleştirme "Left Outer Join", "Right Outer Join" ve "Full Outer Join" ile gerçekleştirilir.
- İki tablo birleştirilirken ilk tabloda yer alan tüm kayıtların listelenmesi, ikinci tablodan ise tabloların birleştirme şartını sağlayan kayıtların listelenmesi istenirse "Left Outer Join" kullanılır. Bu birleştirmede ikinci tabloda karşılığı olmayan satırlar için "NULL" değeri döner. Birleştirme sonucunda ilk tablodaki veriler sol tarafa ikinci tablodan gelen veriler de sağ tarafa yerleşmiş olarak listelenir.
- Birleştirilen iki tablodan ikinci tablodaki tüm kayıtların birinci tablodan ise birleştirme koşuluna uyan kayıtların listelenmesi istenirse "Right Outer Join" kullanılır. Birinci tablodan birleştirme koşuluna uymayan satırlar için "NULL" değeri döner. Birleştirme sonucunda birinci tablodaki veriler sol tarafa ikinci tablodan gelen veriler ise sağ tarafa listelenir. Sağ dış birleştirmede birleştirilen iki tablodan ikinci (sağdaki) tablonun tüm kayıtlarını, birinci (soldaki) tablonun ise koşula uyan kayıtları listelenmektedir.
- Tüm dış birleştirme (Full Outer Join), "Left Join" ile "Right Join" in birleşmesinden oluşan tablodur denilebilir. Tüm dış birleştirmede Left Join ya da Right Join ayrımı yapılmaksızın iki tabloda yer alan tüm kayıtlar listelenir.

DEĞERLENDİRME SORULARI

"SELECT FROM WHERE

Select_Listesi Tablo1, Tablo2

Tablo1.Sütun1 = Tablo2.Sütun2"

1. Yukarıdaki sorgu yazım biçimi hangi tür birleştirmeye örnektir?

   a) İç birleştirme (Inner Join)
   b) Klasik birleştirme
   c) Sol dış birleştirme (Left Outer Join)
   d) Çapraz birleştirme (Cross Join)
   e) Sağ dış birleştirme (Right Outer Join)

2. Yukarıdaki gibi çapraz sorgu ile birleştirilen "Ders" tablosunda 7 ve "Sinav" tablosunda 10 kayıt olduğuna göre sorgudan kaç kayıt döner?

   a) 3
   b) 7
   c) 10
   d) 17
   e) 70

3. Aşağıdaki sorguların hangisinden "Ders" tablosundaki tüm kayıtlar dönmez?

   a) "SELECT * FROM Ders d LEFT JOIN Ogretmen_Ders od ON d.Ders_Id = od.Ders_Id"
   b) "SELECT * FROM Ders CROSS JOIN Ogretmen_Ders "
   c) "SELECT * FROM Ders, Ogretmen_Ders "
   d) "SELECT * FROM Ders FULL [OUTER] JOIN Ogretmen_Ders ON d.Ders_Id = od.Ders_Id"
   e) "SELECT * FROM Ders d RIGHT JOIN Ogretmen_Ders od ON d.Ders_Id = od.Ders_Id"

4. Birleştirilen iki tablodan, birinci tablodaki tüm satırları ve ikinci tablodan ise birleştirme şartına uyan satırların getirildiği birleştirme türü aşağıdakilerden hangisidir?

   a) Sol dış birleştirme (Left Outer Join)
   b) Tüm dış birleştirme (Full Outer Join)
   c) Sağ dış birleştirme (Right Outer Join)
   d) İç birleştirme (Inner Join)
   e) Klasik birleştirme

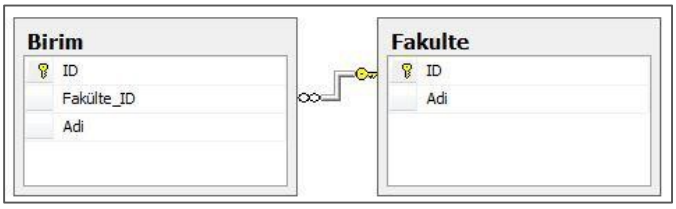
5. Sağ dış birleştirme (Right Outer Join) ile ilgili aşağıdakilerden hangisi yanlıştır?

   a) Eşiti olmayan birleştirme türündendir.
   b) Birinci tablodaki tüm kayıtlar gelmektedir.
   c) Birleştirme koşulu "ON" ifadesinden sonra yazılmalıdır.
   d) "Outer" ifadesinin kullanımı isteğe bağlıdır.
   e) Birinci tablodan birleştirme şartına uymayan satırlar için "NULL" değeri döner.

SELECT * FROM Birey AS B1 INNER JOIN Birey B2 WHERE B1.ID =B2.ID

6. Yukarıdaki sorguyla ilgili yapılan yazım hatası aşağıdakilerden hangisidir?

   a) "Birey" tablosu kendisiyle birleştirilmiştir.
   b) "Birey" tablosuna "B2" takma adı verilirken "AS" ifadesi kullanılmamıştır.
   c) "ON" ifadesi yerine "WHERE" ifadesi kullanılmıştır.
   d) "left join" yerine "Inner join" ifadesi kullanılmıştır.
   e) "B1.*,B2.*" ifadesi yerine "*"ifadesi kullanılmıştır.

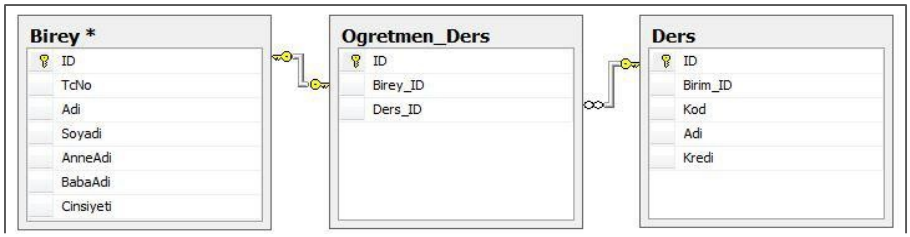


7. Tablo yapıları yukarıdaki gibi olan "Birim" ve "Fakulte" tabloları kullanılarak fakülte bilgisi doğru girilmeyen (Fakulte tablosunda karşılığı olmayan) birimleri listeleyen sorgu aşağıdakilerden hangisidir?
   a) SELECT * FROM Birim WHERE Fakülte_ID IS NULL
   b) SELECT * FROM Fakulte WHERE ID IS NULL
   c) SELECT * FROM Birim LEFT JOIN Fakulte f ON Fakülte_ID=f.ID WHERE f.ID IS NULL
   d) SELECT * FROM Fakulte LEFT JOIN Birim f ON Fakülte_ID=f.ID WHERE f.ID IS NULL
   e) SELECT * FROM Birim RIGHT JOIN Fakulte f ON Fakülte_ID=f.ID WHERE f.ID IS NULL

SELECT * FROM Birim RIGHT JOIN Fakulte f ON Fakülte_ID=f.ID WHERE f.ID IS NULL

8. "Birim" tablosunda 5 ve "Fakulte" tablosunda 3 kayıt olduğuna göre yukarıdaki sorgu sonucunda kaç kayıt döner?

   a) 0
   b) 1
   c) 3
   d) 5
   e) 15



Tablo yapıları yukarıdaki gibi olan "Birey", "Ogretmen_Ders" tabloları kullanılarak öğretmenin adı ve verdiği dersin adı birlikte listelenmek isteniyor.

9. Aşağıdaki eşitlemelerden hangisi yanlıştır?
   a) Birey.ID = Ogretmen_Ders.ID
   b) Birey.ID = Ogretmen_Ders.Birey_ID
   c) Ogretmen_Ders.Ders_ID = Ders.ID
   d) Ogretmen_Ders.Ders_ID is not null
   e) Ogretmen_Ders.Ders_ID is not null

10. Aşağıdakilerden hangisi "Birey" tablosundaki kayıt sayısının iki katından fazla kayıt döndürmektedir?
    a) SELECT * FROM Birey, Birey
    b) SELECT * FROM Birey b1, Birey b2 where b1.ID=b2.ID
    c) SELECT * FROM Birey b1 INNER JOIN Birey b2 ON b1.ID=b2.ID
    d) SELECT * FROM Birey b1 LEFT JOIN Birey b2 ON b1.ID=b2.ID
    e) SELECT * FROM Birey b1 CROSS JOIN Birey b2 ON b1.ID=b2.ID

Cevap Anahtarı
1.b, 2.e, 3.e, 4.a, 5.b, 6.c, 7.c, 8.a, 9.a, 10.e

YARARLANILAN KAYNAKLAR

Adar, İ., 2012. SQL Server 2012 Programlama ve Yönetim. İstanbul, 720

Gözüdeli, Y., 2010. Yazılımcılar için SQL SERVER 2008 R2 ve Veritabanı

Elbahadır, H., 2012. T-SQL SQL SERVER 2012. İstanbul, 294
Programlama. Ankara, 671

SORGU OLUŞTURMAK VE ÇEŞİTLERİNİ KULLANMAK -4

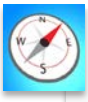
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Arş. Gör. Yakup BAYOĞLU

İÇİNDEKİLER

- Tabloya Satır Ekleme
  - INSERT Deyimi
- NULL Değer Ekleme
- Bir Diğer Tablo Kullanarak Yeni Satır Ekleme
  - SELECT ... INTO ... FROM Deyimi

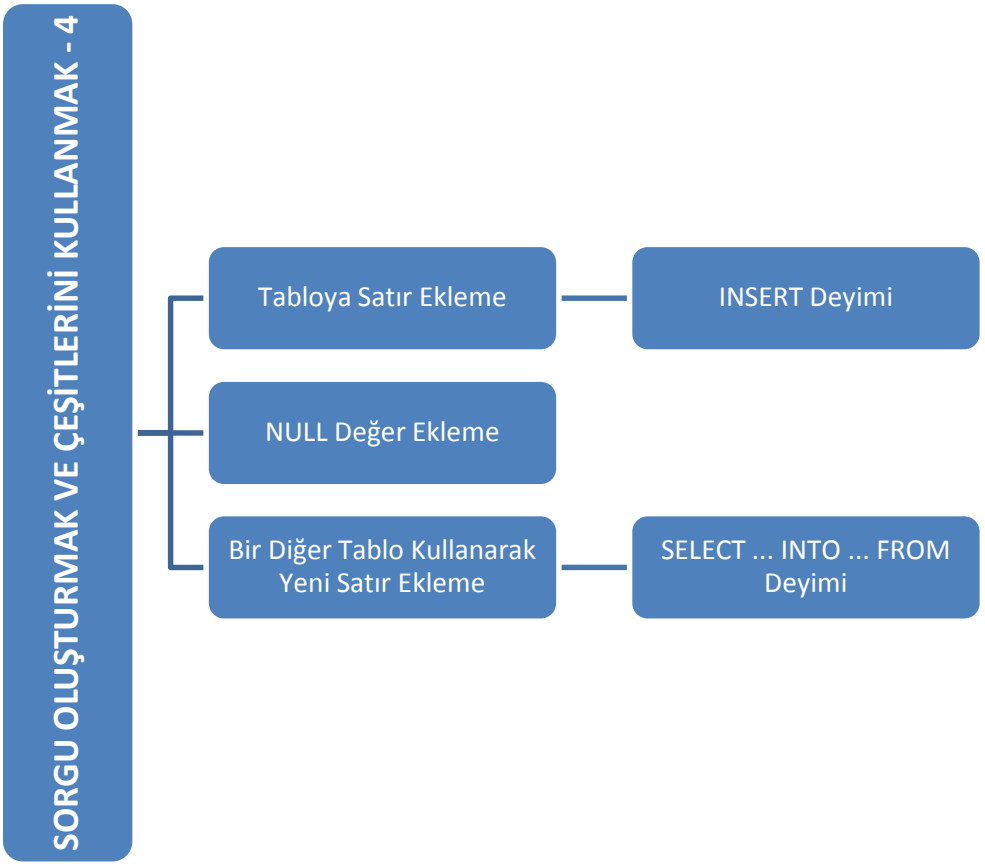


HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Tabloya satır eklemeyi,
  - NULL değer eklemeyi,
  - Ekleme işlemindeki kısıtları ve dikkat edilmesi gereken durumları öğrenebileceksiniz.

ÜNİTE

9


SORGU OLUŞTURMAK VE ÇEŞİTLERİNİ KULLANMAK - 4
Tabloya Satır Ekleme
INSERT Deyimi
NULL Değer Ekleme
Bir Diğer Tablo Kullanarak Yeni Satır Ekleme
SELECT ... INTO ... FROM Deyimi

GİRİŞ

Önceki birkaç ünite boyunca veri tabanından çeşitli sorgulamalar yapmayı ve veriler üzerinde analiz yapmayı öğrendik. Tablo(lar) üzerinde bu tür sorgulama ve analizlerin yapılabilmesi için tablolarda veri olmalıdır, aksi halde sorgular boş dönecektir. Bu ünitede tablolara nasıl veri ekleneceğini işleyeceğiz.

Tablolara veri eklemek için INSERT INTO' komutu kullanılır. Genel kullanımı "INSERT INTO Tablo Adı (Sütun1, Sütun2, Sütun3, …) VALUES (Değer1, Değer2, Değer3, …)" şeklindedir. Tabloya eklenecek verileri kendimiz yazabileceğimiz gibi ekleme yapılacak olan tablo veya veri tabanındaki diğer tablolardaki, hatta sunucu üzerindeki başka bir veri tabanına ait olan tablolardaki verileri kullanarak da ekleme yapılabilir. Tabloya aynı anda tek bir satır veya birden çok satır eklenebilir fakat aynı anda sadece tek bir tabloya ekleme yapılabilir. Tabloya veri eklerken veri tiplerine ve veri sırasına dikkat edilmelidir. Tabloya veri eklenirken tüm sütunlara veri girilmesi zorunlu değildir. Fakat NULL değer kabul edilmeyen sütunlara mutlaka veri girilmelidir. Tablodaki birincil anahtar sütunlara veri girilmez, bu sütunların değeri ekleme esnasında SQL Server tarafından otomatik olarak oluşturulur. Aynı şekilde tablodaki hesaplanmış sütunların da içeriği SQL Server tarafından oluşturulacağı için bu sütunlara da veri girilmez. Yabancı anahtar sütunlar için belirtilen değerlerin hedef tabloda bir karşılığı bulunmalıdır. Bir tabloya eklenen en son Id değerine ihtiyaç duyulduğunda @@IDENTITY sistem değişkeni veya IDENT_CURRENT() Fonksiyonu kullanılabilir. Ünite boyunca veri eklemenin çeşitli yöntemlerini ve veri eklerken dikkat edilmesi gereken hususları ve bazı özel durumları inceleyeceğiz.

TABLOYA SATIR EKLEME

INSERT Deyimi

Tabloya veri (satır) eklemek için 'INSERT INTO' komutu kullanılır. (Gözüdeli, 2012). Genel kullanımı aşağıdaki gibidir:

```
INSERT INTO Tablo_Adı
(
     Sütun1,
     Sütun2,
     Sütun3,
     [...]
)
VALUES
(
     Değer1,
     Değer2,
     Değer3,
     [...]
)
```

Tabloya veri eklemek için INSERT komutu kullanılır.

'INSERT INTO' deyiminden sonra tablo adı ve parantez içerisinde veri girilecek olan sütunlar yer alır. 'VALUES' deyiminden sonra yine parantez içerisinde ilgili sütunlara girilecek olan değerler yer alır.

Örnek veri tabanındaki Birey tablosunun ilk 10 kaydına bakalım:

```
SELECT TOP 10
     *
FROM
     Birey
ORDER BY
     ID
```

Results | Messages

	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	2	12312312312	Hidayet	Çölkusu	Ayse	Yusuf	Erkek
2	3	35462349764	Taha	Bayram	Fatma	Ali	Erkek
3	4	79854631320	Ayse	Yildiz	Hayriye	Cihan	Kiz
4	5	54678431348	Nazli	Yasar	Neslihan	Enver	Kiz
5	6	65874110244	Erkan	Toprak	Hatice	Cihad	Erkek
6	7	78984654164	AHMET	Akan	Özlem	Samet	Erkek
7	8	98971354212	Mustafa	Toroman	Irem	Yusuf	Erkek
8	9	38312472124	Selmani	Hatipoglu	Meltem	Musa	Erkek
9	10	78974564762	Samet	Öztürk	Yasemin	Murat	Erkek
10	11	74547984626	Nazife	Kibar	Aynur	Burak	Kiz

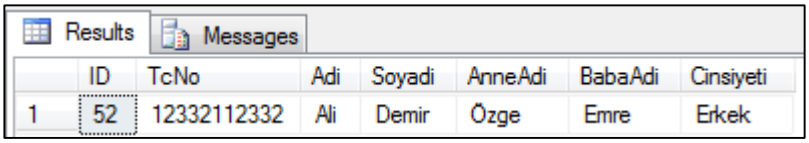
Şekil 9.1. Birey Tablosundaki İlk On Kayıt

Şimdi Birey tablosuna yeni bir satır ekleyelim:

```
INSERT INTO Birey
(
     TcNo,
     Adi,
     Soyadi,
     AnneAdi,
     BabaAdi,
     Cinsiyeti
)
VALUES
(
     '12332112332',
     'Ali',
     'Demir',
     'Özge',
     'Emre',
     'Erkek'
)
```

'INSERT INTO' deyiminden sonra belirtilen sütun isimleri ile 'VALUES' deyiminden sonra yazılan değerler aynı sırada olmalıdır.

Yukarıdaki örnekte Birey tablosuna 'Ali Demir' isimli, anne adı 'Özge', baba adı 'Emre', T.C. kimlik numarası '12332112332' olan erkek bir birey eklemiş olduk.



Results | Messages

	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	52	12332112332	Ali	Demir	Özge	Emre	Erkek

Şekil 9.2. Birey Tablosuna En Son Eklenen Kayıt

Not: 'INSERT INTO' deyiminden sonra belirtilen sütun isimleri ile 'VALUES' deyiminden sonra yazılan değerler *aynı sırada* olmalıdır.

Şimdi, az önce Birey tablosuna eklediğimiz 'Ali Demir' isimli kişi için Kamu Yönetimi bölümüne öğrenci kaydı ekleyelim. Bunun için Ogrenci_Bolum tablosuna ekleme yapmamız gerekecek:

```
INSERT INTO Ogrenci_Bolum
(
     Birey_ID,
     Birim_ID,
     OgrenciNo,
     KayitTarihi,
     OgrenimYili
)
VALUES
(
     52,  -- Ali Demir'in Birey_ID'si
     5,   -- Kamu Yönetimi'nin Birim_ID'si
     '130303555',  -- Öğrenci No
     '2013-09-08', -- Kayıt Tarihi
     1
)
```

Yukarıda Ali Demir (52) için Kamu Yönetimi (5) bölümüne öğrenci numarası '130303555' ve kayıt tarihi '2013-09-08' olan 1. sınıf öğrencisi kaydı eklemiş olduk. 'INSERT INTO ... VALUES' kalıbında 'VALUES' yerine 'SELECT' deyimi de kullanılabilir. Bir önceki örneğimizi 'SELECT' deyimi ile yeniden yazalım:

```
INSERT INTO Ogrenci_Bolum
(
     Birey_ID,
     Birim_ID,
     OgrenciNo,
     KayitTarihi,
     OgrenimYili
)
SELECT
     52,  -- Ali Demir'in Birey_ID'si
     5,   -- Kamu Yönetimi'nin Birim_ID'si
     '130303555',  -- Öğrenci No
     '2013-09-08', -- Kayıt Tarihi
     1
```

Veri girişi için değerler yazılırken karakter veya tarih veri tipindeki sütunların değerleri tek tırnak içerisinde yazılmalıdır.

Not: veri girişi için değerler yazılırken karakter veya tarih veri tipindeki sütunların değerleri tek tırnak içerisinde yazılmalıdır.

Veri girişi yapacağımız tablonun (birincil anahtar olan sütunlar hariç) tüm sütunlarına birden veri girişi yapılacaksa eğer tablo adından sonra sütunlar belirtilmeden de ekleme işlemi yapılabilir. Bir önceki örneğimizde birincil anahtar olan ID sütunu hariç tüm sütunlara veri girişi yapmıştık, aynı işlemi aşağıdaki kod ile de gerçekleştirebiliriz:

```
INSERT INTO Ogrenci_Bolum
SELECT
     52,  -- Ali Demir'in Birey_ID'si
     5,   -- Kamu Yönetimi'nin Birim_ID'si
     '130303555',  -- Öğrenci No
     '2013-09-08', -- Kayıt Tarihi
     1
```

Tabloya aynı anda birden fazla satır eklememiz de mümkündür; bunun için 'UNION [ALL]' deyimi kullanılır. Birey tablosuna Yusuf ve Yeliz isimli iki kardeşin kayıtlarını aynı anda ekleyelim:

```
INSERT INTO Birey
SELECT
     '47852147852',
     'Yusuf',
     'Yerli',
     'Emine',
     'Mehmet',
     'Erkek'
UNION [ALL]
SELECT
     '12365478963',
     'Yeliz',
     'Yerli',
     'Emine',
     'Mehmet',
     'Kız'
```

'UNION' deyimi ile (SQL Server'in müsaade ettiği limiti aşmamak kaydı ile) dilediğimiz kadar select ifadesini birleştirip tek bir insert deyimi ile aynı anda ekleme yapabiliriz.

Aynı anda sadece bir tabloya ekleme işlemi yapılabilir.

Bireysel Etkinlik

- 'UNION' ve 'UNION ALL' deyimlerinin farkını araştırınız.
- Bu farkın, tabloya satır ekleme işlemini nasıl etkileyeceğini düşününüz.

Not: Aynı anda sadece *bir* tabloya ekleme işlemi yapılabilir.

Tabloya yeni bir satır eklerken tüm sütunlar için veri eklemek zorunda değiliz. Örneğin bir öğrenci otomasyon sisteminde öğrenciler için, ilk olarak ön kayıt alındığını ve kesin kayıt yapıldıktan sonra kayıt tarihinin oluşturulduğunu düşünelim. Bu durumda ön kayıt alınırken Ogrenci_Bolum tablosuna ekleme işlemi aşağıdaki gibi olacaktır:

```
INSERT INTO Ogrenci_Bolum
(
     Birey_ID,
     Birim_ID,
     OgrenciNo,
     OgrenimYili
)
SELECT
     52,  -- Ali Demir'in Birey_ID'si
     5,   -- Kamu Yönetimi'nin Birim_ID'si
     '130303555',  -- Öğrenci No
     1
```

Dikkat ederseniz yukarıdaki ekleme işleminde sütunları belirtirken KayitTarihi sütununu belirtmedik ve dolayısıyla değerler kısmında da kayıt tarihini yazmadık, KayitTarihi sütunu bu ekleme işleminden sonra NULL (boş) kalacaktır.

Results | Messages

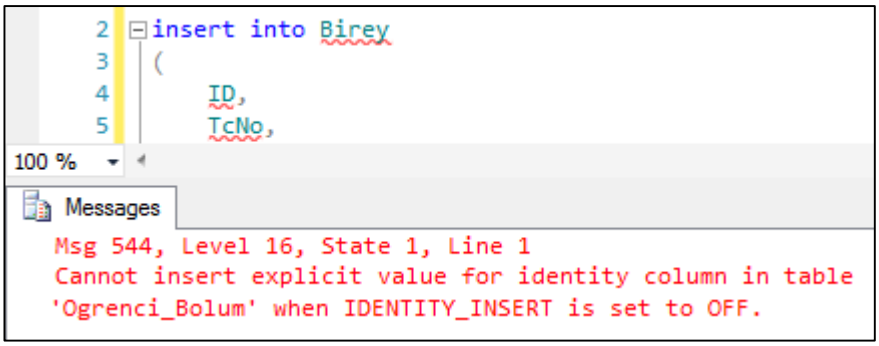
	ID	Birey_ID	Birim_ID	OgrenciNo	KayitTarihi	OgrenimYili
1	21	52	5	130303555	NULL	1

Şekil 9.3. Kayıt Tarihi Olmadan Veri Eklemek

Değeri SQL Server tarafından oluşturlan birincil anahtar durumundaki sütuna veri girişi yapılmaz (*).

Not: Değeri SQL Server tarafından oluşturlan birincil anahtar durumundaki sütuna veri girişi yapılmaz (*), bu sütun(lar)ın alacağı (sıradaki benzersiz) değer SQL Server tarafından otomatik olarak oluşturulur ve sütuna atanır. (Petkovic, 2006).

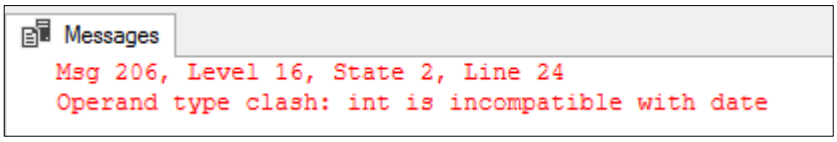
İlk örneğimizde Birey tablosuna yeni bir satır eklerken veri girişi yapılacak sütunların içerisinde birincil anahtar olan ID sütununu belirtmedik fakat ekleme işleminden sonra gördük ki ID sütununa de değer (52) SQL Server tarafından eklendi. Değeri SQL Server tarafından otomatik olarak üretilen birincil anahtar sütununa veri eklemeye çalışılması SQL Server'in hata döndürmesine sebep olur. Bu hatanın bir örneği aşağıda gösterilmiştir.



Şekil 9.4. Değeri SQL Server Tarafından Otomatik Olarak Üretilen Birincil Anahtar Sütununa Veri Eklenmeyeceğine Dair Hata Uyarısı

Veri ekleme işleminde eklenen verilerin veri tipininin tablonun sütun veri tipleri ile uyumlu olması gerekir. Örneğin Ogrenci_Bolum tablosuna ekleme yaparken tarih veri tipindeki KayitTarihi sütununa tamsayı bir değer eklersek şekil 9.5.teki hatayı alırız:

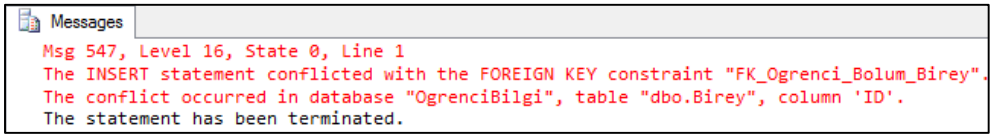
```
INSERT INTO Ogrenci_Bolum
(
     Birey_ID,
     Birim_ID,
     OgrenciNo,
     KayitTarihi,
     OgrenimYili
)
VALUES
(
     52,  -- Ali Demir'in Birey_ID'si
     5,   -- Kamu Yönetimi'nin Birim_ID'si
     '130303555',  -- Öğrenci No
     2013,     -- Kayıt Tarihi
     1
)
```



Şekil 9.5. Sütun Veri Tipinden Farklı Bir Veri Tipi ile Kayıt Eklenildiğinde SQL Server'in Verdiği Hata Mesajı

Dikkate alınması gereken bir diğer husus da yabancı anahtar sütunlarına girilen değerlerin referans tabloda bir karşılığının olması gerektiğidir. Ogrenci_Bolum tablosundaki Birey_ID sütunu Birey Tablosundaki ID sütununa, Birim_ID sütunu da Birim tablosundaki ID sütununa yabancı anahtar olduğu için Birey_ID ve Birim_ID sütunlarına eklenen verilerin ilgili tablolarda karşılığı olması gerekmektedir. Bir önceki örneğimizde Birey_ID sütunu için 52 değerini değil de 525 değerini girmiş olsa idik Birey tablosunda 525 ID değerine sahip bir kayıt olmadığı için SQL Server hata verecekti:

```
INSERT INTO Ogrenci_Bolum
(
     Birey_ID,
     Birim_ID,
     OgrenciNo,
     OgrenimYili
)
SELECT
     525, -- Ali Demir'in Birey_ID'si
     5,   -- Kamu Yönetimi'nin Birim_ID'si
     '130303555',  -- Öğrenci No
     1
```



Şekil 9.6. Yabancı Anahtar Bir Sütuna Referans Tabloda Karşılığı Olmayan Bir Değer Girildiğinde SQL Server'in Verdiği Hata

Hesaplatılmış sütunlara veri eklenemez.

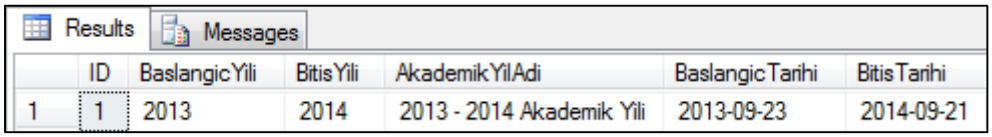
Satır ekleme işleminde dikkat edilmesi gereken bir diğer husus hesaplatılmış sütunlara veri eklenemeyeceğidir. Üniversite akademik takvim bilgilerini tutmak üzere AkademikYil isimli bir tablo oluşturalım:

```
CREATE TABLE AkademikYil
(
     ID             INT IDENTITY,
     BaslangicYili  INT NOT NULL,
     BitisYili      AS BaslangicYili + 1,
     AkademikYilAdiAS   CAST(BaslangicYili AS
VARCHAR(4))
                        + ' - '
                        + CAST(BaslangicYili + 1 AS
VARCHAR(4))
                        + ' Akademik Yılı',
     BaslangicTarihi    DATE NULL,
     BitisTarihi        DATE NULL
)
```

Yukarıdaki AkademikYil tablosunda, BitisYili ve AkademikYilAdi kolonları BaslangicYili sütunundan türetilmektedir (hesaplanmaktadır). BaslangicYili sütununa 2013 değeri girelim ve diğer sütunların alacağı değerleri görelim:

```
INSERT INTO AkademikYil
(
     BaslangicYili,
     BaslangicTarihi,
     BitisTarihi
)
SELECT
     2013,
     '2013-09-23',
     '2014-09-21'
```

```
SELECT * FROM AkademikYil
```

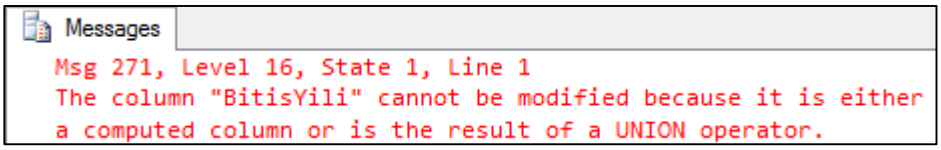


	ID	BaslangicYili	BitisYili	AkademikYilAdi	BaslangicTarihi	BitisTarihi
1	1	2013	2014	2013 - 2014 Akademik Yili	2013-09-23	2014-09-21

Şekil 9.7. Hesaplatılmış Sütun Örneği

BaslangıcYili sütununa 2013 değeri girildiğinde Şekil 9.6.da görüldüğü gibi BitisYili sütununda 2014, AkademikYilAdi sütununda da '2013 - 2014 Akademik Yılı' değerleri oluştu (hesaplatıldı).

BitisYili ve AkademikYilAdi sütunları hesaplatılmış sütun oldukları için bu sütunlara veri girilmek istendiğinde SQL Server aşağıdaki hatayı döndürür.

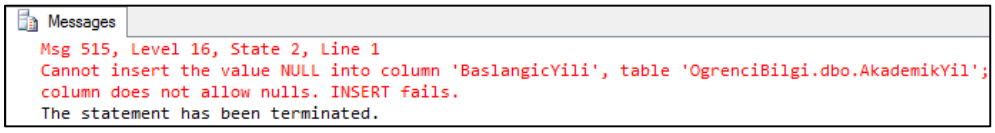


```
Msg 271, Level 16, State 1, Line 1
The column "BitisYili" cannot be modified because it is either
a computed column or is the result of a UNION operator.
```

Şekil 9.8. Hesaplatılmış Bir Sütuna Veri Eklemeye Çalışıldığında SQL Server'in Verdiği Hata

Tabloya yeni bir satır eklenirken tablo tasarımında NULL değer girilmesine izin verilmeyen sütunlar için mutlaka bir değer girilmelidir.

Tabloya yeni bir satır eklenirken tablo tasarımında NULL değer girilmesine izin verilmeyen sütunlar için mutlaka bir değer girilmelidir. Yukarıda AkademikYil tablosu oluşturulurken BaslangicYili sütunu bu şekilde tanımlanmış bir sütun olduğu için AkademikYil tablosuna satır ekleneceği zaman BaslangicYili sütununa mutlaka bir değer girilmelidir. Aksi takdirde SQL server aşağıdaki hata uyarısını verir:



```
Msg 515, Level 16, State 2, Line 1
Cannot insert the value NULL into column 'BaslangicYili', table 'OgrenciBilgi.dbo.AkademikYil';
column does not allow nulls. INSERT fails.
The statement has been terminated.
```

Şekil 9.9. NULL Değer Girilmesine İzin Verilmeyen Bir Sütuna Veri Girilmeden Satır Eklemeye Çalışıldığında SQL Server'in Verdiği Hata Uyarısı

Tablo tasarımında varsayılan (default) değer belirtilmiş bir sütun var ise ve yeni satır ekleme işleminde bu sütun için bir değer girilmemiş ise sütunda NULL değer değil, varsayılan değer oluşur. Örnek olarak Ders tablosunda bool tipinde ZorunluMu isimli bir sütunun olduğunu ve varsayılan değerinin 'true' olduğunu varsayalım. Bu durumda Ders tablosuna yeni satır eklerken ZorunluMu sütunu için bir değer girilmişse eğer girilen değer, girilmemişse varsayılan değer (true) ZorunluMu sütununda oluşur. Bu şekilde yeni ders eklerken zorunlu mu, seçmeli mi olduğu bilinmeyen dersler zorunlu kabul edilerek eklenmiş olur.

NULL Değer Ekleme

Tabloya NULL değer ekleme iki şekilde gerçekleşebilir.

Birinci durum: Daha önce de belirtildiği üzere veri eklenirken 'INSERT INTO' deyiminden sonra belirtilmeyen sütunlara herhangi bir değer eklenmez, yani sütun içeriği NULL olur.

İkinci durum: Veri girilecek sütunlar belirtilirken NULL olması istenen sütun(lar) da dâhil edilir ve 'SELECT' kısmında bu sütunlar için NULL değeri yer alır. Daha önce Ogrenci_Bolum tablosuna KayitTarihi sütunu NULL olacak şekilde veri eklemiştik ki o örnek birinci durum için uygun bir örnektir. Şimdi aynı ekleme işlemini ikinci durumda bahsettiğimiz şekilde yeniden yazalım:

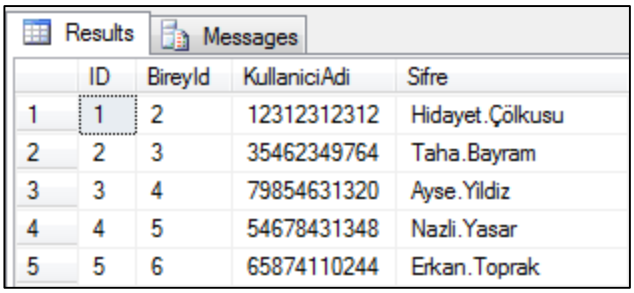

```
INSERT INTO Ogrenci_Bolum
(
     Birey_ID,
     Birim_ID,
     OgrenciNo,
     KayitTarihi,
     OgrenimYili
)
SELECT
     52,  -- Ali Demir'in Birey_IDsi
     5,   -- Kamu Yönetimi'nin Birim_ID'si
     '130303555',  -- Öğrenci No
     NULL,     -- Kayıt Tarihi
     1
```

Bir Diğer Tablo Kullanarak Yeni Satır Ekleme

Şimdiye kadar tabloya eklediğimiz verileri hep değerleri kendimiz yazmak sureti ile gerçekleştirdik. Eklenecek verilerin başka bir tablo veya tablolardan getirilmesi de mümkündür. Hatta aynı sunucu üzerinde yer alan farklı bir veri tabanındaki tablolardan da getirilebilir.

Kullanici adında yeni bir tablo oluşturalım ve içerisine kullanıcı adları T.C. Kimlik numaraları, şifreleri de isim.soyisim formatında olacak şekilde her birey için veri ekleyelim:

```
CREATE TABLE Kullanici
(
     ID        INT IDENTITY,
     BireyId   INT,
     KullaniciAdi  VARCHAR(50),
     Sifre         VARCHAR(50)
)

INSERT INTO Kullanici
SELECT
     ID,
     TcNo,
     Adi + '.' + Soyadi
FROM
     Birey

SELECT * FROM Kullanici
```



Results | Messages

	ID	BireyId	KullaniciAdi	Sifre
1	1	2	12312312312	Hidayet.Çölkusu
2	2	3	35462349764	Taha.Bayram
3	3	4	79854631320	Ayse.Yildiz
4	4	5	54678431348	Nazli.Yasar
5	5	6	65874110244	Erkan.Toprak

Şekil 9.10. Kullanici Tablosunun İlk Beş Satırı

SELECT ... INTO ... FROM Deyimi

Kullanici tablosunda birtakım denemeler yapmak istersek ve verileri bozmamak için de denemeleri üzerinde yapmak için tablonun bir kopyasını almak istersek yazacağımız kod aşağıdaki gibi olacaktır.

'SELECT ... INTO ... FROM' deyimi ile aynı anda tablo oluşturup içine satır(lar) eklenilebilir.

```
CREATE TABLE KullaniciYedek
(
     ID         INT,
     BireyId    INT,
     KullaniciAdi   VARCHAR(50),
     Sifre          VARCHAR(50)
)

INSERT INTO KullaniciYedek
SELECT
     *
FROM
     Kullanici
```

Yukarıda önce KullaniciYedek isimli bir tablo oluşturduk, sonra bu tabloya Kullanıcı tablosundaki tüm satırları ekledik. Sıfırdan bir tablo oluşturup içerisine veri ekleneceği zaman ne yapmamız gerektiğini yukarıdaki örnekte gördük. Fakat bu işlemi 'SELECT ... INTO ... FROM' komutu ile daha pratik bir şekilde gerçekleştirebiliriz. Bu komutun genel kullanımı aşağıdaki gibidir:

```
SELECT
     sütun1,
     sütun2
     [...]
INTO
     YeniTabloAdı
FROM
     ReferansTabloAdı
```

SELECT ifadesinden sonra gelen sütunları içeren bir tablo oluşturulur ve select sorgusunun sonucu bu tabloya eklenir. Bir önceki örneğimizi yeniden yazalım:

```
SELECT
     *
INTO
```

```
        KullaniciYedek
FROM
        Kullanici
```

(*) Birincil anahtar sütununa veri ekleme

Normâlde değeri SQL Server tarafından otomatik olarak üretilen birincil anahtar sütununa veri eklenmez. İstisnai bir durum olarak böyle bir birincil anahtar sütununa veri ekleme ihtiyacı duyarsak geçici olarak ilgili tablonun birincil anahtar sütununa veri eklemeyi aktifleştirebiliriz. Mesela bu ünitenin ilk örneğinde Birey tablosuna eklenilen Ali Demir'in kaydının (ID numarası 52 olan kayıt) tablodan silindiğini ama Birey tablosunun ID sütununa yabancı anahtar barındıran Ogrenci_Bolum, Kullanici, Ogrenci_Ders vs. gibi diğer tablolardaki Birey_ID değeri 52 olan satırların hala mevcut olduğunu varsayalım. 'Ali Demir' kaydını yeniden eklenirse bir önceki seferde aldığı ID değerinden farklı bir ID değeri alacaktır ve bu durumda Ali Demir'e ait diğer tablolardaki kayıtlar, yeni eklenilen Ali Demir kaydı ile ilişkili olmayacaktır; çünkü diğer tablolarda yer alan 'Ali Demir'e ait bilgi içeren satırlarda Birey_ID sütununun içeriği hâlâ 52 dir ama 'Ali Demir'in Birey tablosundaki ID değeri artık 52 değildir. Dolayısı ile silindikten sonra Birey tablosuna aynı bireyi yine eski birey_ID ile yeniden eklemeye ihtiyacımız olabilir, bu durumda yazacağımız kod aşağıdaki gibi olacaktır:

Bir tablo üzerinde geçici olarak ilgili tablonun birincil anahtar sütununa veri ekleme aktifleştirilebilir.

```
SET IDENTITY_INSERT Birey ON

INSERT INTO Birey
(
        ID,
        TcNo,
        Adi,
        Soyadi,
        AnneAdi,
        BabaAdi,
        Cinsiyeti
)
SELECT
(
        52,
        '12332112332',
        'Ali',
        'Demir',
        'Özge',
        'Emre',
        'Erkek'
)

SET IDENTITY_INSERT Birey OFF
```

'`SET IDENTITY_INSERT Birey ON`' komutu ile Birey tablosu için geçici olarak birincil anahtar sütununa veri eklemeye izin verilmiş oldu ve devamında 'Ali Demir' kaydı eski ID değeri (52) ile yeniden eklenmiş oldu. Son satırda yer alan

'SET IDENTITY_INSERT Birey OFF' komutu ile Birey tablosuna geçici olarak verilmiş olan birincil anahtar sütununa veri ekleme izni kaldırılmış oldu.

Tablonun birincil anahtar sütununa veri ekleme özelliği aktif hâle getirildiği zaman, işlem bittikten hemen sonra bu özelliğin geri kapatılması unutulmamalıdır. Yukarıdaki kod, eklenmek istenen birincil anahtar değerinin tabloda yer alıp almadığından bağımsız olarak çalışır; yani Birey tablosunda hâlihazırda Birey_ID değeri 52 olan bir satır var mıdır diye kontrol edilmeden çalışır. Dolayısı ile bu şekilde birincil anahtar sütununa veri eklerken çok dikkatli olmak gerekir. Zira Birey tablosunda aynı Id değerinin birden fazla kere bulunması diğer tablolarla ilişki kurarken sorun teşkil edecektir.

@@IDENTITY değişkeni

@@IDENTITY bir sistem değişkenidir ve veri tabanına en son eklenen birincil anahtar değerini içinde barındırırır.

@@IDENTITY değişkeni veri tabanına en son eklenen birincil anahtar değerini tutar.

Bazen bir tabloya bir satır veri ekledikten sonra, son eklenen bu satırdaki kayıt için kullanılan Id değerinin bilinmesine ihtiyaç olabilir. Örneğin bir Öğrenci Bilgi Sistemine yeni bir öğrenci kaydı eklediğimizi düşünelim. Öğrencinin kaydı sisteme eklenirken farklı tablolara kayıtlar eklenecektir. Öncelikle kişinin özlük bilgilerinin yer aldığı Birey tablosuna kayıt eklenecektir. Akabinde de Ogrenci_Bolum tablosuna bu yeni öğrenci için kayıt eklenecektir. Ogrenci_bolum tablosuna kayıt eklenirken öğrencinin Birey tablosundaki Birey_Id değeri bilinmelidir. İşte bu noktada @@Identity sistem değişkeni kullanılabilir. Bu ünitenin başlarında Ali Demir isimli bir öğrenciyi önce Birey tablosuna eklemiş ve akabinde de Ali Demir için Ogrenci_Bolum tablosuna veri eklemiştik. Bu işlemi aşağıdaki gibi yapabiliriz:

```
INSERT INTO Birey
(
     TcNo,
     Adi,
     Soyadi,
     AnneAdi,
     BabaAdi,
     Cinsiyeti
)
VALUES
(
     '12332112332',
     'Ali',
     'Demir',
     'Özge',
     'Emre',
     'Erkek'
)


INSERT INTO Ogrenci_Bolum
(
     Birey_ID,
```

```
    Birim_ID,
    OgrenciNo,
    KayitTarihi,
    OgrenimYili
)
VALUES
(
    @@IDENTITY,    -- Veri tabanına en son eklenen Id
değeri
    5,             -- Kamu Yönetimi'nin Birim_ID'si
    '130303555',   -- Öğrenci No
    '2013-09-08',  -- Kayıt Tarihi
    1
)
```

Yukarıda Ali Demirin kaydı Birey tablosuna eklendi. Bu işlemin sonunda @@IDENTITY değişkeninde 52 değeri mevcuttur çünkü veri tabanında en son olarak Birey tablosuna veri eklenmiştir ve Birey tablosuna eklenen son ID değeri de 52 dir. Ogrenci_Bolum tablosuna veri eklerken de Birey_Id sütununa eklenecek değer olarak @@IDENTITY değişkeni kullanılmıştır.

IDENT_CURRENT() Fonksiyonu

IDENT_CURRENT() fonksiyonu parametre olarak tablo adı alır ve bu tabloya en son eklenen Id değerini döner.

Belirli bir tabloya eklenen en son ID değeri IDENT_CURRENT() fonksiyonu ile öğrenilebilir.

@@IDENTITY değişkeninin sağladığı kolaylığı yukarıda gördük. Fakat @@IDENTITY değişkeni her zaman istediğimiz sonucu vermeyebilir. Birey tablosunda meydana gelen değişikliklerin (ekleme, silme ve güncelleme) tetikleyiciler vasıtasıya Birey_Log tablosunda tutulduğunu varsayalım. Bu durumda Birey tablosuna bir kayıt eklendiğinde otomatik olarak Birey_Log tablosuna da bir kayıt eklenecektir. Dolayısı ile Birey tablosuna Ali Demir'in kaydı eklendikten hemen sonra Birey_Log tablosuna da bir kayıt eklenecektir. Bu durumda @@IDENTITY değişkeninin içerisinde 52 yani Birey tablosuna en son eklenen Id değeri değil, Birey_Log tablosuna eklenen en son ID değeri yer alacaktır. Fakat bize Birey_log tablosuna eklenen en son Id değeri değil, Birey tablosuna eklenen en son Id değeri lazımdır. İşte bu noktada IDENT_CURRENT() fonksiyonu kullanılabilir. IDENT_CURRENT('Birey') bize Birey tablosuna en son eklenen Id değerini dönecektir. Bu durumda bir önceki örneğimizi aşağıdaki gibi yeniden yazabiliriz:

```
INSERT INTO Birey
(
    TcNo,
    Adi,
    Soyadi,
    AnneAdi,
    BabaAdi,
    Cinsiyeti
)
VALUES
(
```

```
        '12332112332',
        'Ali',
        'Demir',
        'Özge',
        'Emre',
        'Erkek'
)


INSERT INTO Ogrenci_Bolum
(
        Birey_ID,
        Birim_ID,
        OgrenciNo,
        KayitTarihi,
        OgrenimYili
)
VALUES
(
        IDENT_CURRENT('Birey'), -- Birey tablosuna en son
eklenen ID değeri
        5,              -- Kamu Yönetimi'nin Birim_ID'si
        '130303555',    -- Öğrenci No
        '2013-09-08', -- Kayıt Tarihi
        1
)
```

Bireysel Etkinlik

- Birey tablosuna:
  - 'INSERT INTO ... VALUES' kalıbını kulanarak yeni kayıt ekleyiniz.
  - 'INSERT INTO ... SELECT' kalıbını kulanarak yeni kayıt ekleyiniz.
  - Bazı sütunların içeriği NULL kalacak şekilde ekleme yapınız.
  - UNION deyimini kullanarak aynı anda birden fazla kayıt ekleyiniz.
  - Aynı değerleri içeren select ifadelerini 'UNION' deyimi ile birleştirerek ekleme yapınız.
  - Aynı değerleri içeren select ifadelerini bu sefer 'UNION ALL' deyimi ile birleştirerek ekleme yapınız ve 'UNION' ile 'UNION ALL' deyimlerinin farkını gözlemleyiniz.
- Bir önceki adımda birey tablosuna eklediğiniz kişiler için Ogrenci_Bolum tablosuna kayıtlar ekleyiniz.
- Birey tablosunun bir kopyasını oluşturunuz ve Birey tablosundaki tüm verileri oluşturduğunuz kopya tabloya ekleyiniz.
- Bir önceki adımda yaptığınız işlemleri 'SELECT ... INTO ... FROM' kalıbı kullanarak tekrarlayınız.
- Birincil anahtar sütununa ekleme yapmaya çalışınız ve SQL Server'in vereceği hatayı görünüz.
- Birincil anahtar sütununa ekleme yapmaya izin veriniz ve bir önceki adımdaki işlemi yapınız işiniz bitince az önce verdiğiniz birincil anahtar sütununa ekleme iznini kaldırınız.
- Az önce oluşturduğunuz yedek Birey tablosuna yeni bir (hesaplatılmış) sütun ekleyiniz ve bu sütun için veri eklemeye çalışınız ve SQL Server'in vereceği hatayı gözlemleyiniz.

Özet

- Tabloya veri (satır) eklemek için 'INSERT INTO' komutu kullanılır.
- 'INSERT INTO' komutunun genel kullanımı "INSERT INTO Tablo_Adı (Sütun1, Sütun2, Sütun3, …) VALUES (Değer1, Değer2, Değer3, …)" şeklindedir.
- 'INSERT INTO' deyiminden sonra belirtilen sütun isimleri ile 'VALUES' deyiminden sonra yazılan değerler aynı sırada olmalıdır.
- 'INSERT INTO ... VALUES' kalıbında 'VALUES' yerine 'SELECT' deyimi de kullanılabilir.
- Veri girişi için değerler yazılırken karakter veya tarih veri tipindeki sütunların değerleri tek tırnak içerisinde yazılmalıdır.
- Veri girişi yaparken girilen verilerin tipi, tablodaki sütun tipleri ile aynı olmalıdır.
- Veri girişi yapacağımız tablonun (birincil anahtar olan sütunlar hariç) tüm sütunlarına birden veri girişi yapılacaksa eğer tablo adından sonra sütunlar belirtilmeden de ekleme işlemi yapılabilir.
- Tabloya yeni bir satır eklerken tüm sütunlar için veri eklemek zorunlu değildir.
- Tabloya aynı anda tek bir satır veya birden çok satır eklenebilir fakat aynı anda sadece tek bir tabloya ekleme yapılabilir.
- Tabloya aynı anda birden çok satır eklemek için 'UNION [ALL]' deyimi kullanılır.
- Tabloya eklenecek verileri kendimiz yazabileceğimiz gibi ekleme yapılacak olan tablo veya veri tabanındaki diğer tablolardaki, hatta sunucu üzerindeki başka bir veri tabanına ait olan tablolardaki verileri kullanarak da ekleme yapılabilir.
- Veri eklenirken 'INSERT INTO' deyiminden sonra belirtilmeyen sütunlara herhangi bir değer eklenmez, sütun içeriği NULL olur.
- Veri eklenirken ilgili sütun değeri olarak NULL yazılarak da tabloya NULL değer ekleme işlemi gerçekleştirilebilir.
- Tablo tasarımında NULL değer girilmesine izin verilmeyen sütunlar için mutlaka bir değer girilmelidir.
- Tablodaki birincil anahtar durumundaki sütun(lar)a veri girişi yapılamaz (bu kuralın istisnası "Birincil anahtar sütununa veri ekleme" başlığı altında belirtildi); bu sütun(lar)ın alacağı değer SQL Server tarafından otomatik olarak oluşturulur ve sütuna atanır.
- Birincil anahtar sütununa veri eklemeye çalışılması SQL Serverin hata döndürmesine sebep olur.
- Hesaplatılmış sütunlara veri eklenemez. Bu sütunların içeriği SQL Server tarafından otomatik olarak oluşturulur.

Özet (devamı)

- Yabancı anahtar sütunlarına girilen değerlerin referans tabloda bir karşılığının olması gerekir.
- Tablo tasarımında varsayılan (default) değer belirtilmiş bir sütun var ise ve yeni satır ekleme işleminde bu sütun için bir değer girilmemiş ise sütunda NULL değer değil, varsayılan değer oluşur.
- "SELECT ... INTO ... FROM" deyimi ile aynı anda bir tablo oluşturulup içerine veri eklenebilir. Bu kullanımda SELECT ifadesinden sonra gelen sütunları içeren bir tablo oluşturulur ve select sorgusunun sonucu bu tabloya eklenir.
- İstisnai bir durum olarak birincil anahtar sütununa veri ekleme ihtiyacı duyulduğunda geçici olarak tablonun birincil anahtar sütununa veri eklemeye izin verilebilir. Bunun için SET IDENTITY_INSERT '
- TabloAdı' ON komutu kullanılır. Ekleme işlemi bittikten sonra SET IDENTITY_INSERT 'TabloAdı' OFF komutu ile iznin kaldırılması unutulmamalıdır.
- @@IDENTITY bir sistem değişkenidir ve veri tabanına en son eklenen birincil anahtar değerini içinde barındırırır.
- IDENT_CURRENT() fonksiyonu parametre olarak tablo adı alır ve bu tabloya en son eklenen Id değerini döner.

DEĞERLENDİRME SORULARI

1. Tabloya veri eklemek için hangi komut kullanılır?
   a) SELECT
   b) INSERT
   c) UPDATE
   d) DELETE
   e) TRUNCATE

2. Bir sorgu içerisinde aynı anda kaç farklı tabloya veri eklenebilir?
   a) 1
   b) 2
   c) 256
   d) 1024
   e) Sınırsız

3. Tabloya değerleri kendimiz yazarak veri ekleme işleminde birden fazla satırın aynı anda eklenebilmesi için hangi komut kullanılır?
   a) INNER JOIN
   b) LEFT JOIN
   c) RIGHT JOIN
   d) CROSS JOIN
   e) UNION

```
... ... Tablo1
...
  Sütun1,
  Sütun2,
  Sütun3
...
  Tablo2
```

4. Bir tabloya, başka bir tablodaki verileri kullanarak ekleme işlemi yapan yukarıdaki kod içerisinde boş bırakılan yerlere sırası ile seçeneklerden hangisi gelmelidir?
   a) SELECT – INTO – INSERT – FROM
   b) SELECT – INSERT – INTO – FROM
   c) INSERT – INTO – FROM – SELECT
   d) INSERT – INTO – SELECT – FROM
   e) INSERT – SELECT – INTO – FROM

```
INSERT INTO Ogrenci_Bolum
(
      Birey_ID,
      Birim_ID,
      OgrenciNo,
      OgrenimYili
)
VALUES (52, 5, '130303555', 1)
```

5. Yukarıdaki kod ile Ogrenci_Bolum tablosuna bir satır eklendikten sonra Ogrenci_Bolum tablosunun birincil anahtar sütunu olan ID sütununun içeriği ne olur?
   a) NULL değer yazılır.
   b) Rasgele bir değer yazılır.
   c) Herhangi bir değer yazılmaz.
   d) Sıradaki benzersiz değer yazılır.
   e) ID sütununa en son verilen değer yazılır.

```
INSERT INTO Ogrenci_Bolum
(
      Birey_ID,
      Birim_ID,
      OgrenciNo,
      OgrenimYili
)
SELECT 52, 5, '130303553',  1
UNION
SELECT 53, 6, '130303553',  1
UNION
SELECT 53, 7, '130303553',  1
```

6. Yukarıdaki kod ile Ogrenci_Bolum tablosuna kaç satır eklenir?
   a) 0
   b) 1
   c) 2
   d) 3
   e) 4

I. Hesaplatılmış sütuna veri eklenemez.

II. Varsayılan değere sahip sütuna veri eklenemez.

III. NULL veri kabul etmeyen sütun için değer yazılması zorunludur.

7. Tabloya satır ekleme ile ilgili olarak aşağıdakilerin hangisi ya da hangileri yanlıştır?
   a) Yalnız I
   b) Yalnız II
   c) Yalnız III
   d) I ve II
   e) II ve III

```
...
      Sütun1,
      Sütun2
...
      Tablo2
...
      Tablo1
```

8. Başka bir tablodaki verileri kullanarak ekleme işlemini, eklenecek olan tabloyu o anda oluşturarak yapan yukarıdaki kod içerisinde boş bırakılan yerlere sırası ile aşağıdakilerden hangisi gelmelidir?
   a) SELECT – INTO – FROM
   b) INSERT – INTO – FROM
   c) INSERT – SELECT – FROM
   d) SELECT – INSERT – FROM
   e) INSERT – INTO – SELECT

9. '`SET IDENTITY_INSERT Birey ON`' komutu Birey tablosu için ne yapar?
   a) Tabloya veri eklenmesini engeller.
   b) Tabloya veri eklenmesine izin verir.
   c) Tablo için birincil anahtar sütununa veri eklenmesine izin verir.
   d) Tablo için yabancı anahtar sütununa veri eklenmesine izin verir.
   e) Tablo için hesaplatılmış sütununa veri eklenmesine izin verir.

   I. 'INSERT INTO ... VALUES' kalıbında 'VALUES' yerine 'SELECT' deyimi kullanılabilir.
   II. Yabancı anahtar sütunlarına girilen değerlerin referans tabloda bir karşılığının olması gerekir.
   III. Tabloya yeni bir satır eklerken tüm sütunlar için veri eklemek zorunludur.

10. Tabloya veri ekleme ile ilgili olarak yukarıdakilerden hangisi ya da hangileri doğrudur?
    a) Yalnız I
    b) I ve II
    c) I ve III
    d) II ve III
    e) I, II ve III

Cevap Anahtarı

1.b, 2.a, 3.e, 4.d, 5.d, 6.d, 7.b, 8.a, 9.c. 10.b

YARARLANILAN KAYNAKLAR

Ekeleme Örnekleri (Transact-SQL) 03.06.2019 tarihinde http://technet.microsoft.com/tr-tr/library/dd776381%28v=sql.105%29.aspx adresinden erişildi.

Gözüdeli, Y. (2012). Yazılımcılar için SQL Server 2008 R2 & Veri tabanı Programlama, 5. Baskı, Ankara: Seçkin Yayıncılık

INSERT (Transact-SQL) 02.06.2019 tarihinde http://technet.microsoft.com/en-us/library/ms174335%28v=sql.105%29.aspx adresinden erişildi.

INSERT (Transact-SQL) 02.06.2019 tarihinde http://technet.microsoft.com/tr-tr/library/ms174335%28v=sql.105%29.aspx adresinden erişildi.

INSERT Examples (Transact-SQL) 03.06.2019 tarihinde http://technet.microsoft.com/en-us/library/dd776381%28v=sql.105%29.aspx adresinden erişildi.

Özhan, C. (2013). İleri Seviye T-SQL Programlama, 1. Baskı, İstanbul: KODLAB Yayınevi

Petkovic, D. (2006). Microsoft SQL Server, 1. Baskı, İstanbul: Alfa Yayınları

İLİŞKİLİ TABLOLAR İLE SORGU HAZIRLAMAK

İÇİNDEKİLER

- Tablodaki Verileri Güncelleme
- UPDATE Deyimi
  - Bir Başka Tablodan Alınan Verilerle Güncelleme Yapma
- Tablolardan Veri Silme
  - DELETE Deyimi Yapısı
  - Bir Başka Tablodan Okunan Verileri Kullanarak Silme

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Tablolardaki verileri güncelleyebilecek,
  - Bir başka tablodaki verileri kullanarak güncelleme yapabilecek,
  - Tablolardaki verileri silebilecek,
  - Bir başka tablodaki verileri kullanarak silme işlemi yapabilecek,
  - TRUNCATE komutunu kullanmayı öğrenebileceksiniz.

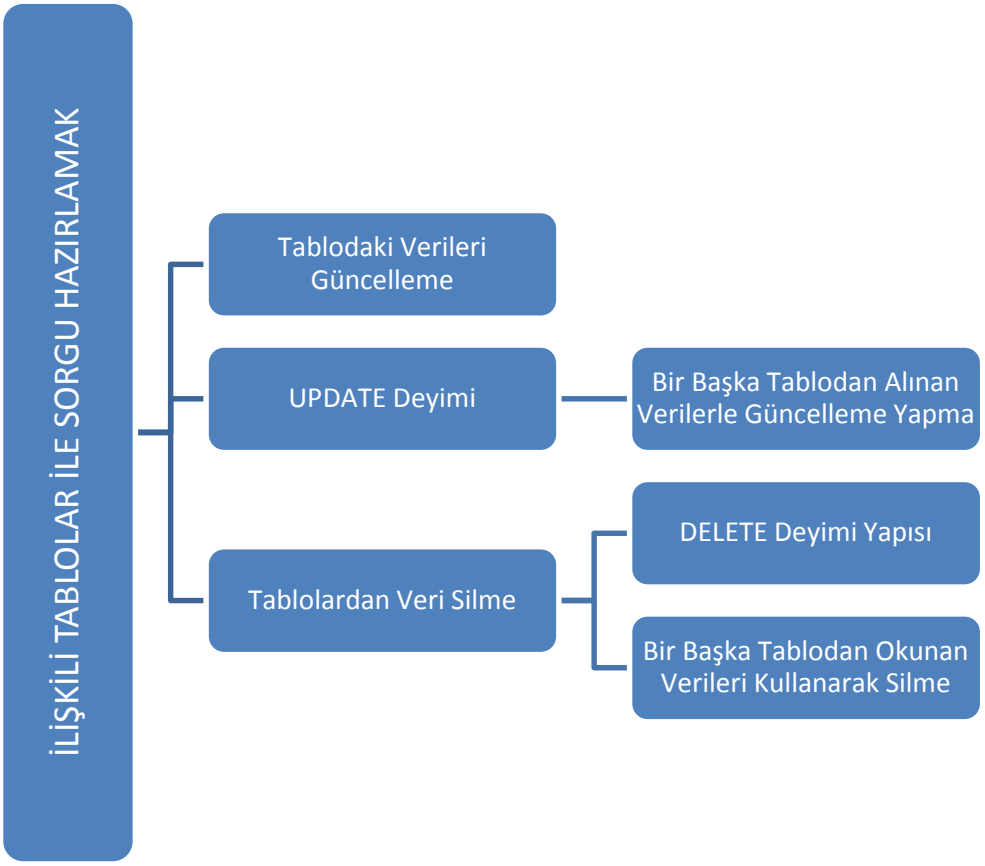
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Arş. Gör. Yakup BAYOĞLU

ÜNİTE

10


İLİŞKİLİ TABLOLAR İLE SORGU HAZIRLAMAK
Tablodaki Verileri Güncelleme
UPDATE Deyimi
Bir Başka Tablodan Alınan Verilerle Güncelleme Yapma
Tablolardan Veri Silme
DELETE Deyimi Yapısı
Bir Başka Tablodan Okunan Verileri Kullanarak Silme

GİRİŞ

Bir önceki ünitede tablolara nasıl veri girileceği işlenmişti. Tabloya hatalı veri girildiği veya fazladan veri eklendiği zaman ne yapılabilir? Fazladan girilmiş olan veri silinebilir, aynı şekilde hatalı girilmiş olan veri(ler) de silinip yeniden girilebilir veya değiştirilebilir (güncellenebilir). Hatalı veriyi silip yeniden eklemek yerine güncellemek daha doğru bir yaklaşımdır.

Tablodaki verilerin güncellenmesi için UPDATE komutu, silinmesi için de DELETE kullanılır.

UPDATE ve DELETE komutlarında where koşul cümleciğinin doğru ve eksiksiz olması hayati önem taşımaktadır.

UPDATE ve DELETE ile başlayan komutlar çalıştırılmadan önce aynı koşul cümleciklerini içeren select sorgusunun çalıştırılması, güncellenecek kayıtların güncellenmeden önce yeniden gözden geçirilmesine olanak sağlayacağından hata yapılmasını (yanlış kayıtların ya da istenenden daha fazla kaydın güncellenmesini/silinmesini vs.) önlemek adına alınabilecek iyi bir tedbir olur.

Bir başka tablodan okunan verileri kullanarak güncelleme ve silme işlemi yapmak mümkündür, bu durumda ilgili tablolar join ifadeleri ile birleştirilerek işlem yapılır.

Bir tablodan satır silerken silinecek satırdaki kayıtları referans alan başka bir tabloda kayıtlar varsa silme işleminden sonra veri bütünlüğü bozulma ihtimali vardır. Bu yüzden silinecek satırların bağlantılarını kontrol ederek silinmesi gerekir. Birbiri ile ilişkili tablolar yabancı anahtar ile birbirlerine bağlı ise bu kontrolü SQL Server'in kendisi yapmaktadır.

Bu ünitede tablodaki verilerin nasıl güncelleneceğini / silineceğini, tablodaki sütun(lar)ı veya başka bir tablodaki sütunları kullanarak da güncellenebileceğini / silinebileceğini, son olarak TRUNCATE komutunu ve DELETE komutundan farkını göreceğiz.

Tablodaki verileri güncellemek için UPDATE komutu kullanılır.

TABLODAKİ VERİLERİ GÜNCELLEME

UPDATE Deyimi

Tablodaki verilerin güncellenmesi için UPDATE komutu kullanılır, genel kullanımı:

```
UPDATE TabloAdi
SET
(     SütunAdi1 = Deger1
      [,SütunAdi2 = Deger2 ]
      [,SütunAdi3 = Deger3, ..]
)
[WHERE Koşul]
```

şeklindedir. (Özhan, 2013)

Örnek veri tabanındaki Birey tablosunun ilk 10 kaydına bakalım:

```
SELECT TOP 10
      *
FROM
      Birey
ORDER BY
      ID
```

Results | Messages

	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti
1	2	12312312312	Hidayet	Çölkusu	Ayse	Yusuf	Erkek
2	3	35462349764	Taha	Bayram	Fatma	Ali	Erkek
3	4	79854631320	Ayse	Yildiz	Hayriye	Cihan	Kiz
4	5	54678431348	Nazli	Yasar	Neslihan	Enver	Kiz
5	6	65874110244	Erkan	Toprak	Hatice	Cihad	Erkek
6	7	78984654164	AHMET	Akan	Özlem	Samet	Erkek
7	8	98971354212	Mustafa	Toroman	Irem	Yusuf	Erkek
8	9	38312472124	Selmani	Hatipoglu	Meltem	Musa	Erkek
9	10	78974564762	Samet	Öztürk	Yasemin	Murat	Erkek
10	11	74547984626	Nazife	Kibar	Aynur	Burak	Kiz

Şekil 10.1. Birey Tablosundaki İlk On Kayıt

ID numarası 7 olan kayıtta isim sütunu diğer kayıtlardan farklı olarak kelimenin tamamı büyük harf ile yazılmış. Bütünlük olması açısından bu kaydı diğerleri gibi sadece ilk harfi büyük kalacak şekilde güncellemek isteyebiliriz. Bunun için UPDATE komutunu kullanmamız gerekecek:

```
UPDATE Birey
SET
      Adi = 'Ahmet'
WHERE
      ID = 7
```

Yukarıdaki kod ID numarası 7 olan kayıttaki “Adi” sütununu “Ahmet” diye günceller. “WHERE ID = 7” koşul cümleciği olmasa idi tüm kayıtların “Adi” sütunları “Ahmet” olarak güncellenecekti.

Not: update ve ileride göreceğimiz delete komutlarında where koşul cümleciğinin doğru ve eksiksiz olması hayati önem taşımaktadır.

UPDATE ile başlayan komutlar çalıştırılmadan önce aynı koşul cümleciklerini içeren select sorgusunun çalıştırılması, güncellenecek kayıtların güncellenmeden önce yeniden gözden geçirilmesine olanak sağlayacağından hata yapılmasını (yanlış kayıtların ya da istenenden daha fazla kaydın güncellenmesini vs.) önlemek adına alınabilecek iyi bir tedbir olur.

Update ve ileride göreceğimiz delete komutlarında where koşul cümleciğinin doğru ve eksiksiz olması hayati önem taşımaktadır.

ID numarası 11 olan kayıttaki Nazife Kibar’ın evlilik sebebi ile soyadının “Duru” olarak değiştiğini varsayalım. Bu durumda yazacağımız sorgu:

```
UPDATE Birey
SET
      Soyadi = 'Duru'
```

```
WHERE
      ID = 11
```

şeklinde olacaktır.

İlerleyen aşamalarda veri tabanında yabancı uyruklu insanların da verilerini tutma ihtimaline binaen Birey tablosuna Uyruk bilgisini tutmak üzere bir sütun ekleyelim:

```
ALTER TABLE [Birey]
ADD Uyruk VARCHAR(100)
```

Uyruk sütununu ekledik ama şuanda tablodaki tüm kayıtlarda bu veri boş (NULL). Tablodaki tüm kayıtların Uyruk sütununu "T.C." olacak şekilde güncelleyelim:

```
UPDATE Birey
SET
      Uyruk = 'T.C.'
```

Burada koşul cümleciği kullanmadık çünkü yaptığımız güncellemenin tüm kayıtları etkilemesini istedik.

Tablodaki herhangi bir sütunun hatta güncellenecek olan sütunun içerisindeki veri kullanılarak da güncelleme yapılabilir.

Yukarıdaki örneklerde güncelleme yaparken sabit veri kullanarak ("Ahmet", "Duru", "T.C.") güncelleme yaptık, şimdi hâlihazırda tabloda bulunan verileri kullanarak güncelleme yapalım. Tablodaki Soyadı sütunundaki verileri kelimenin tamamı büyük harf olacak şekilde güncellemek istersek yazacağımız kod:

```
UPDATE Birey
SET
      Soyadi = UPPER(Soyadi)
```

şeklinde olacaktır. Görüldüğü gibi dışarıdan veri almadan tablodaki veriyi, hatta güncellenecek sütunun kendi içerisindeki veriyi kullanarak güncelleme yapmış olduk. (UPPER() fonksiyonu için 11. üniteye bakabilirsiniz.)

```
SELECT TOP 10
      'Adı Soyadı' = Adi + ' ' + Soyadi
FROM
      Birey
ORDER BY
      ID
```

Results | Messages

	Adı Soyadı
1	Hidayet ÇOLKUSU
2	Taha BAYRAM
3	Ayse YILDIZ
4	Nazli YASAR
5	Erkan TOPRAK
6	Ahmet AKAN
7	Mustafa TOROMAN
8	Selmani HATIPOGLU
9	Samet ÖZTÜRK
10	Nazife DURU

Şekil 10.2. Birey Tablosundaki İlk 10 Adı Soyadı Verisi

```
ALTER TABLE [Birey]
ADD AdiSoyadi VARCHAR(100)
```

Komutu ile tabloya 'AdiSoyadi' isimli bir sütun eklemiş olduk fakat şu anda bu sütun tüm tabloda içi boş (NULL) durumundadır. Şimdi AdiSoyadi sütununun içini tablodaki Adi ve Soyadi sütunlarını kullanarak dolduralım.

```
UPDATE Birey
SET
      AdiSoyadi = Adi + ' ' + Soyadi
```

UPDATE deyimi ile aynı anda birden fazla sütunun verileri güncellenebilir.

Tablomuzun son hâli aşağıdaki gibidir:

Results | Messages

	ID	TcNo	Adi	Soyadi	AnneAdi	BabaAdi	Cinsiyeti	Uyruk	AdiSoyadi
1	2	12312312312	Hidayet	ÇÖLKUSU	Ayse	Yusuf	Erkek	T.C.	Hidayet ÇÖLKUSU
2	3	35462349764	Taha	BAYRAM	Fatma	Ali	Erkek	T.C.	Taha BAYRAM
3	4	79854631320	Ayse	YILDIZ	Hayriye	Cihan	Kiz	T.C.	Ayse YILDIZ
4	5	54678431348	Nazli	YASAR	Neslihan	Enver	Kiz	T.C.	Nazli YASAR
5	6	65874110244	Erkan	TOPRAK	Hatice	Cihad	Erkek	T.C.	Erkan TOPRAK
6	7	78984654164	Ahmet	AKAN	Özlem	Samet	Erkek	T.C.	Ahmet AKAN
7	8	98971354212	Mustafa	TOROMAN	Irem	Yusuf	Erkek	T.C.	Mustafa TOROMAN
8	9	38312472124	Selmani	HATIPOGLU	Meltem	Musa	Erkek	T.C.	Selmani HATIPOGLU
9	10	78974564762	Samet	ÖZTÜRK	Yasemin	Murat	Erkek	T.C.	Samet ÖZTÜRK
10	11	74547984626	Nazife	DURU	Aynur	Burak	Kiz	T.C.	Nazife DURU

Şekil 10.3. Birey Tablosunun Son Hâli

Not: UPDATE deyimi ile aynı anda sadece bir tablonun verileri güncellenebilir. (Petkovic, 2006)

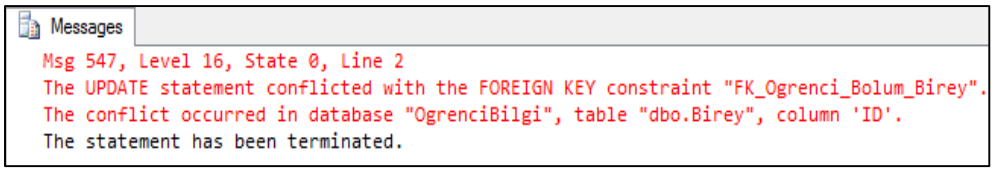
Not: Değeri SQL Server tarafından oluşturulan birincil anahtar sütunu güncellenemez. Bunun haricindeki birincil anahtar sütunları güncellenebilir fakat tekrarlı veri oluşmaması için güncellerken kullanılan yeni değerin hâlihazırda tabloda var olmaması gerekmektedir.

İpucu: UPDATE deyimi ile aynı anda birden fazla sütunun verileri güncellenebilir.

İpucu: UPDATE deyimi ile yabancı anahtar olan bir sütunun içeriği güncellenirken yeni eklenen değerin, referans tablosunda bir karşılığı olduğundan emin olunmalıdır. Aksi takdirde SQL Server hata verecektir. Örnek olarak Ogrenci_Bolum tablosunda Birey_ID sütununu, Birey tablosunda karşılığı olmayan bir ID değeri ile güncellemeye çalışalım. Bu durumda SQL Server'in vereceği hata Şekil 10.4.te gösterilmiştir.

Yabancı anahtar sütunu güncellenirken verilen yeni değerin, referans tabloda bir karşılığının olması gerekir.

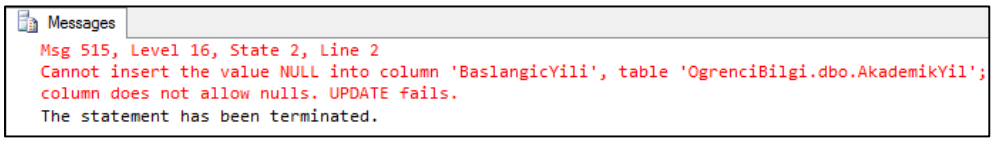
```
update Ogrenci_Bolum
set
      Birey_ID = 525
where
      ID = 21
```



Şekil 10.4. Yabancı Anahtar Bir Sütunun Referans Tabloda Karşılığı Olmayan Bir Değer İle Güncellenmeye Çalışıldığında SQL Server'in Verdiği Hata Uyarısı

Tablo tanımlanırken NULL olmasına izin verilmeyen bir sütunu güncellerken mutlaka bir değer girilmelidir. Bu şekildeki bir sütun NULL değer ile güncellemek istenirse SQL Server Şekil 10.5.te gösterilen hata uyarısını verir:

```
update AkademikYil
set
      BaslangicYili = null
```



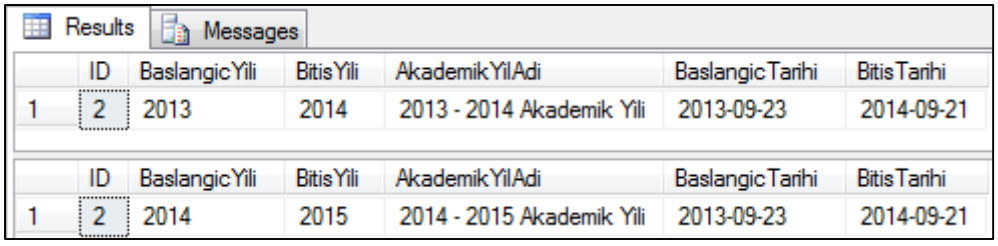
Şekil 10.5. Tablo Tanımı Gereği NULL Olmasına İzin Verilmeyen Bir Sütunu NULL Değer İle Güncellemeye Çalışıldığında SQL Server'in Verdiği Hata Uyarısı

Hesaplatılmış bir sütunun içeriği güncellenemez.

Dikkat edilmesi gereken bir diğer husus *hesaplatılmış* bir sütunun içeriğinin güncellenemeyeceğidir. Hesaplatılmış sütunun içeriğini belirleyen sütunları güncellemek sureti ile istenen gerçekleştirilebilir ama direk hesaplatılmış sütunu güncellemeye çalışmak SQL Server'in hata vermesine sebep olur. Bir önceki ünitede oluşturduğumuz Akademikyil tablosunda BitisYili ve AkademikYilAdi sütunları BaslangicYili sütunundan türetilmekte idi. BaslangicYili sütunu güncellendiği zaman otomatik olarak BitisYili ve AkademikYilAdi sütunları da güncellenecektir.

```
select * from AkademikYil

update AkademikYil
set
      BaslangicYili = 2014
Select * from AkademikYil
```

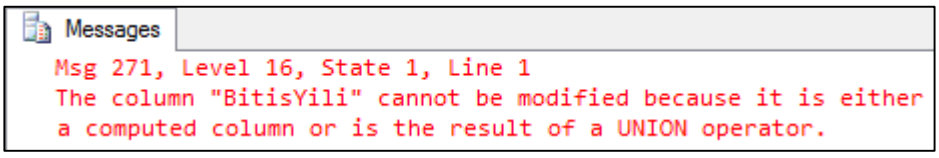
Yukarıda AkademikYil tablosunun başlangıçtaki hâlini ve BaslangicYili sütununun güncellenmesinden sonraki hâlini sorguladık:



Şekil 10.6. AkademikYil Tablosunun İlk Hâli ve Hesaplatılmış Bir Sütun Olan BaslangicYili Sütunu Güncellendikten Sonraki Hâli

BitisYili veya AkademikYilAdi sütununu güncellemeye çalışmak hata ile sonuçlanacaktır, hata metni Şekil 10.6.da gösterilmiştir.

```
update AkademikYil
set
      BitisYili = 2015
```



Şekil 10.7. Hesaplatılmış Bir Sütun Güncellemeye Çalışıldığında SQL Server'in Verdiği Hata Uyarısı

Bir Başka Tablodan Alınan Verilerle Güncelleme Yapma

Bazen bir tablodaki sütunu(ları) başka bir tabloda bulunan verilerle güncellememiz gerekebilir. Bu durumda join ifadesi ile tablolar birleştirilerek güncelleme işlemi yapılır:

```
UPDATE TabloAdi1
SET
      Sütun1 = T2.SütunX
FROM
      TabloAdi1 T1
      INNER JOIN TabloAdi2 T2
      ON Birleşme koşulları
```

Yukarıda Birey tablosuna AdiSoyadi isimli bir sütun eklemiştik, aynı şekilde bir de OgrenciNo sütunu eklemiş olduğumuzu varsayalım. Birey tablosundaki OgrenciNo sütununu, Ogrenci_Bolum tablosunda yer alan OgrenciNo sütununda bulunan veriler ile güncellemek için yazmamız gereken kod:

```
UPDATE Birey
SET
      OgrenciNo = OB.OgrenciNo
FROM
      Birey B
      INNER JOIN Ogrenci_Bolum OB
      ON OB.Birey_ID = B.ID
```

Şeklinde olacaktır. Benzer şekilde, güncelleme işleminde, aynı sunucuda yer alan başka bir veri tabanındaki tablolardan da veri alınabilir.

TABLOLARDAN VERİ SİLME

Tablodan veri silmek için DELETE komutu kullanılır.

DELETE Deyimi

Daha önce tabloya nasıl veri eklenileceğini görmüştük; bu bölümde tablodan nasıl veri silineceğini göreceğiz. Tablodan veri silmek için DELETE komutu kullanılır ve genel kullanımı:

```
DELETE FROM tabloAdi
```

Şeklindedir. Delete komutu ile artık ihtiyaç duyulmayan kayıtları, yanlışlıkla eklenmiş kayıtları vs. silebiliriz.

```
DELETE FROM Birey
```

Sorgusu Birey tablosundaki tüm kayıtları siler. Eğer tüm kayıtların silinmesi gerekmiyorsa koşul cümlecikleri sorguya dâhil edilip sadece silinmesi istenen kayıtların sorgudan etkilenmesi sağlanmalıdır.

Not: UPDATE komutunda olduğu gibi DELETE komutunda da koşul cümlecikleri hayati önem arz etmektedir.

DELETE ile başlayan komutlar çalıştırılmadan önce aynı koşul cümleciklerini içeren select sorgusunun çalıştırılması, silinecek kayıtların silinmeden önce yeniden gözden geçirilmesine olanak sağlayacağından hata yapılmasını (yanlış kayıtların ya da istenenden daha fazla kaydın silinmesini vs.) önlemek adına alınabilecek iyi bir tedbir olur.

DELETE komutundan tüm satır etkilenir. Sadece bir(kaç) sütunun verisini silmek gibi bir durum söz konusu değildir.

```
DELETE FROM Birey
WHERE
      ID = 3
```

Yukarıdaki kod Birey tablosundaki ID numarası 3 olan kaydın silinmesini sağlar.

Not: Delete komutundan tüm satır etkilenir. Sadece bir(kaç) sütunun verisini silmek gibi bir durum söz konusu değildir. Tüm satırı değil de bazı sütunları silmemiz gerekiyor ise bunu silmek istediğimiz sütunları NULL veyahut'' gibi boş değerler kullanarak update komutu ile güncellemek sureti ile gerçekleyebiliriz. Eğer silmek istenen sütunlar tablonun tamamı için silinmek isteniyorsa (where koşul cümleciği kullanılmadan silinmek isteniyorsa) ve sütunlar bir daha kullanılmayacaksa 'alter table ... drop column ...' komutu ile istenen sütunlar tablodan tamamen silinebilir.

```
DELETE FROM Birey
WHERE
      Adi = 'Taha'
```

Yukarıdaki kod adı "Taha" olan tüm kayıtların silinmesini sağlar.

Not: DELETE komutu ile aynı anda sadece bir tablodan veri silinebilir.

Bir Başka Tablodan Okunan Verileri Kullanarak Silme

Bazen bir tablodan veri silineceği zaman, başka tablo(lar)daki veriler kullanılarak silinmesi gerekebilir. Bu durumda join ifadesi ile tablolar birleştirilerek silme işlemi yapılır:

```
DELETE FROM TabloAdi1
FROM
      TabloAdi1
      INNER JOIN TabloAdi2
      ON Birleştirme koşulları
[WHERE Koşul]
```

Birey tablosundan, kayıt tarihi 2005 yılından daha eski olan öğrencilerin kayıtlarını silelim:

```
DELETE FROM Birey
FROM
      Birey B
      INNER JOIN Ogrenci_Bolum OB
      ON Ob.Birey_ID = B.ID
WHERE
      YEAR(OB.KayitTarihi) < 2005
```

Hatırlatma: Yukarıdaki sorgu sonucunda sadece Birey tablosundan kayıt(lar) silinmiş olur; Ogrenci_Bolum tablosundan herhangi bir kayıt silinmez.

Ogrenci_Bolum tablosundaki KayitTarihi sütunundaki veriler kullanılarak Birey tablosundan kayıt silinmesini sağlar.

Silme işlemi yaparken silinecek satırların bağlantılarının kontrol edilerek silinmesi veri bütünlüğü açısından önemlidir.

Bir önceki örnekte yer alan silme işlemi tablo birleştirme yerine alt sorgu kullanılarak da gerçekleştirilebilir:

```
DELETE FROM Birey
WHERE
      ID in (SELECT
                  Birey_ID
            FROM
                  Ogrenci_Bolum
            WHERE
                  YEAR(KayitTarihi) < 2005)
```

Veri bütünlüğü

Tablodan satır silerken veri tabanındaki bütünlüğü bozmamamız gerekir, dolayısı ile bir tablodan satır silerken silinecek satırı referans alan başka bir tablo varsa silme işleminden sonra veri bütünlüğü bozulacaktır. Bu yüzden silinecek satırların bağlantılarını kontrol ederek silmemiz gerekir. Birbiri ile ilişkili tablolar yabancı anahtar ile birbirlerine bağlı ise bu kontrolü SQL Server'in kendisi zaten yapmaktadır. Ogrenci_Bolum tablosundaki Birey_ID sütunu Birey tablosundaki ID sütununa yabancı anahtar olduğu için Birey tablosundan bir satır silinirken başka tablolarda silinecek satıra bağlı veri var mı diye bakılıp ona göre silme işlemi gerçekleşir veya gerçekleşmez.

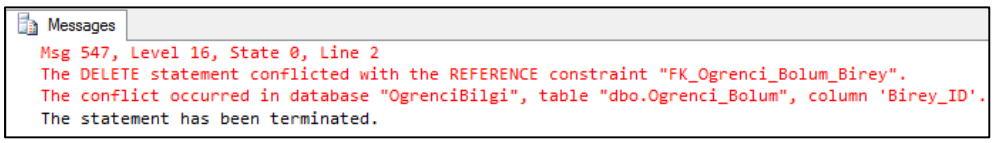
Results | Messages

	ID	Birey_ID	Birim_ID	OgrenciNo	KayitTarihi	OgrenimYili
1	1	2	1	120357032	2012-09-01	2
2	2	3	12	120358965	2010-09-01	3
3	3	10	1	120357001	2013-09-01	1
4	4	4	3	120358985	2000-09-01	4
5	5	5	3	120358985	2000-09-01	4

Şekil 10.8. Ogrenci_Bolum Tablosundaki İlk Beş Kayıt

Birey tablosundan ID değeri 5 olan satırı silmek istersek Ogrenci_Bolum tablosunda Birey_Id değeri 5 olan bir satır bulunduğu için SQL Server bu silme işlemine izin vermeyecektir:

```
DELETE FROM Birey
WHERE ID = 5
```



Şekil 10.9. Başka Bir Tabloda Kendisine Yabancı Anahtar ile Bağlı Bir Kayıt Bulunan Bir Satır Silinmek İstendiğinde SQL Server'in Verdiği Hata Uyarısı

DELETE komutunda TOP deyiminin kullanımı

Tablodan veri silineceği zaman select komutunda olduğu gibi TOP deyimi kullanılabilir.

```
DELETE TOP 5
FROM Birey
```

Yukarıdaki kod Birey tablosundan rastgele 5 satırı siler. TOP deyimi 'ORDER BY' deyimi ile beraber kullanıldığı zaman efektif bir şekilde kullanılmış olur fakat tablodan satır silme işleminde 'ORDER BY' deyimi kullanılmaz. Peki, silme işleminde TOP deyiminin efektif bir şekilde kullanımı nasıl olur? Cevap: alt sorgu içerisinde TOP ve 'ORDER BY' deyimlerini kullanarak. Örnek olarak Birey tablosundaki en eski 5 satırı silmek istediğimizi varsayalım:

```
DELETE FROM Birey
WHERE
     ID IN (SELECT TOP 5
                ID
           FROM
                Birey
           ORDER BY
                ID
           )
```

TRUNCATE deyimi

TRUNCATE deyimi koşul cümleciği olmadan yazılan DELETE komutu ile aynı silme işlemini yapar.

TRUNCATE deyimi koşul cümleciği olmadan yazılan DELETE komutu ile aynı silme işlemini yapar. (Gözüdeli, 2012)

```
TRUNCATE TABLE Birey
```

Yukarıdaki kod Birey tablosundaki tüm kayıtları siler.

TRUNCATE komutu, bir tablodaki tüm verilerin silinmek istendiği fakat bu silme işlemi için satır bazlı transaction log kayıtlarının oluşturulmasının istenmediği durumlarda kullanılır. Dolayısı ile TRUNCATE komutu kullanıldıktan sonra ROLLBACK komutu ile işlem geri alınamaz (Çünkü TRUNCATE komutu sonrası transaction log kaydı tutulmaz.).

Bireysel Etkinlik

- Transaction kavramını ve nasıl kullanıldığını öğreniniz. Bunun için Referans kısmındaki 'Yazılımcılar için SQL Server 2008 R2 & Veritabanı Programlama' kitabındaki 20. üniteye ya da 'http://www.csharpnedir.com/articles/read/?id=375' adresine veyahut da web arama motorlarına başvurabilirsiniz.

Bir tablodan herhangi bir kayıt silindikten sonra silinen kaydın birincil anahtar sütunundaki değer artık boşta olmasına rağmen yeniden kullanılmaz.

Not: TRUNCATE komutunda istense de WHERE deyimi kullanılamaz. Koşul belirtilecekse DELETE komutu kullanılmalıdır.

Bir tablodan herhangi bir kayıt silindikten sonra silinen kaydın birincil anahtar sütunundaki değer artık boşta olmasına rağmen yeniden kullanılmaz. İlk silme örneğinde Birey tablosundan ID numarası 3 olan kayıt silinmişti. Bu silme işleminden sonra Birey tablosuna yeni kayıt eklenirse eğer 3 ID değeri tabloda var olmamasına rağmen kullanılmaz, en son ekleme işleminde (veri tabanının kendisi tarafından) verilmiş ID değerinden sonra gelmesi gereken değer ne ise otomatik olarak o değer ID sütununa verilir. Birey tablosundan koşul cümleciği yazmadan silme işlemi yapıldığında tabloda kayıt kalmaz ama tabloya ekleme yapıldığında ID sütununun alacağı değer, en son verilmiş ID değerinden devam eder.

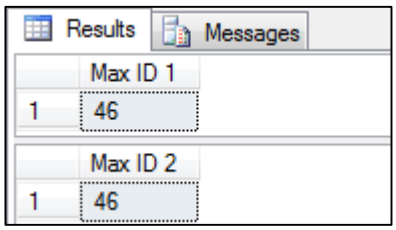
```
SELECT
      MAX(ID)
FROM
      Birey
```

Sorgusu tablodaki en yüksek ID değerini (52) döndürür. Birey tablosundaki tüm kayıtlar silindikten sonra ekleme yapılırsa ID sütunu 53 den devam eder. Bu durum silme işleminin DELETE komutu ile yapılması halinde geçerlidir. Tablodaki tüm veriler TRUNCATE deyimi ile silinirse eğer tabloya yeniden ekleme yapıldığında birincil anahtar sütununa (ID) başlangıç değerinden itibaren değer verilir (yeniden 1'den başlar). Bunu bir örnek üzerinden görmek için Birey tablosunun iki adet kopyasını oluşturalım, her iki tablodaki tüm verileri silelim fakat birinde DELETE komutunu, diğerinde TRUNCATE komutunu kullanalım sonra her iki tabloya da yeniden birer satır kayıt ekleyelim ve son olarak ID sütunlarının aldığı değerleri görelim:

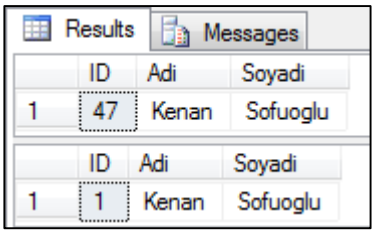
```
CREATE TABLE BireyKopya1
(
     ID    INT IDENTITY,
     Adi   VARCHAR(50),
     Soyadi     VARCHAR(50)
)
CREATE TABLE BireyKopya2
(
     ID    INT IDENTITY,
     Adi   VARCHAR(50),
```

```
      Soyadi     VARCHAR(50)
)

INSERT INTO BireyKopya1
SELECT
      Adi,
      Soyadi
FROM
      Birey

INSERT INTO BireyKopya2
SELECT
      Adi,
      Soyadi
FROM
      Birey

SELECT 'Max ID 1' = MAX(ID) FROM BireyKopya1
SELECT 'Max ID 2' = MAX(ID) FROM BireyKopya2

DELETE FROM BireyKopya1
TRUNCATE TABLE BireyKopya2

INSERT INTO BireyKopya1
SELECT
      'Kenan',
      'Sofuoğlu'

INSERT INTO BireyKopya2
SELECT
      'Kenan',
      'Sofuoğlu'

SELECT * FROM BireyKopya1
SELECT * FROM BireyKopya2

DROP TABLE BireyKopya1
DROP TABLE BireyKopya2
```

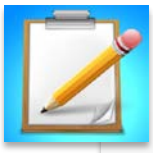
Yukarıda önce Bireykopya1 ve BireyKopya2 isimli iki tane Birey tablosunun kopyası olacak yeni tablolar oluşturduk. Her iki tabloya da orijinal Birey tablosundan ekleme yaptık ve en yüksek ID değerlerini sorguladık. Akabinde Bireykopya1 tablosundaki tüm verileri Delete komutu ile BireyKopya2 tablosundaki verileri ise Truncate komutu ile sildik. Sonra her iki tabloya da yeniden birer satır veri ekledik ve bu satırlardaki ID değerlerinin farklı olduğunu gördük:



Şekil 10.10. Veriler Silinmeden Önce Her İki Tablodaki Maksimum ID Değerleri



Şekil 10.11. Tablolara Yeniden Birer Satır Veri Eklendikten Sonraki ID Değerleri



Bireysel Etkinlik

- Elinizdeki veritabanının bir yedeğini alınız ve Birey tablosunda aşağıda belirtildiği şekilde çeşitli update komutları gerçekleyiniz:
  - Elle değer girerek,
  - Aynı tablodaki sütun(lar)ı kullanarak,
  - Farklı tablolardaki sütun(lar)ı kullanarak,
  - WHERE koşul cümlecik(ler)i kullanarak ve kullanmayarak.
- WHERE koşul cümleciği kullanarak ve kullanmadan ayrı ayrı DELETE komutu ile silme işlemi yapınız.
- Başka bir tabloda yer alan sütunları kullanarak silme işlemi yapınız.
- DELETE ve TRUNCATE komutlarının farkını pekiştirmek için (birincil anahtar içeren) bir tablodaki tüm verileri siliniz ve yeniden veri ekleyiniz ve birincil anahtar sütununun aldığı değerleri gözlemleyiniz. Bu işlemi önce DELETE komutunun kullanarak daha sonra da TRUNCATE komutunu kullanarak yapınız.
- Az önce aldığınız yedeği geri yükleyerek veri tabanını eski hâline getiriniz.
- Henüz veri tabanı yedeklemeyi ve geri yüklemeyi bilmiyorsanız, veritabanındaki tüm tabloların birer kopyasını oluşturup bu kopya tablolar üzerinde yukarıda istenenleri yapınız. İşiniz bittiği zaman bu kopya tabloları siliniz.

Özet

- **UPDATE Deyimi**
- Tablodaki verilerin güncellenmesi için UPDATE komutu kullanılır.
- UPDATE komutunda where koşul cümleciğinin doğru ve eksiksiz olması hayati önem taşımaktadır.
- UPDATE ile başlayan komutlar çalıştırılmadan önce aynı koşul cümleciklerini içeren select sorgusunun çalıştırılması, güncellenecek kayıtların güncellenmeden önce yeniden gözden geçirilmesine olanak sağlayacağından hata yapılmasını (yanlış kayıtların ya da istenenden daha fazla kaydın güncellenmesini/silinmesini vs.) önlemek adına alınabilecek iyi bir tedbir olur.
- Bazen bir tablodaki sütunu başka bir tabloda bulunan verilerle güncellememiz gerekebilir. Bu durumda join ifadesi ile tablolar birleştirilerek güncelleme işlemi yapılır.
- Benzer şekilde, güncelleme işleminde, aynı sunucuda yer alan başka bir veri tabanındaki tablolardan da veri alınabilir.

- **DELETE Deyimi**
- Tablodaki verilerin silinmesi için DELETE komutu kullanılır.
- DELETE FROM tabloAdi şeklinde kullanılır.
- Delete komutu ile artık ihtiyaç duyulmayan kayıtları, yanlışlıkla eklenmiş kayıtları vs. silebiliriz.
- "DELETE FROM Birey" Sorgusu Birey tablosundaki tüm kayıtları siler. Eğer tüm kayıtların silinmesi gerekmiyorsa, koşul cümlecikleri sorguya dâhil edilip sadece silinmesi istenen kayıtların sorgudan etkilenmesi sağlanmalıdır.
- UPDATE komutunda olduğu gibi DELETE komutunda da koşul cümlecikleri hayati önem arz etmektedir.
- DELETE ile başlayan komutlar çalıştırılmadan önce aynı koşul cümleciklerini içeren select sorgusunun çalıştırılması, silinecek kayıtların silinmeden önce yeniden gözden geçirilmesine olanak sağlayacağından hata yapılmasını (yanlış kayıtların ya da istenenden daha fazla kaydın silinmesini vs.) önlemek adına alınabilecek iyi bir tedbir olur.
- DELETE komutundan tüm satır etkilenir. Sadece bir(kaç) sütunun verisini silmek gibi bir durum söz konusu değildir. Tüm satırı değil de bazı sütunları silmemiz gerekiyor ise bunu silmek istediğimiz sütunları NULL veyahut'' gibi boş değerler kullanarak update komutu ile güncellemek sureti ile gerçekleyebiliriz. Eğer silmek istenen sütunlar tablonun tamamı için silinmek isteniyorsa (where koşul cümleciği kullanılmadan silinmek isteniyorsa) ve sütunlar bir daha kullanılmayacaksa 'alter table ... drop column ... ' komutu ile istenen sütunlar tablodan tamamen silinebilir.

Özet (devamı)

- DELETE komutu ile aynı anda sadece bir tablodan veri silinebilir.
- Bazen bir tablodan veri silineceği zaman, başka tablo(lar)daki veriler kullanılarak silinmesi gerekebilir. Bu durumda join ifadesi ile tablolar birleştirilerek silme işlemi yapılır.

- **Veri bütünlüğü**
- Tablodan satır silerken veri tabanındaki bütünlüğü bozmamamız gerekir, dolayısı ile bir tablodan satır silerken silinecek satırı referans alan başka bir tablo varsa silme işleminden sonra veri bütünlüğü bozulacaktır. Bu yüzden silinecek satırların bağlantılarını kontrol ederek silmemiz gerekir. Birbiri ile ilişkili tablolar yabancı anahtar ile birbirlerine bağlı ise bu kontrolü SQL Server'in kendisi zaten yapmaktadır.

- **TRUNCATE deyimi**
- TRUNCATE deyimi koşul cümleciği olmadan yazılan DELETE komutu ile aynı silme işlemini yapar.
- TRUNCATE komutu, bir tablodaki tüm verilerin silinmek istendiği fakat bu silme işlemi için satır bazlı transaction log kayıtlarının oluşturulmasının istenmediği durumlarda kullanılır. Dolayısı ile TRUNCATE komutu kullanıldıktan sonra ROLLBACK komutu ile işlem geri alınamaz (çünkü TRUNCATE komutu sonrası transaction log kaydı tutulmaz).
- TRUNCATE komutunda istense de WHERE deyimi kullanılamaz. Koşul belirtilecekse DELETE komutu kullanılmalıdır.
- Bir tablodan herhangi bir kayıt silindikten sonra silinen kaydın birincil anahtar sütunundaki değer artık boşta olmasına rağmen yeniden kullanılmaz. Birey tablosundan koşul cümleciği yazmadan silme işlemi yapıldığında tabloda kayıt kalmaz ama tabloya ekleme yapıldığında ID sütununun alacağı değer, en son verilmiş ID değerinden devam eder. Bu durum silme işleminin DELETE komutu ile yapılması hâlinde geçerlidir. Tablodaki tüm veriler TRUNCATE deyimi ile silinirse eğer tabloya yeniden ekleme yapıldığında birincil anahtar sütununa (ID) başlangıç değerinden itibaren değer verilir.

DEĞERLENDİRME SORULARI

1. Tablodaki verileri güncellemek için aşağıdaki komutlardan hangisi kullanılır?
   a) SELECT
   b) INSERT
   c) UPDATE
   d) DELETE
   e) TRUNCATE
2. Bir sorgu içerisinde aynı anda kaç farklı tablodaki veriler güncellenebilir?
   a) 1
   b) 2
   c) 256
   d) 1024
   e) Sınırsız

3. UPDATE komutu aşağıdaki işlemlerden hangisini yapar?
   a) Tabloda sütun adı değiştirir.
   b) Tablodaki verileri siler.
   c) Tabloya yeni kayıt ekler.
   d) Tablodaki verileri günceller.
   e) Tablonun bir kopyasını oluşturur.

4. Tablodaki verileri silmek için aşağıdaki komutlardan hangisi kullanılır?
   a) SELECT
   b) INSERT
   c) UPDATE
   d) DELETE
   e) DROP

5. Aynı anda en fazla kaç tablodan veri silinebilir?
   a) 1
   b) 2
   c) 256
   d) 1024
   e) 4096

6. Tablodan aynı anda en fazla kaç sütundaki veriler silinebilir?
   a) 1
   b) 2
   c) 256
   d) 1024
   e) Sütun sayısından bağımsız olarak tüm satır silinir.

7. Aşağıdakilerden hangisi TRUNCATE komutu ile DELETE komutunun farklarından biridir?
   a) DELETE komutunda WHERE koşul cümlecikleri kullanılamaz.
   b) DELETE komutu ile silinen veriler ROLLBACK komutu ile geri getirilemez.
   c) TRUNCATE komutu ile tüm tablo silindikten sonra birincil anahtar sütunu başlangıç değerinden itibaren değer alır.
   d) TRUNCATE komutu ile sadece bazı sütunların silinmesi mümkündür.
   e) Her iki komutun uygulamada bir farkı yoktur.

   ID (birincil anahtar) sütunundaki maksimum değerin 1110 olduğu birbirinin kopyası iki ayrı tablodan birincisinden DELETE komutu ile ikincisinden TRUNCATE komutu ile tüm satırlar silindikten sonra yine her iki tabloya birer satır veri ekleniyor.

8. Buna göre yeni eklenen satırlarda sırası ile birinci tablodaki ve ikinci tablodaki ID sütunun değeri kaç olur?
   a) 1110 – 1111
   b) 1 – 1
   c) 1 - 1111
   d) 1111 – 1
   e) 1111 – 1110

9. UPDATE ve DELETE komutları ile alakalı olarak aşağıda belirtilen ifadelerden hangisi yanlıştır?
   a) Değeri SQL tarafından oluşturulan birincil anahtar sütunu güncellenemez.
   b) Yabancı anahtar sütunu güncellenemez.
   c) Bir sütun hâlihazırda içerisinde bulunan değer kullanılarak güncellenebilir.
   d) Bir sütun başka bir tablodaki sütundaki veriler kullanılarak güncellenebilir.
   e) Bir tablodan, başka bir tablodaki veriler kullanılarak silme işlemi yapılabilir.

I. Hesaplatılmış bir sütun güncellenebilir.
II. Bir tabloda aynı anda birden fazla satır güncellenebilir.
III. Aynı anda birden fazla tablo güncellenebilir.

10. Tablodaki verileri güncelleme ile ilgili olarak yukarıdakilerden hangisi veya hangileri doğrudur?
    a) Yalnız I
    b) Yalnız II
    c) Yalnız III
    d) I ve II
    e) II ve III

Cevap Anahtarı

1.c, 2.a, 3.d, 4.d, 5.a, 6.e, 7.c, 8.d, 9.b, 10.b

YARARLANILAN KAYNAKLAR

DELETE (Transact-SQL) 11.06.2019 tarihinde http://technet.microsoft.com/en-us/library/ms189835.aspx adresinden erişildi.

Gözüdeli, Y. (2012). Yazılımcılar için SQL Server 2008 R2 & Veritabanı Programlama, 5. Baskı, Ankara: Seçkin Yayıncılık.

Güncelleştirme (Transact-sql) 10.06.2019 tarihinde http://technet.microsoft.com/tr-tr/library/ms177523.aspx adresinden erişildi.

Özhan, C. (2013). İleri Seviye T-SQL Programlama, 1. Baskı, İstanbul: KODLAB Yayınevi.

Petkovic, D. (2006). Microsoft SQL Server, 1. Baskı, İstanbul: Alfa Yayınları

Sil (Transact-sql) 11.06.2019 tarihinde http://technet.microsoft.com/tr-tr/library/ms189835.aspx adresinden erişildi.

UPDATE (Transact-SQL) 10.06.2019 tarihinde http://technet.microsoft.com/en-us/library/ms177523.aspx adresinden erişildi.

GÖRÜNTÜ (VIEW), STORE PROSEDÜR VE FONKSİYONLAR

İÇİNDEKİLER

- Görüntü Kavramı
  - Görüntü Oluşturma
  - Görüntü Silme
- Store Prosedür Kavramı
  - Store Prosedür Oluşturmak
  - Store Prosedür Çalıştırmak
  - Store Prosedürlerde Parametre Kullanımı
- Fonksiyon Kavramı
  - Kullanıcı Tanımlı Fonksiyon Oluşturma

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Görüntü oluşturabilecek, silebilecek, değiştirebilecek, kullanabilecek,
  - Store Prosedür oluşturabilecek, silebilecek, değiştirebilecek, farklı şekillerde kullanımlarını öğrenebilecek,
  - Fonksiyon oluşturabilecek, silebilecek, değiştirebilecek, farklı şekillerde kullanımlarını öğrenebileceksiniz.

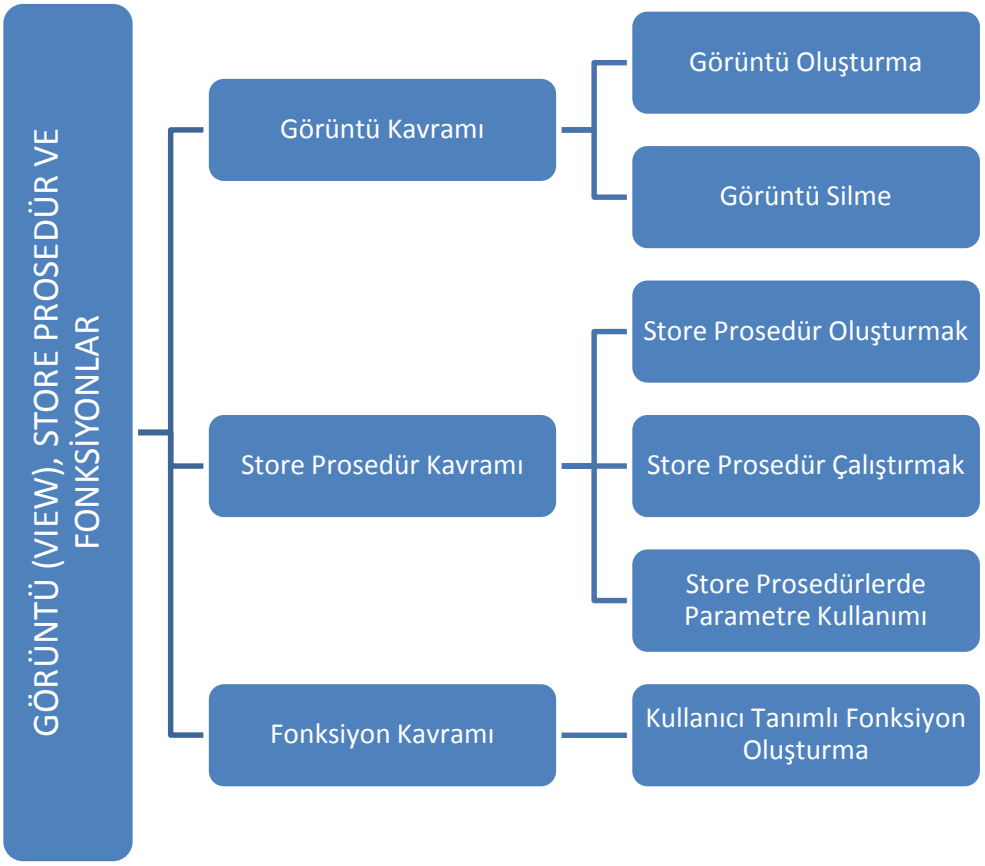
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Arş. Gör. Yakup BAYOĞLU

ÜNİTE

11


GÖRÜNTÜ (VIEW), STORE PROSEDÜR VE FONKSİYONLAR
Görüntü Kavramı
Görüntü Oluşturma
Görüntü Silme
Store Prosedür Kavramı
Store Prosedür Oluşturmak
Store Prosedür Çalıştırmak
Store Prosedürlerde Parametre Kullanımı
Fonksiyon Kavramı
Kullanıcı Tanımlı Fonksiyon Oluşturma

GİRİŞ

Görüntüler (view), select ifadeleri ile oluşturulmuş gerçek tablo(lar)ın bir görünümüdür. İlgili tablolar join ifadeleri ile birleştirilir, koşul cümlecikleri ile filtrelenir ve sadece ihtiyaç duyulan sütun(lar) ve/veya hesaplatılan sütun(lar) sorgu sonucuna dahil edilerek tanımlanır. Görüntüler de gerçek tablolar gibi sorgulanabilir, diğer tablolarla birleştirilerek sorgularda kullanılabilir.

Store prosedürler (SP), birçok programlama dilinde var olan fonksiyonlara karşılık gelir. İçerisinde T-SQL kodları, değişkenler, if-else yapıları, while döngüsü vs. kullanılır, parametre alabilir ve geriye bir değer veya tablo(lar) döndürebilir. Sık kullanılan kod blokları SP olarak kaydetmek ve ihtiyaç duyulan yerlerde ilgili kod bloğunu yeniden yazmak yerine SP'yi çağırmak kod yazımında önemli ölçüde kolaylık sağlar. Ayrıca bu kod bloğunda bir değişiklik yapılması gerektiğinde, değişikliği sadece SP'de yapmak yeterli olacaktır, değişiklik SP'nin kullanıldığı her yere yansıyacaktır. SP kullanılmadığı takdirde yapılması gereken değişiklik ilgili kod bloğunun kullanıldığı her yerde yapılması gerekecektir.

Store prosedürler oluşturulduğunda ayrıştırma, derleme ve çalıştırma aşamalarından geçer. SP'ler ilk seferden sonraki çağrılmalarında yeniden derlenmezler, hafızada saklanan derlenmiş hâli çalıştırılır. Bu sebeple SP'ler hızlı çalışır.

Fonksiyonlar yapısal olarak store prosedürlere benzer. Store prosedürler ile görüntülerin avantajlarını bünyesinde toplamıştır. Fonksiyonlar parametre alabilir, geriye *bir* değer veya *bir* tablo döndürebilir, görüntülerde olduğu gibi select sorgularına dâhil edilebilir ve join ifadelerinde kullanılabilir. SQL'in hazır fonksiyonları mevcuttur. Kullanıcılar kendi ihtiyaçlarına yönelik kullanıcı tanımlı fonksiyon oluşturabilir.

GÖRÜNTÜ KAVRAMI

Görüntüler (view), select ifadeleri ile oluşturulmuş sanal tablolardır. Bazen mevcut tablolarımızın daha değişik hâllerine ihtiyaç duyarız. Sık kullanılan bazı sorgularda aynı tablonun sadece birkaç sütununa ihtiyaç duyduğumuzda, bahsi geçen tablonun sadece ilgili sütunlarını içeren bir görüntü tanımlayıp sorgularımızı bu görüntü üzerinden çalıştırabiliriz. Bu görüntüye, tablonun daraltılmış bir kopyası gözü ile bakabiliriz. Aynı şekilde tablodaki sadece bazı satırlara ihtiyaç duyduğumuz zamanlar olur. Normalde bunu select ifadesinin sonuna where koşul ifadesi ekleyerek elde edebiliyoruz fakat ilgili kıstas(lar)ı where koşuluna yazarak bu şekilde bir görüntü tanımlarsak eğer where koşulunu yazmaya gerek kalmadan sorgumuzu görüntü üzerinden çalıştırabiliriz. Bu şekildeki bir görüntüyü de tablonun filtrelenmiş bir kopyası olarak değerlendirebiliriz. Bazen de tablonun hem daraltılmış, hem de filtrelenmiş hâline ihtiyaç duyabiliriz ki bu durumda da yine görüntü tanımlayarak isteğimizi gerçekleştirebiliriz. Bu ihtiyacımızı zaten;

Görüntüler (view), select ifadeleri ile oluşturulmuş sanal tablolardır.

```
SELECT
     Sütun1,
     sütun2,
     [...]
```

```
FROM
     TabloAdi
[WHERE Koşul]
```

şeklinde bir sorgu yazarak da karşılayabiliyoruz, öyle ise görüntü tanımlamaya neden ihtiyaç duyduk? Cevap: Sık kullandığımız bazı sorguları daha kolay bir hâle getirmek için. Çoğunlukla aynı sütunları seçerek yazdığımız sorgularda select ifadesinden sonra sütun adlarını sıralamak yerine 'SELECT * FROM görüntü_adı' şeklinde daha kısa bir sorgu cümlesi kullanabiliriz. Aynı şekilde belirli bir tablo üzerinde sık kullandığımız where koşul(lar)ı var ise 'SELECT * FROM tablo_adı WHERE koşul1, koşul2 …' şeklinde uzunca bir sorgu cümlesi yerine 'SELECT * FROM görüntü_adı' şeklinde daha kısa bir sorgu cümlesi kullanabiliriz.

Görüntüler için sanal tablolardır demiştik, her ne kadar tıpkı tablolarda olduğu gibi 'SELECT * FROM görüntü_adı' şeklinde sorgular çalıştırabiliyor olsak da gerçekte böyle bir tablo yoktur. Görüntüler sadece kaydedilmiş sorgulardan ibarettir.

Bir görüntünün verilerini çektiği tabloya temel tablo denir ve temel tablonun bazı sütunlarını ve/veya bazı satırlarını içerir.

Bir görüntünün verilerini çektiği tabloya temel tablo denir ve yukarıda da ifade edildiği gibi temel tablonun bazı sütunlarını ve/veya bazı satırlarını içerir. Görüntü, temel tablonun tamamını da kapsayabilir; teknik olarak böyle bir görüntü tanımı mümkündür fakat anlamlı bir kullanım şekli değildir. Bu şekil bir tanımlama görüntü oluşturmanın amacının dışına çıkmış olur.

Görüntüler birden fazla temel tablodan sütunlar içerebilir ki bu şekildeki kullanım görüntünün daha efektif bir şekildeki kullanımıdır.

Görüntülere neden ihtiyaç duyarız?

- Bazı kullanıcıların, önemli tabloların sadece belirli sütunlarına ve/veya belirli satırlarına erişmelerine müsaade edilmek istendiğinde: Uygun bir görüntü tanımı yapılıp kullanıcıların ilgili tablolara değil, sadece tanımlanan görüntüye erişme yetkisi verilmek sureti ile istenen amaca ulaşılmış olur.
- Kullanıcılara hesaplatılmış değer gösterilmek istendiğinde: Örneğin öğrencilerin TC kimlik numaralarının sadece ilk 8 hanesinin gösterilmek istendiği durumda veya öğrencinin doğum tarihi bilgisini kullanarak yaşını hesaplatıp kullanıcılara bu bilginin gösterilmek istendiği durumda görüntü kullanılabilir.
- Farklı tablolarda yer alan bilgilerin beraber gösterilmek istendiği durumlarda: Örneğin öğrenciye ait bilgilerden 4 farklı tabloda yer alan adı, soyadı, öğrenci numarası, bölümü ve adresi bilgilerini tek bir tablodan sorguluyor gibi sorgulayabilmek için. Bu durumda 4 farklı tabloyu 'INNER JOIN' ifadeleri ile birleştirmek yerine tek bir görüntü üzerinden sorgumuzu yazabiliriz.
- Çok karmaşık sorguları basitleştirmek için. (Gözüdeli, 2012)

Görüntü Oluşturma

Tek bir tablo kullanarak görüntü oluşturmak

Genel olarak aşağıda ifade edildiği gibi oluşturulur.

```
CREATE VIEW Görüntü_Adı
AS
SELECT
      Sütun1,
      Sütun2,
      [...]
FROM
      TemelTabloAdı
```

Mesela Birey tablosundan sadece adı, soyadı ve doğum tarihi bilgilerini içeren bir vw_BireyOzet isimli bir görüntü oluşturalım:

```
CREATE VIEW vw_BireyOzet
AS
SELECT
      Adi,
      Soyadi,
      DogumTarihi
FROM Birey
GO
```

İpucu: tüm görüntüleri isimlendirirken 'vw_' ile başlamak yazacağımız kodların anlaşılabilirliğine katkı sağlayacaktır. Böylece yazmış olduğumuz sorgular incelenirken vw_ şeklinde başlayan nesnelerin birer görüntü olduğu kolayca hatırlanabilir.

'SELECT * FROM vw_BireyOzet' şeklindeki bir sorgu Birey tablosundaki sadece görüntü tanımında belirtilen sütunların sonuç olarak döndürülmesini sağlar.

```
CREATE VIEW vw_BireyErkek
AS
SELECT
      *
FROM
      Birey
WHERE
      Cinsiyeti = 'Erkek'
GO
```

Görüntü tanımlanırken sütunlar için veri tipi belirtilmez, ilgili sütun temel tablo tanımında hangi veri tipi ile tanımlanmış ise görüntüde de aynı veri tipi ile bulunur.

Yukarıda sadece erkek bireylerin sonuç olarak döndürüleceği bir görüntü tanımlanmıştır.

```
CREATE VIEW vw_BireyOzetErkek
AS
SELECT
      TCNo,
      Adi,
      Soyadi,
      Dogumtarihi
FROM Birey
WHERE
      Cinsiyeti = 'Erkek'
```

```
GO
```

Şeklindeki görüntü tanımı ise sadece erkek bireylerin TC kimlik numarası, adı, soyadı ve doğum tarihi bilgilerini sonuç olarak döndürecek bir görüntü tanımıdır.

Not: Görüntü tanımlarken sütunlar için veri tipi belirtilmez, ilgili sütun temel tablo tanımında hangi veri tipi ile tanımlanmış ise görüntüde de aynı veri tipi ile bulunur.

Not: Görüntü tanımlarken en fazla 1024 sütun kullanabiliriz.

İpucu: Görüntü üzerinden sorgu yaparken de sütun adlarını belirterek sorgu sonuçları daraltılabilir. Aynı şekilde görüntü üzerinden sorgu yapılırken where koşul ifadeleri kullanılarak sonuçlar filtrelenebilir. Kısaca görüntüleri gerçek tablolar gibi kullanabilmekteyiz.

```
SELECT
      TCNo,
      Soyadi
FROM
      vw_BireyOzetErkek
WHERE
      Adi = 'Ali'
```

Şeklinde sorgular yazabiliriz.

Birden fazla temel tablo kullanarak görüntü oluşturmak

Görüntü tanımlarken birden fazla temel tablo kullanılabilir. Genel olarak şu şekilde ifade edilebilir:

```
CREATE VIEW görüntü_Adı
AS
SELECT
      Sütun1,
      Sütun2,
      Sütun3,
      [...]
FROM
      Tablo1,
      Tablo2,
      [...]
WHERE
      Birleşme koşulları
GO
```

Tabloları birleştirebildiğimiz gibi görüntüleri de JOIN deyimleri ile birleştirerek sorgu yazabiliriz.

Örnek olarak dört farklı tablo kullanarak ‘vw_OgrenciGenelBilgi’ isimli bir görüntü oluşturalım:

```
CREATE VIEW vw_OgrenciGenelBilgi
AS
SELECT
      BRY.Adi,
      BRY.Soyadi,
      OB.OgrenciNo,
      BRM.Adi,
      A.Adresi
FROM
      Birey BRY,
```

```
        Ogrenci_Bolum OB,
        Birim BRM,
        Adres A
WHERE
        OB.Birey_ID = BRY.ID
        And OB.Birim_ID = BRM.ID
        And A.Birey_ID = BRY.ID
GO
```

Yukarıdaki örnek, tablolar join ifadeleri ile birleştirilerek de yazılabilirdi.

İpucu: Tabloları birleştirebildiğimiz gibi görüntüleri de JOIN deyimleri ile birleştirerek sorgu yazabiliriz. Aynı şekilde görüntüleri de tablolar ile birleştirerek tek sorgu içerisinde kullanabiliriz.

Not: Görüntü tanımlarken en fazla 256 adet tabloyu birleştirebiliriz.

İpucu: Bir görüntü oluşturulurken kullanılan temel tablolar içerisinde bir başka görüntü de yer alabilir.

Görüntü Yapısını Değiştirme

Tanımlı görüntülere bazen sütun eklemek ya da çıkarmak gerekebilir. Bu durumda görüntü tanımını değiştirmemiz gerekir. Genel kullanımı:

Görüntü tanımında değişiklik yapmak için ALTER, görüntüyü silmek için DROP komutu kullanılır.

```
ALTER VIEW görüntü_Adı
AS
SELECT
        Sütun1,
        Sütun2,
        [...]
FROM
        TemelTabloAdı
GO
```

Şeklindedir. Dikkat edilirse yukarıdaki kalıp görüntü oluşturma kalıbına çok benzemektedir. Yapacağımız şey sadece 'CREATE' komutu yerine 'ALTER' komutunu kullanmak ve select ifadesinden sonrasını ihtiyacımız doğrultusunda yeniden yazmaktır. Örnek olarak yukarıda tanımladığımız vw_BireyOzet görüntüsüne Birey tablosundaki TCNo sütununu da dâhil etmek istersek yazacağımız kod şu şekilde olacaktır:

```
ALTER VIEW vw_BireyOzet
AS
SELECT
        TCNo,
        Adi,
        Soyadi,
        DogumTarihi
FROM Birey
GO
```

Görüntü Silme

Görüntüleri silmek için 'DROP' komutu kullanılır:

```
DROP VIEW görüntü_Adı
```

'vw_BireyOzet' isimli görüntümüzü silmek istersek yazmamız gereken kod:

```
DROP VIEW vw_BireyOzet
```

şeklindedir.

STORE PROSEDÜR (SAKLI YORDAM) KAVRAMI

Store prosedürler (SAKLI YORDAM) (SP) birçok programlama dilinde var olan fonksiyon yapılarına karşılık gelir. Blok(lar) hâlindeki bir yığın SQL komutlarını bir prosedür içerisine yazıp daha sonra lazım olduğu yerde bu prosedürü çağırmak/çalıştırmak sureti ile tüm bahsi geçen kod bloğunu çalıştırmış oluruz. SP'lerin bu yapısı, çok sık veya çeşitli yerlerde ihtiyaç duyduğumuz aynı kod bloğunu her lazım olduğunda tüm kod bloğunu yazmak yerine bir defa SP olarak tanımladıktan sonra, ihtiyaç anında sadece SP'ü çağırmak sureti ile kod yazımını daha kolay ve daha anlaşılır hâle getirebiliriz. SP tanımı içinde yapacağımız değişiklik SP'ün kullanıldığı her yere yansıyacaktır, SP kullanmamış olsa idik aynı kod bloğunun kullanıldığı her yere ilgili değişikliğin tek tek yansıtılması gerekecekti.

SP'lerin oluşturulurken geçtiği aşamalar şu şekildedir:

Ayrıştırma (Parsing): Bu aşamada sentaks kontrolü yapılır. Yani yazılan ifadelerin SQL yazım standartları ve kurallarına uyup uymadığı, komut ifadelerinin doğru yazılıp yazılmadığı vs. kontrol edilir. Veri tabanında bulunmayan bir tablodan select ile veri çekiliyor olması bu aşamada hata oluşturmaz ama "select" ifadesi yanlış yazılmış olursa (mesela "slect" şeklinde yazılmış olsa) hata oluşur. Bu aşamada sorgu ağacı ya da sıra ağacı denen yapı ortaya çıkar.

Derleme aşamasından sonra oluşan çalışma planı hafızada (RAM) saklanır ve yeniden oluşturulmasına gerek kalmaz. Bu yüzden SP'ler hızlı çalışır.

Derleme (Compiling): Bir önceki aşamada oluşturulan sorgu ağacından çalışma planı çıkartılır, hak ve yetkiler kontrol edilerek güvenlik denetlenir. Çalışma planı, hangi aşamada hangi kontrollerin, kısıtların (constraint) veya indekslerin kullanılacağı gibi tanımları içerir.

Çalıştırma (Executing): Bir önceki aşamada oluşturulan çalışma planı üzerinden işlemler gerçekleştirilir.

SP'ler ilk defa çalıştırılırken yukarıdaki aşamaların hepsinden geçilir, daha sonraki çağrılmalarında ise sadece üçüncü aşama olan çalıştırma (executing) aşaması gerçekleşir. Çünkü derleme aşamasından sonra oluşan çalışma planı hafızada (RAM) saklanır ve (SP'ün sonraki kullanımlarında) yeniden oluşturulmasına gerek kalmaz. Bu yüzden SP'ler aynı içeriğe sahip script (kod) bloğuna göre daha hızlı çalışır. Bu durum SP kullanmanın en önemli avantajlarındandır.

Store Prosedür Oluşturmak

```
CREATE PROCEDURE prosedür_Adı
      @parametre1,
      @parametre2,
      [...]
AS
      T-SQL kodları
GO
```

Her SP parametre almak zorunda değildir, yukarıdaki kalıp parametreli bir SP ye aittir. SQL Server, içerisinde hazır sistem store prosedürleri barındırmaktadır ve bu SP'ler standart olarak 'sp' ile başlar. Örnek 'sp_tables' bir sistem SP'dür ve veri tabanı içindeki tabloların listesini sonuç olarak getirir.

```
EXEC sp_tables
```

Yukarıdaki sorgu aşağıdaki sonucu getirir:

Results | Messages

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	REMARKS
1	OgrenciBilgi	dbo	Birey	TABLE	NULL
2	OgrenciBilgi	dbo	Birey_Iletisim	TABLE	NULL
3	OgrenciBilgi	dbo	Birim	TABLE	NULL
4	OgrenciBilgi	dbo	Cinsiyet	TABLE	NULL
5	OgrenciBilgi	dbo	Ders	TABLE	NULL
6	OgrenciBilgi	dbo	Fakulte	TABLE	NULL
7	OgrenciBilgi	dbo	Ogrenci_Bolum	TABLE	NULL
8	OgrenciBilgi	dbo	Ogrenci_Ders	TABLE	NULL
9	OgrenciBilgi	dbo	Ogretmen_Bolum	TABLE	NULL
10	OgrenciBilgi	dbo	Ogretmen_Ders	TABLE	NULL

Şekil 11.1. 'sp_tables' SP'ünün Çalıştırılmasından Elde Edilen Sonuç

İpucu: Kendi tanımlayacağınız SP'leri isimlendirirken 'usp' ile başlatmanız kendi SP'leriniz ile sistem SP'lerinin karıştırılmaması için tavsiye edilir.

SP içerisinde başka bir SP çağrılabilir.

Not: SP içinden başka bir SP çağırmaya SQL Server en fazla 32. seviyeye kadar müsaade etmektedir. Rekürsif bir SP yazılmak istenirse bu duruma dikkat edilmelidir.

SQL server SP içinden başka bir SP çağırmaya SQL Server en fazla 32. seviyeye kadar müsaade etmektedir. Rekürsif bir SP yazılmak istenirse bu duruma dikkat edilmelidir.

Fakülteler ve bölümlerini sonuç olarak getirecek bir SP aşağıdaki gibi oluşturulabilir:

```
CREATE PROCEDURE usp_FakulteBolumleri
AS
      SELECT
            F.Adi,
            B.Adi
      FROM
            Birim B
            INNER JOIN Fakulte F
            ON F.ID = B.Fakülte_ID
GO
```

Store Prosedür Çalıştırmak

Çalıştırılacak olan kod yığını sadece SP'den ibaret ise veya kod bloğu SP ile başlıyorsa sadece SP adının (ve varsa parametrelerinin) yazılması yeterlidir. Fakat SP, kod bloğunun başında değil ise "exec" veyahut "execute" deyimi ile beraber yazılmalıdır.

```
sp_tables

sp_tables
SELECT * FROM Birey

SELECT * FROM Birey
EXEC sp_tables
```

Yukarıdaki üç farklı kullanım da doğrudur.

Store Prosedürlerde Değişiklik Yapmak

SP'lerde değişiklik yapmak için 'ALTER' komutu kullanılır. (Petkovic, 2006) Genel kullanımı:

```
ALTER PROC[EDURE] Prosedür_Adı
AS
      T-SQL kodları
GO
```

Şeklindedir.

Daha önce oluşturduğumuz "usp_FakulteBolumleri" isimli SP'ü fakülte ve bölüm isimlerini sıralı getirecek şekilde değiştirelim:

Parametreler SP'lerin daha dinamik, dolayısı ile daha efektif bir şekilde kullanımını sağlar.

```
ALTER PROCEDURE usp_FakulteBolumleri
AS
      SELECT
            F.Adi,
            B.Adi
      FROM
            Birim B
            INNER JOIN Fakulte F
            ON F.ID = B.Fakülte_ID
      ORDER BY
            F.Adi,
            B.Adi
GO
```

Store Prosedürlerde Parametre Kullanımı

SP'ler parametre ile de çağrılabilir. Bu şekilde kullanımı SP'lerin daha dinamik, dolayısı ile daha efektif bir şekilde kullanımını sağlar. SP'ler geriye sonuç da döndürebilir. Aldıkları parametreler girdi parametresi veya çıktı parametresi şeklinde olabilir.

Not: SP'ler en fazla 2100 parametre alabilir.

Girdi parametreler

"usp_FakulteBolumleri" isimli SP'yi parametre ile çalışacak şekilde güncelleyelim:

```
ALTER PROCEDURE [dbo].[usp_FakulteBolumleri]
      @FakulteId INT
AS
      SELECT
            F.Adi,
            B.Adi
      FROM
```

```
            Birim B
            INNER JOIN Fakulte F
            ON F.ID = B.Fakülte_ID
      WHERE
            F.ID = @FakulteId
      ORDER BY
            B.Adi
GO
```

Yukarıdaki değişiklikten sonra usp_FakulteBolumleri SP'ünü yanında @FakulteId parametresi ile çağırmamız gerekmektedir, SP artık sadece parametre ile ID numarasını gönderdiğimiz fakültenin bölümlerini geri döndürmektedir.

Girdi parametresi ile store prosedür çağırmak

```
EXEC usp_FakulteBolumleri @FakulteId = 2
```

Veya parametre adını yazmadan da

```
EXEC usp_FakulteBolumleri 2
```

Şeklinde SP ler parametre ile çağrılabilir.

Not: Birden fazla parametrenin kullanıldığı durumlarda eğer SP, parametre adı yazılmadan çağrılacak ise parametrelerin doğru sıra ile yazılması gerekmektedir.

Birden fazla parametrenin kullanıldığı durumlarda eğer SP, parametre adı yazılmadan çağrılacak ise parametrelerin doğru sıra ile yazılması gerekmektedir.

Bazen gelen parametrenin zorunlu olmaması istenebilir. Bu durumda SP'nin tanımı içinde ilgili parametreye varsayılan (default - sabit) bir değer atanır ve SP çağrılırken bahsi geçen parametre yazılmadan çağırılırsa parametrenin varsayılan değeri üzerinden işlem görülür. Örneğin @Dil parametresi ile gelen Dil ID numarası değerine göre geri döndüreceği mesajın dilini ayarlayan bir SP'ümüz olduğunu, mesajların @Dil parametresi 1 olduğunda Türkçe, 2 olduğunda İngilizce ve 3 olduğunda Almanca olarak geri döndürüldüğünü ve @Dil parametresinin varsayılan değerinin 1 olduğunu varsayalım. Bu durumda SP çağırılırken eğer dil parametresi belirtilmez ise SP sonuç mesajlarını Türkçe olarak döndürecektir.

```
CREATE PROCEDURE usp_Karsilama
      @BireyID   INT,
      @Dil       INT = 1
AS
      SELECT
            CASE @Dil
                  WHEN 1 THEN 'Hoşgeldiniz '
                  WHEN 2 THEN 'Welcome '
                  WHEN 3 THEN 'Willkommen '
            END + B.Adi
      FROM
            Birey B
      WHERE
            ID = @BireyID
GO
```

Şeklinde tanımlanan usp_Karsilama SP'ü:

```
EXEC usp_Karsilama 6, 2
```

Şeklinde çağrıldığında mesajını İngilizce olarak;

```
EXEC usp_Karsilama 6
```

Şeklinde @Dil parametresi yazılmadan çağrıldığında ise mesajını Türkçe olarak döndürecektir.

Not: SP'ler, görünümler ve fonksiyonların aksine select sorgularına dâhil edilemez.

Çıktı parametreleri ile çalışmak

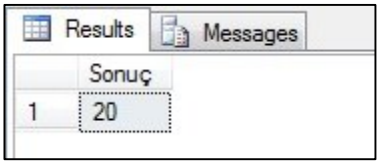
Kısaca SP'e gönderilen parametrenin içinin doldurularak çağrıldığı yere geri döndürülmesi şeklinde özetlenebilir. SP tanımında geri döndürülecek olan parametre(ler), veri tipinden sonra 'OUT' ifadesi yazılarak çıktı parametresi olduğu belirtilir.

```
CREATE PROCEDURE [dbo].[usp_Carpma]
      @Sayi1     INT,
      @Sayi2     INT,
      @Sonuc     INT OUT
AS
      SELECT
            @Sonuc = @Sayi1 * @Sayi2
GO
```

Yukarıda aldığı ilk iki parametredeki değerleri çarpıp üçüncü parametreye atayan "usp_Carpma" SP'ü tanımlanmıştır.

```
DECLARE @Sonuc AS INT
EXEC usp_Carpma 4,5, @Sonuc OUT
SELECT @Sonuc
```

Yukarıdaki kod çalıştırıldığında aşağıdaki sonuç elde edilir:



Şekil 11.2. Yukarıdaki Kod Çalıştırıldıktan Sonraki Ekran Görüntüsü

SP'nin paramatre üzerinden değil, direk kendisinin bir sonuç döndürmesi istendiğinde 'Return' deyimi kullanılır.

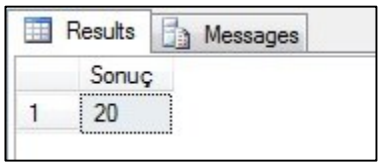
Return deyimi

Bazen SP'ün parametre üzerinden değil, direk kendisinin bir sonuç döndürmesi istenebilir. Bu durumda return deyimi ile istenen / hesaplanan değer geri döndürülür ve kod içinde bu değer bir değişkene atanır. Bir önceki SP'ümüzü aşağıdaki şekilde güncelleyelim:

```
ALTER PROCEDURE usp_Carpma
      @Sayi1     INT,
      @Sayi2     INT
AS
      RETURN @Sayi1 * @Sayi2
GO
```

Örnek kullanım:

```
DECLARE @Sonuc AS INT
EXEC @Sonuc = usp_Carpma 4,5
SELECT 'Sonuç' = @Sonuc
```



Şekil 11.3. Yukarıdaki Kod Çalıştırıldıktan Sonraki Ekran Görüntüsü

Not: Return deyimi ile geriye sadece bir değer döndürülebilir.
İpucu: Birden fazla değerin geri döndürülmesine ihtiyaç varsa çıktı parametresi kullanılabilir.

Return deyimi SP'ü sonlandırmak için de kullanılabilir. Bazen SP içinde birtakım kontroller yapıldıktan sonra duruma göre SP'ün çalıştırılmasına devam edilmek istenilmeyebilir. Daha önce tanımladığımız usp_Karsilama SP'ünü geçerli bir dil ID numarası girilip girilmediğini kontrol edecek şekilde güncelleyelim:

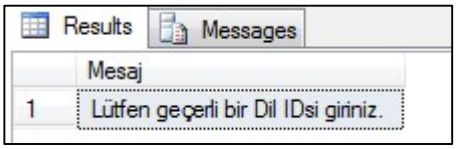
```
ALTER PROCEDURE usp_Karsilama
      @BireyID   INT,
      @DilID          INT = 1
AS
      IF @DilID NOT IN (1,2,3)
      BEGIN
            SELECT 'Mesaj' = 'Lütfen geçerli bir
                  Dil IDsi giriniz.'
            RETURN
      END
      SELECT
            CASE @DilID
                  WHEN 1 THEN 'Hoşgeldiniz '
                  WHEN 2 THEN 'Welcome '
                  WHEN 3 THEN 'Wilcommen '
            END + B.Adi
      FROM
            Birey B
      WHERE
            ID = @BireyID
GO
```

SP artık @DilID parametresinin geçerli bir ID numarası olup olmadığını kontrol etmekte ve buna göre SP devam etmekte veya etmemektedir.

```
EXEC usp_Karsilama 6, 4
```

Sorgusu, karşılama mesajı değil hata mesajı döndürecektir.

SP kod bloğunun başına "set nocount on" yazarak işlemden etkilenen kayıt sayısı mesaj(lar)ının gelmesi engellenebilir.



Şekil 11.4. 'usp_Karsilama' SP'sinin Hata Mesajı Döndürecek Şekilde Bir Kullanım Örneği

Nocount kullanımı

SQL Server her işlemden sonra etkilenen kayıt sayısını mesaj olarak döndürür. SP içerisindeki işlem(ler)in etkilemiş olduğu kayıt sayıları ile ilgilenilmiyorsa eğer SP kod bloğunun başına "set nocount on" yazarak bu mesaj(lar)ın gelmesi engellenebilir, bu yaklaşım performans artışı sağlamak içindir. (Özhan, 2013). SP kod bloğunun başında "set nocount on" ayarı yapılırsa blok sonunda "set nocount off" deyimi ile bu ayarı yeniden aktifleştirmek unutulsa bile SQL Server aktifleştirir.

FONKSİYON KAVRAMI

Fonksiyon kavramı programcılığın temel kavramlarındandır. Temel olarak sıkça kullanılacak kod yığını (bloğu) bir fonksiyon içine yazılır ve ihtiyaç duyulduğu her yerde ilgili kod bloğunun yeniden yazılması yerine fonksiyon çağrılır. Bu şekilde kod karmaşası azaltılır, programın okunurluğu ve anlaşılırlığı artar. Ayrıca store prosedürlerde olduğu gibi fonksiyon tanımında değişiklik yapıldığı zaman, bu değişiklik fonksiyonun kullanılmış olduğu her yere yansır, fonksiyon kullanılmasa idi aynı kod bloğunun kullanıldığı her yerde aynı değişikliğin yapılması gerekirdi. Bu yüzden fonksiyon kullanımı programın yönetimini kolaylaştırır.

SQL Server açısından bakıldığında fonksiyonlar, view (görüntü) ve Store prosedür (Saklı Yordam) kullanmanın avantajlarını kendi bünyesinde toplamıştır. Fonksiyon içerisinde, Store prosedür de olduğu gibi çeşitli T-SQL sorguları, if else yapıları vs. kullanılabilir ve fonksiyonlar da parametre alabilir. Fonksiyonun görüntüye benzer yönü ise select sorgularına dahil edilebilmeleridir, ki store prosedürlerde bulunmayan bu özellik ciddi kullanışlılık sağlar. (Henderson, 2013)

Çeşitli SQL fonksiyonları:

```
SELECT MAX(ID) FROM Birey
```

Sorgusundaki MAX() fonksiyonu Birey tablosunun ID sütunundaki en yüksek (Maksimum) değeri döndürür.

ID numarası 6 olan sınavdaki en yüksek notu getiren sorgu:

```
SELECT
       MAX(Notu)
FROM
       Sinav_Ogrenci
WHERE
       Sinav_ID = 6
```

şeklindedir.

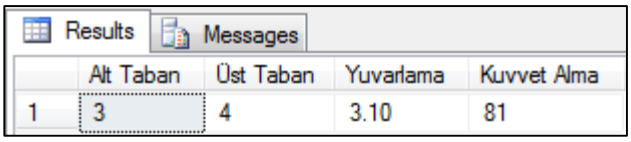
```
SELECT GETDATE()
```

Sorgusundaki GETDATE() fonksiyonu içinde bulunduğumuz tarih ve saati döndürür.

```
SELECT
      'Alt Taban' = CEILING(3.14),
      'Üst Taban' = FLOOR(3.14),
      'Yuvarlama' = ROUND(3.14, 1),
      'Kuvvet Alma' = POWER(3, 4)
```

Sorgusunun sonucu şekildeki gibi olur:

SQL Server açısından bakıldığında fonksiyonlar, view (görüntü) ve Store prosedür (Saklı Yordam) kullanmanın avantajlarını kendi bünyesinde toplamıştır.



Results | Messages

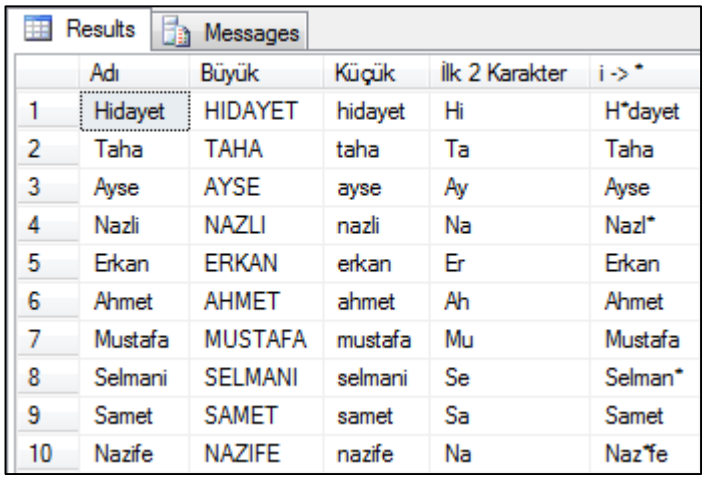
	Alt Taban	Üst Taban	Yuvarlama	Kuvvet Alma
1	3	4	3.10	81

Şekil 11.5. Yukarıdaki Çeşitli Hazır Matematiksel Fonksiyonların Kullanım Örneği

FLOOR() fonksiyonu parametre olarak aldığı kesirli sayının alt taban değerini, CEILING() fonksiyonu ise parametre olarak aldığı kesirli sayının üst taban değerini döndürür. ROUND() fonksiyonu iki parametre alır ve birinci parametredeki sayıyı, virgülden sonra ikinci parametredeki değer kadar basamak sayısı kalacak şekilde yuvarlama işlemi yapar. POWER() fonksiyonu da iki parametre alır ve ilk parametredeki sayının ikinci parametrede girilen dereceden kuvvetini (örneğimizde 3^4'ü) döndürür.

```
SELECT TOP 10
      'Adı' = Adi,
      'Büyük' = UPPER(Adi),
      'Küçük' = LOWER(Adi),
      'İlk 2 Karakter' = SUBSTRING(Adi, 1, 2),
      'i -> *' = REPLACE(Adi, 'i', '*')
FROM
      Birey
```

Sorgusunun sonucu aşağıdaki gibi olur:



Results | Messages

	Adı	Büyük	Küçük	İlk 2 Karakter	i -> *
1	Hidayet	HIDAYET	hidayet	Hi	H*dayet
2	Taha	TAHA	taha	Ta	Taha
3	Ayse	AYSE	ayse	Ay	Ayse
4	Nazli	NAZLI	nazli	Na	Nazl*
5	Erkan	ERKAN	erkan	Er	Erkan
6	Ahmet	AHMET	ahmet	Ah	Ahmet
7	Mustafa	MUSTAFA	mustafa	Mu	Mustafa
8	Selmani	SELMANI	selmani	Se	Selman*
9	Samet	SAMET	samet	Sa	Samet
10	Nazife	NAZIFE	nazife	Na	Naz*fe

Şekil 11.6. Yukarıdaki kod çalıştırıldıktan sonraki ekran görüntüsü

Fonksiyon içerisinde başka bir fonksiyon çağırırken iç içe fonksiyon çağrılmasına SQL Server 32. seviyeye kadar müsaade etmektedir.

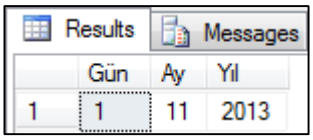
UPPER() fonksiyonu aldığı parametredeki ifadenin tüm harflerini büyük harfe dönüştürür, LOWER() fonksiyonu da aynı şekilde parametre olarak aldığı ifadenin tüm harflerini küçük harfe dönüştürür. SUBSTRING() fonksiyonu üç parametre alır, birinci parametrede üzerinde işlem yapılacak ifade yer alır, ikinci parametrede kaçıncı karakterden başlanacağı ve üçüncü parametrede de kaç karakter alınacağı bilgisi yer alır. Yukarıdaki örneğimizde öğrenci isimlerinin birinci karakterden başlayarak 2 karakteri alınmıştır. REPLACE() fonksiyonu da üç parametre alır: Birinci parametredeki ifade içerisinde yer alan ikinci parametredeki ifadeleri üçüncü parametrede yer alan ifade ile (örneğimizde Adı sütununda yer alan 'i' karakterlerini '*' karakteri ile) yer değiştirir.

İpucu: Bir fonksiyon başka bir fonksiyonu çağırabilir.

Not: Fonksiyon içerisinde başka bir fonksiyon çağırırken iç içe fonksiyon çağrılmasına SQL Server 32. seviyeye kadar müsaade etmektedir.

```
SELECT
      'Gün' = DAY(GETDATE()),
      'Ay'  = MONTH(GETDATE()),
      'Yıl' = YEAR(GETDATE())
```

Sorgusunun sonucu şekildeki gibidir:



Şekil 11.7. Yukarıdaki Kod Çalıştırıldıktan Sonraki Ekran Görüntüsü

DAY() fonksiyonu tarih formatında bir parametre alır ve aldığı tarih ifadesi içerisindeki gün bilgisini geri döndürür. Aynı şekilde MONTH() fonksiyonu ay bilgisini, YEAR() fonksiyonu da yıl bilgisini geri döndürür. Yukarıdaki örnekte DAY() fonksiyonuna parametre olarak GETDATE() fonksiyonunu vererek fonksiyon içinde fonksiyon çağırmış olduk.

Not: SP içerisinde fonksiyon çağrılabilmekte fakat fonksiyon içerisinde SP çağrılamamaktadır.

Kullanıcı Tanımlı Fonksiyon Oluşturma

Yukarıdaki hazır SQL fonksiyonları gibi biz de kendi ihtiyacımıza göre fonksiyon yazabiliriz. Fonksiyonlar geri dönüş tiplerine göre ikiye ayrılır: Tek bir değer döndüren skaler fonksiyonlar ve tablo geri döndüren fonksiyonlar.

Skaler fonksiyon

Skaler fonksiyon daha önce de bahsedildiği gibi geriye sadece tek *bir* değer döndürür. Yukarıda örnek olarak verdiğimiz hazır SQL fonksiyonlarının hepsi birer skaler fonksiyon örneğidir. Genel tanımı:

Skaler fonksiyon, geriye sadece tek *bir* değer döndürür.

```
CREATE FUNCTION [SahipAdi].[fonksiyon_Adi]
(
      @Parametre VERİTİPİ
)
RETURNS geri dönüş tipi
AS
BEGIN
      ...

      RETURN Geri dönüş değeri
END
```

şeklindedir.

Not: Fonksiyonlar en fazla 2100 parametre alabilir.

Örnek olarak girilen parametrenin karesini alan bir fonksiyon yazalım:

```
CREATE FUNCTION [dbo].[ufn_KareAl]
```

```
(
      @SayiFLOAT
)
RETURNS FLOAT
AS
BEGIN
      DECLARE @Sonuc  FLOAT
      SELECT
            @Sonuc = @Sayi * @Sayi
      RETURN @Sonuc
END
```

İpucu: Kendi tanımladığınız (oluşturduğunuz) fonksiyonları isimlendirirken 'ufn_' diye başlatmak genel bir alışkanlıktır ve kendi fonksiyonlarınızı sistem fonksiyonlarından ayırmak için de yardımcı olur.

Yukarıdaki fonksiyonda parametre olarak gelen @Sayi değişkenindeki değer kendisi ile çarpılarak elde edilen değer @Sonuc değişkenine atanmış ve @Sonuc değişkenindeki değer geri döndürülmüştür. Dönüş değeri değişken üzerinden geri döndürülmek zorunda değildir, ufn_KareAL() fonksiyonunu şu şekilde de tanımlanabilirdi:

```
CREATE  FUNCTION [dbo].[ufn_KareAl]
(
      @SayiFLOAT
)
RETURNS FLOAT
AS
BEGIN
      RETURN @Sayi * @Sayi
END
```

Kendi tanımladığınız fonksiyonları isimlendirirken 'ufn_' diye başlatmak genel bir alışkanlıktır ve kendi fonksiyonlarınızı sistem fonksiyonlarından ayırmak için de yardımcı olur.

Şimdi de yukarıda gördüğümüz hazır SQL fonksiyonlarından POWER() fonksiyonunu basit bir şekilde kendimiz yazalım:

```
CREATE  FUNCTION [dbo].[ufn_KuvvetAl]
(
      @Sayi      FLOAT,
      @Kuvvet    INT
)
RETURNS FLOAT
AS
BEGIN
      DECLARE @Sonuc  FLOAT = 1

      IF @Kuvvet = 0
            RETURN 1

      WHILE @Kuvvet > 0
      BEGİN
            SELECT @Sonuc = @Sonuc * @Sayi
            SELECT @Kuvvet = @Kuvvet -1
      END

      RETURN @Sonuc
END
```

Öğrencilerin bir dersten aldıkları ham başarı puanını hesaplayalım. Parametre olarak DersID ve BireyID alalım ve bu ders için öğrencinin girmiş olduğu tüm sınavlardan aldığı puan ile sınavın etki oranlarını çarpıp, toplayalım.

```
CREATE  FUNCTION [dbo].[ufn_HamBasariHesapla]
(
      @DersId    INT,
      @OgrenciId INT
)
RETURNS FLOAT
AS
BEGIN
      DECLARE @Sonuc  FLOAT = 0

      SELECT
            @Sonuc = SUM(SO.Notu * S.EtkiOrani / 100)
      FROM
            Sinav S
            INNER JOIN Sinav_Ogrenci SO
            ON S.ID = SO.Sinav_ID
      WHERE
            S.Ders_ID = @DersId
            AND SO.Ogrenci_ID = @OgrenciId

      RETURN @Sonuc
END
```

Tablo fonksiyon, geriye sadece *bir* tablo döndürür.

Tablo fonksiyon

Geriye *bir* tablo döndürür. Genel tanımı aşağıdaki gibidir:

```
CREATE  FUNCTION [SahipAdi].[FonksiyonAdi]
(
      @Parametre VERİTİPİ
)
RETURNS TABLE
AS
      [...]
      RETURN (SELECT ifadesi)
```

Bir derse ait sınav tanımlarını geri döndüren bir fonksiyon yazalım:

```
CREATE  FUNCTION [dbo].[ufn_SinavTanimListele]
(
      @DersId    INT
)
RETURNS TABLE
AS
      RETURN (
            SELECT
                  'Sınav Turu' = SinavTur,
                  'Etki Oranı' = EtkiOrani
            FROM
                  Sinav
            WHERE
                  Ders_ID = @DersId
            )
```

Bir bölümdeki öğrencilerin listesini geri döndüren bir fonksiyon yazalım:

```
CREATE  FUNCTION [dbo].[ufn_BirimOgrenciListele]
```

```
(
      @BirimId   INT
)
RETURNS TABLE
AS
      RETURN (
            SELECT
                  'Adı Soyadı' = B.AdiSoyadi,
                  'Öğrenci No' = OB.OgrenciNo
            FROM
            Birey B
                  INNER JOIN Ogrenci_Bolum OB
                  ON OB.Birey_ID = B.ID
            WHERE
                  OB.Birim_ID = @BirimId
            )
```

Aynı fonksiyon şu şekilde de tanımlanabilir:

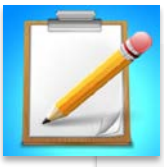
```
CREATE  FUNCTION [dbo].[ufn_BirimOgrenciListele]
(
      @BirimId   INT
)
RETURNS @Ogrenci TABLE
(
      AdiSoyadi  VARCHAR(50),
      OgrenciNo  VARCHAR(11)
)
AS
BEGIN
      INSERT INTO @Ogrenci
      SELECT
            'Adı Soyadı' = B.AdiSoyadi,
            'Öğrenci No' = OB.OgrenciNo
      FROM
      Birey B
            INNER JOIN Ogrenci_Bolum OB
            ON OB.Birey_ID = B.ID
      WHERE
            OB.Birim_ID = @BirimId
END
```

Yukarıdaki fonksiyonun bir select içerisinde kullanım örneği aşağıdaki gibidir:

```
SELECT
      *
FROM
dbo.ufn_BirimOgrenciListele(52)
```

Yukarıda birimId'si 52 olan birimdeki öğrencileri listeledik. Bu listede sadece ismi Ahmet olan öğrencileri sorgulamak istersek, sorguyu aşağıdaki gibi yazabiliriz:

```
SELECT
      *
FROM
dbo.ufn_BirimOgrenciListele(52)
WHERE
      [Adı Soyadı] like 'Ahmet%'
```



Bireysel Etkinlik

- Birey tablosunu temel tablo olarak kullanarak sadece bayan bireylerin adı ve soyadını getiren bir görüntü tanımlayınız.
- Bir önceki adımda tanımladığınız görüntüyü TC kimlik numaralarının ilk 9 hanesini içeren bir sütun daha ekleyerek yeniden tanımlayınız.
- Birey, Ogrenci_Bolum ve Birey_Iletisim tablolarını kullanarak kişilerin isim, soyisim, öğrenci numarası ve adreslerinden oluşan bir görüntü tanımlayınız.
- Bir önceki adımda oluşturduğunuz görüntüyü adres bilgisini içermeyecek şekilde güncelleyiniz.
- Oluşturduğunuz görüntüleri siliniz.
- Store prosedürlerin neden hızlı çalıştığını açıklayınız.
- Parametre olarak bölüm Id'sini alan ve geriye bu bölümde okuyan öğrencilerin isim, soyisim, öğrenci numaraları ve kayıt tarihi bilgilerini geri döndüren bir store prosedür yazınız.
- Bir önceki adımda oluşturduğunuz store prosedürü kişilerin adres bilgisini de içerecek şekilde güncelleyiniz.
- Birinci parametresinde BireyId değerini alan ve ilgili bireyin isim ve soyisimlerini ikinci ve üçüncü parametreler üzerinden geri döndüren bir store prosedür yazınız.
- İki önceki adımda oluşturduğunuz store prosüdür ile aynı işlevi gören bir fonksiyon yazınız.
- SQL Server in hazır fonksiyonlarını kullanarak Birey tablosundaki en uzun ismin içerdiği karakter sayısını getiren bir sorgu yazınız.
- Store prsedür ve fonksiyonların farklarını belirtiniz.

Özet

- **GÖRÜNTÜ (VIEW)**
- Görüntüler (view), select ifadeleri ile oluşturulmuş sanal tablolardır.
- Temel tablonun daraltılmış bir hâlidir. Sadece tanımda belirtilen koşulu sağlayan satırları ve yine tanımda belirtilmiş olan sütunları ve varsa hesaplatılmış sütunları içerir.
- Normal tablolar gibi select sorgularına, join ifadelerine dâhil edilebilir.
- Tablolarda olduğu gibi 'SELECT * FROM görüntü_adı' şeklinde sorgular çalıştırabiliyor olsak da gerçekte böyle bir tablo yoktur, görüntüler sadece kaydedilmiş sorgulardan ibarettir.
- Görüntü tanımlarken sütunlar için veri tipi belirtilmez, ilgili sütun temel tablo tanımında hangi veri tipi ile tanımlanmış ise görüntüde de aynı veri tipi ile bulunur.
- Görüntü tanımlamak için 'CREATE', tanımını değiştirmek için 'ALTER', görüntüyü silmek için 'DROP' komutu kullanılır.

- **STORE PROSEDÜR (STORED PROCEDURE)**
- Store prosedürler birçok programlama dilinde var olan fonksiyon yapılarına karşılık gelir.
- Store prosedür tanımı içinde yapacağımız değişiklik store prosedürün kullanıldığı her yere yansıyacağından dolayı kod yazmayı/programlamayı kolaylaştırmaktadır.
- Store prosedürler ilk defa çalıştırılırken ayrıştırma, derleme ve çalıştırma aşamalarının hepsi gerçeklenir, daha sonraki çağrılmalarında ise sadece üçüncü aşama olan çalıştırma (executing) aşaması gerçeklenir. Çünkü derleme aşamasından sonra oluşan çalışma planı hafızada (RAM) saklanır ve yeniden oluşturulmasına gerek kalmaz. Store prosedürler bu yüzden hızlı çalışır.
- Store prosedür tanımlamak için 'CREATE', tanımını değiştirmek için 'ALTER', silmek için 'DROP' komutu kullanılır.
- Store prosedürler parametre alabilir. Parametreler, girdi parametre ve çıktı parametre şeklinde ikiye ayrılır. Çıktı parametreler üzerinden prosedür dışına veri aktarılmış olur. Girdi parametreler için varsayılan değer de atanabilir. SP çağrılırken böyle tanımlanmış parametreler yazılmazsa varsayılan değer ile çağrılır.
- Return deyimi ile prosedür çağrıldığı yere veri döndürebilir.
- Return deyimi prosedürü sonlandırmak için de kullanılabilir. SP içerisinde yapılmak istenen işlemler veya gönderilen parametreler ile ilgili kontroller yapıldıktan sonra devam edilemeyeceği anlaşıldığında SP return deyimi ile sonlandırılır.

Özet (devamı)

- **FONKSİYON**
- SQL Server açısından bakıldığında fonksiyonlar view (görüntü) ve Store prosedür (Saklı Yordam) kullanmanın avantajlarını kendi bünyesinde toplamıştır.
- Fonksiyon içerisinde, store prosedür de olduğu gibi çeşitli T-SQL sorguları, if else yapıları vs. kullanılabilir ve fonksiyonlar da parametre alabilir. Fonksiyonun görüntüye benzer yönü ise select sorgularına dahil edilebilmeleridir.
- Skaler fonksiyon geriye sadece tek bir değer döndürür, tablo fonksiyon ise geriye bir tablo döndürür.
- Görüntü ve store prosedürlerde olduğu gibi fonksiyon tanımlamak için de 'CREATE', tanımını değiştirmek için 'ALTER', silmek için 'DROP' komutu kullanılır.
- Fonksiyon içerisinde başka bir fonksiyon (veya aynı fonksiyon) çağrılabilir. Bu şekilde içiçe fonksiyon çağırmaya SQL 32. Seviyeye kadar müsaade etmektedir. Rekürsif bir fonksiyon veya SP yazılacağı zaman bu duruma dikkat edilmelidir.
- SP içersinide fonksiyon çağrılabilir fakat fonksiyon içerisinde SP çağrılamaz.

DEĞERLENDİRME SORULARI

I. Bazı kullanıcıların sadece belirli sütunlara erişimine izin vermek
II. Kullanıcılara hesaplatılmış değer göstermek
III. Karmaşık sorguları basitleştirmek

1. Yukarıdakilerden hangisi veya hangileri görüntü kullanmanın sebeplerindendir?
   a) Yalnız I
   b) Yalnız II
   c) I ve II
   d) II ve III
   e) I, II ve III

I. Görüntü tanımlarken sütunların veri tipi belirtilmelidir.
II. Görüntü tanımlarken başka bir görüntü kullanılabilir.
III. Görüntüler tablolar ile birleştirilerek sorgulanabilir.

2. Görüntüler ile ilgili olarak yukarıdakilerden hangisi veya hangileri yanlıştır?
   a) Yalnız I
   b) Yalnız II
   c) Yalnız III
   d) II ve III
   e) I, II ve III

3. Görüntü tanımında yer alabilecek maksimum sütun sayısı kaçtır?
   a) 32
   b) 256
   c) 1024
   d) 2100
   f) 4096

4. Store Prosedürler (SP) ile ilgili olarak aşağıdakilerden hangisi yanlıştır?
   a) SP'ler parametre alabilir.
   b) SP'ler geriye bir değer döndürebilir.
   c) SP'ler geriye birden fazla tablo döndürebilir.
   d) SP'ler select sorgularına dâhil edilebilir.
   e) SP'ler başka bir SP'yi çağırabilir.

5. "Set Nocount on" komutu ne için kullanılır?
   a) İşlem(ler)den etkilenen kayıt sayısının belirlenmemesi için
   b) İşlem(ler)den etkilenen kayıt sayısının mesaj olarak döndürülmemesi için
   c) SP'deki işlem sayısının belirlenmemesi için
   d) Sistem tablolarında, SP'deki satır sayısı bilgisinin tutulmaması için
   e) SP'nin kaç defa çağrıldığı bilgisinin tutulmaması için

6. Çıktı parametre ile SP kullanmanın amacı aşağıdakilerden hangisidir?
   a) Geriye parametre üzerinden değer döndürmek
   b) Geriye bir veya daha fazla tablo döndürülmesine imkân tanımak
   c) Parametrenin varsayılan bir değer almasını sağlamak
   d) Girilen parametre için geçerlilik kontrolü yapmak
   e) Beklenmeyen bir durumda SP den çıkılmasını sağlamak

   I. Derleme
   II. Ayrıştırma
   III. Çalıştırma

7. SP'ler oluşturulurken yukarıdaki aşamalar hangi sıra ile gerçekleşir?
   a) III, II ve I
   b) III, I ve II
   c) II, I ve III
   d) II, III ve I
   e) I, III ve II

8. Hazır SQL fonksiyonlarından hangisinin açıklaması yanlış verilmiştir?
   a) MAX() parametre olarak aldığı sütundaki en yüksek değeri döndürür.
   b) GETDATE() aldığı parametredeki tarihin gün bilgisini döndürür.
   c) UPPER() aldığı parametredeki tüm karakterleri büyük harfe çevirir.
   d) ROUND() aldığı parametredeki sayısal ifadeyi yuvarlar.
   e) POWER() aldığı parametrelere göre kuvvet alma işlemi yapar.

9. SQL'de iç içe fonksiyon çağırmaya kaçıncı seviyeye kadar izin verilmektedir?
   a) 32
   b) 64
   c) 256
   d) 512
   e) 1024

10. Fonksiyon en fazla kaç parametre alabilir?
    a) 32
    b) 256
    c) 1024
    d) 2100
    e) 4096

Cevap Anahtarı

1.e, 2.a, 3.c, 4.d, 5.b, 6.a, 7.c, 8.b, 9.a, 10.d

YARARLANILAN KAYNAKLAR

Create User-defined Functions (Database Engine) 13.06.2019 tarihinde http://technet.microsoft.com/en-us/library/ms191320.aspx adresinden erişildi.

Görünümler 10.06.2019 tarihinde http://technet.microsoft.com/tr-tr/library/ms190174.aspx adresinden erişildi.

Gözüdeli, Y. (2012). Yazılımcılar için SQL Server 2008 R2 & Veritabanı Programlama, 5. Baskı, Ankara: Seçkin Yayıncılık

Henderson, K. (2013). SQL Server Depolanmış yordamları, XML ve HTML, 1. Baskı, İstanbul: Alfa Yayınları

Kullanıcı Tanımlı İşlevler Oluşturma (Veritabanı Altyapısı) 13.06.2019 tarihinde http://technet.microsoft.com/tr-tr/library/ms191320.aspx adresinden erişildi.

Maximum Capacity Specifications for SQL Server 13.03.2019 tarihinde https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-2017 adresinden erişildi.

Özhan, C. (2013). İleri Seviye T-SQL Programlama, 1. Baskı, İstanbul: KODLAB Yayınevi

Petkovic, D. (2006). Microsoft SQL Server, 1. Baskı, İstanbul: Alfa Yayınları

Saklı Yordamlar (Veritabanı Altyapısı) 11.06.2019 tarihinde http://technet.microsoft.com/tr-tr/library/ms190782.aspx adresinden erişildi.

Stored Procedures (Database Engine) 11.06.2019 tarihinde http://technet.microsoft.com/en-us/library/ms190782.aspx adresinden erişildi.

Views 10.06.2019 tarihinde http://technet.microsoft.com/en-us/library/ms190174.aspx adresinden erişildi.

VERİ TABANI YÖNETİMİ YAPMAK

İÇİNDEKİLER

- Yönetici paneli
- Veri tabanı işlemleri
  - Sorgulama işlemleri
  - İstemci istatistiği
  - Yürütme planları
- Veri tabanı sunucusunun komut satırından veritabanı işlemleri yapmak

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - SQL sunucu yönetim stüdyosunu kullanabilecek,
  - Veritabanı oluşturabilecek silebilecek ve özelliklerini değiştirebilecek,
  - Sorgu editörünün kullanımını, grafik arayüzünden ve sorgu editör penceresiyle sorgu tasarlanmasını ve çalıştırılmasını kavrayabilecek,
  - Sorguların çalışması sonrasında performansın nasıl analiz edileceğini, yürütme planlarının görüntülenmesi ve incelenmesini öğrenebilecek,
  - "Sqlcmd" komutlarının kullanımı hakkında bilgi sahibi olabileceksiniz.

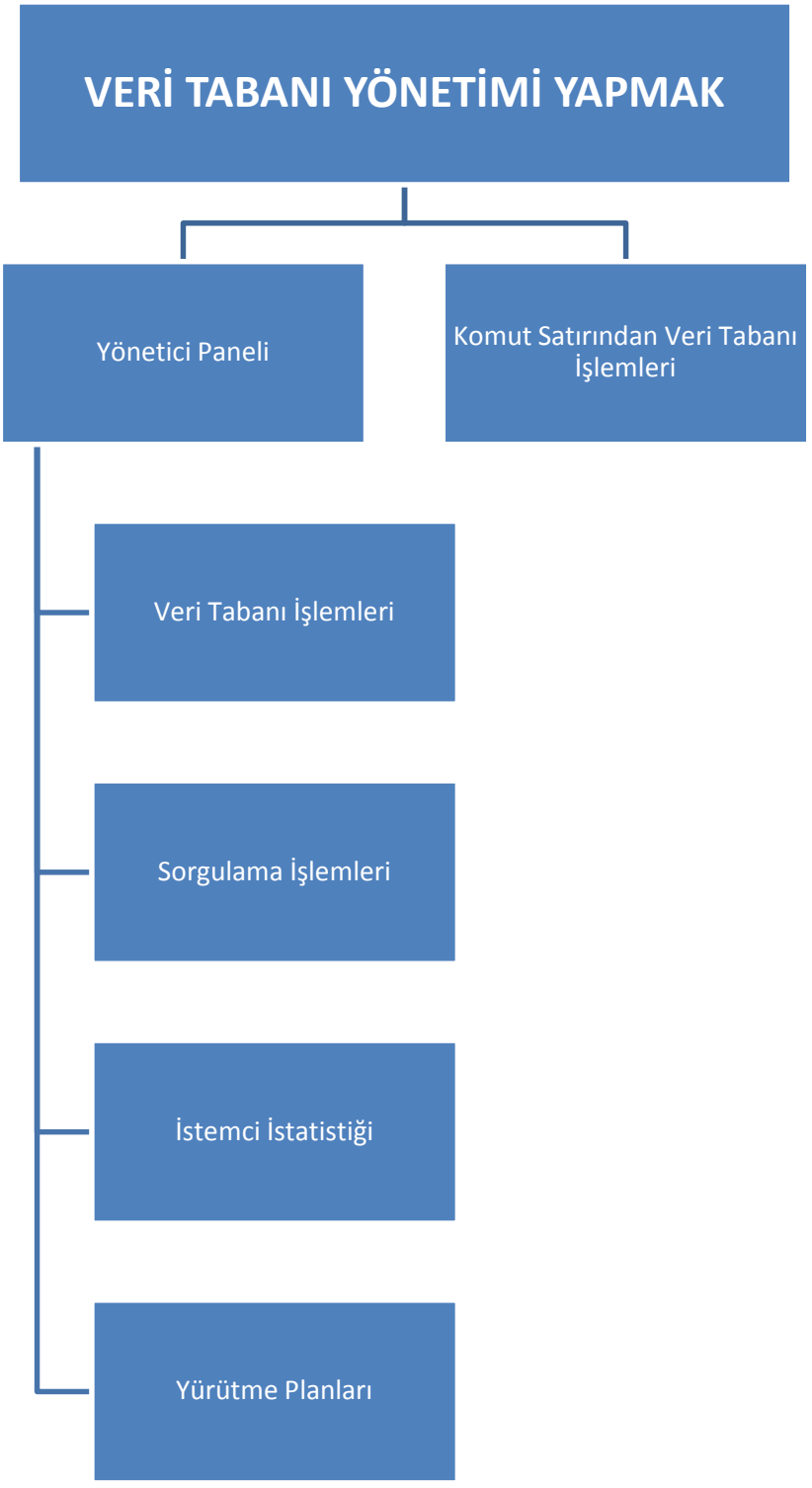
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi Ahmet Kamil KABAKUŞ

ÜNİTE

12


VERİ TABANI YÖNETİMİ YAPMAK
Yönetici Paneli
Komut Satırından Veri Tabanı İşlemleri
Veri Tabanı İşlemleri
Sorgulama İşlemleri
İstemci İstatistiği
Yürütme Planları

GİRİŞ

Önceki ünitelerde çeşitli sorgulamalar ve analizler yapmayı, görüntü tanımlamayı, saklı yordam (stored procedure) ve fonksiyon tanımlamayı kısaca SQL programlamayı işledik. SQL Sunucu Yönetim Stüdyosu (SQL Server Management Studio - SSMS) SQL Sunucu bileşenlerinin tamamına erişmeye, yönetmeye ve geliştirmeye yönelik tümleşik bir ortamdır. Yönetim Stüdyosu veri tabanı yöneticisi kişilere veri tabanı yönetimi işlemlerinde büyük kolaylık sağlamaktadır.

Bu ünitede öncelikle SQL Sunucu Yönetim Stüdyosu her bölümü ile ayrı ayrı tanıtılacaktır. Sihirbaz kullanılarak ve komut ekranından nasıl veri tabanı oluşturulabileceği anlatıldıktan sonra veri tabanının adının ("Rename") ve özelliklerinin ("Properties") nasıl değiştirilebileceği, veri tabanının nasıl güncellenebileceği ("Refresh") ve nasıl ortadan kaldırılabileceğine ("Delete") değinilecektir.

SQL Sunucu'da veri tabanı sorguları "Query Editor" ekranında manüel olarak T-SQL cümlecikleri ile yazılabileceği gibi "Design Query in Editor" ekranında sihirbaz kullanarak da oluşturulabilmektedir. Bu ünitede sorgu editörü ekranının (yeni sorgu sayfasının oluşturulması, sorgu sayfasının kaydedilmesi ve açılması vs.) ve sql editor araç çubuğunun kullanımı anlatılarak sorguların çalıştırılabilmesi için gerekli kısayol kombinasyonları öğretilecektir.

Bu ünitede ayrıca sorgu tasarım ekranı kullanılarak veri tabanı sorgularının nasıl tasarlanabileceği gösterildikten sonra sorgu performanslarının analiz edilmesini sağlayan istemci istatistikleri (client statistics) aracının kullanımı incelenecektir. Bu araç kullanılarak bir sorgu cümlesinde yapılan her bir değişiklik ile istemci istatistiklerinin nasıl değiştiği izlenebilmektedir.

Bu ünitede, sorgu iyileştirme (query optimizer) aracı tarafından hesaplanan ve bir sorgunun en ideal şekilde çalışması için önerilen yürütme planlarının nasıl oluşturulabileceği gösterilecektir. Yürütme planları içinde öncelikle tahmini yürütme planının nasıl görüntülenebileceği ve okunabileceği işlenecektir. Sonrasında ise gerçek yürütme planının ne olduğu açıklandıktan sonra nasıl görüntüleneceği ve detaylı olarak nasıl okunabileceği incelenecektir.

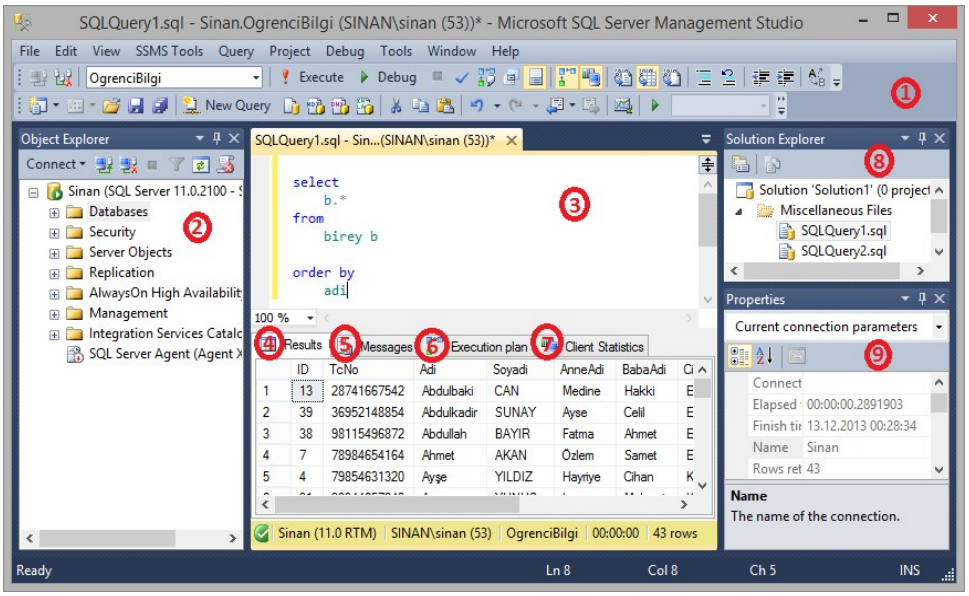
Tahmini yürütme planı adından da anlaşılacağı üzere sorgu çalıştırılmadan oluşturulan tahmini plandır. Gerçek yürütme planı ise SQL Sunucu tarafından sorgu çalıştırılırken oluşturulan plandır ve daha sonra kullanılmak üzere "plan ön belleğine" (plan cache) kaydedilir. Böylece sorgu her çalıştırıldığında tekrar yürütme planı oluşturulmamış olur. Mevcut yürütme planı kullanıldığı için de performans artışı sağlanmış olur.

Ünite kapsamında son olarak komut satırından veri tabanı işlemlerinin nasıl yapılabileceği anlatılarak go, out, error ve quit komutlarının kullanımlarına değinilecektir.

YÖNETİCİ PANELİ

SQL Sunucu Yönetim Stüdyosu (SQL Server Management Studio),

geliştiricilerin ve veri tabanı yöneticilerinin SQL Sunucu'ya erişimini sağlamak ve yönetmek için zengin grafiksel araçlar ve T-SQL komut yazım ortamını barındırır (Adar, 2012).



Şekil 12.1. Sql Sunucu Yönetim Stüdyosu

Yukarıdaki ekranın bölüm açıklamaları aşağıda verilmiştir:

- Bu bölüme hızlı erişim için araç düğmeleri eklenebilmektedir.
- "Object Explorer", veri tabanı nesnelerinin yönetildiği paneldir.
- Sorgu ekranı, T-SQL sorgularının yazıldığı ve çalıştırıldığı ekrandır.
- Sonuçlar ("Results"): Bu sekme, kod çalıştırıldıktan sonra açılmaktadır. Sorgu ile dönen tablolar listelenmektedir.
- Mesajlar ("Messages"): Sorgularla ilgili mesajlar bu sekmede görüntülenmektedir.
- Yürütme planı ("Execution plan"): Tahmini veya gerçek yürütme planını gösterir.
- İstemci istatistiği ("Client Statistics"): İstemci istatistik bilgileri görüntülenir.
- Proje yönetim ekranı ("Solution Explorer"): Yeni bir proje başlatıldığında veya var olan bir proje açıldığında görüntülenir.
- Özellikler ("Properties"): Bu ekranda bazı özellikler görüntülenir.

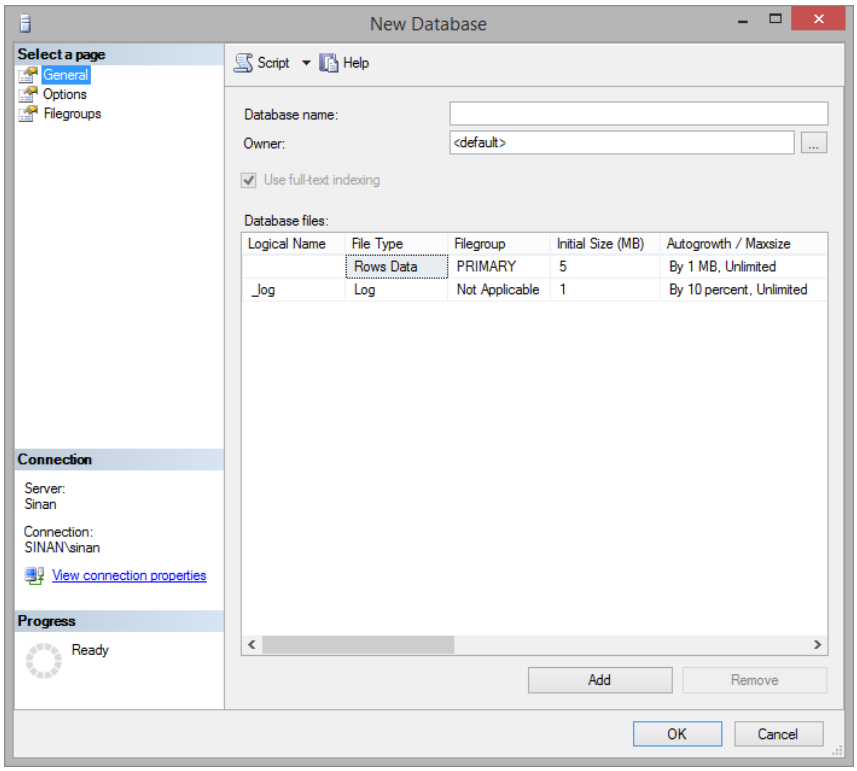
Yönetim Stüdyosundan kendi üzerine kurulmuş bir veri tabanı sunucusuna bağlanılmak istenildiğinde Server Name kısmına (local) yazılmalı ve kurulumdan sonra yapılan ilk çalıştırmalarda SQL Sunucu servislerinin başlaması beklenilmelidir.

SQL Sunucu Yönetim Stüdyosundan sa kullanıcı adı-şifre veya Windows kullanıcı adı- şifre ile giriş yapabilirsiniz.

Veri Tabanı İşlemleri

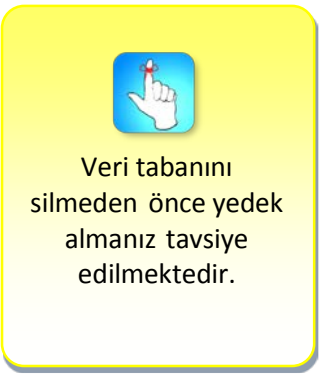
Veri tabanı oluşturmak

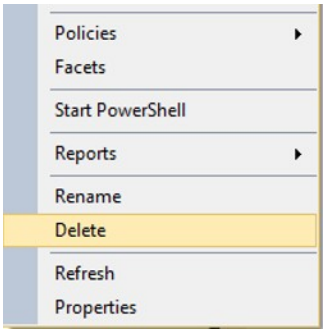
Öncelikle sihirbaz kullanarak bir veritabanının nasıl kolaylıkla oluşturulabileceğini gösterelim. "Object Explorer" penceresinde "Databases" klasörüne sağ tıklanarak veya var olan veri tabanı isimlerinden herhangi birine sağ tıklanarak açılan menüden "New Database" seçeneği seçilir. Veri tabanı oluşturmak için açılan bu pencerede (Şekil 12.2) veri tabanı ismi ("database name"), veri tabanı sahibi ("owner") ve diğer özellikler belirlendikten sonra tamam ("OK") tuşuna basılarak veri tabanı oluşturulabilmektedir.



Şekil 12.2. Yeni Veri Tabanı Oluşturulması

Oluşturulan veri tabanı, "Object Explorer" penceresinde görüntülenmektedir. Bunun için "Databases" klasörü seçiliyken güncelleme ("Refresh") tuşuna basılması veya sağ tıklandığında açılan menüden "Refresh" seçilmesi gereklidir. Bu menü altından ayrıca, veri tabanının adının ("Rename") ve özelliklerinin ("Properties") değiştirilebilmesi ve veri tabanının ortadan kaldırılması ("Delete") mümkündür (Şekil 12.3). "Delete" seçeneği verilerin silinmesi için kullanılan "delete from" komutu ile karıştırılmamalıdır. "Delete from" komutu ilgili tablodaki kayıtların silinmesi için kullanıldığını hatırlatalım.

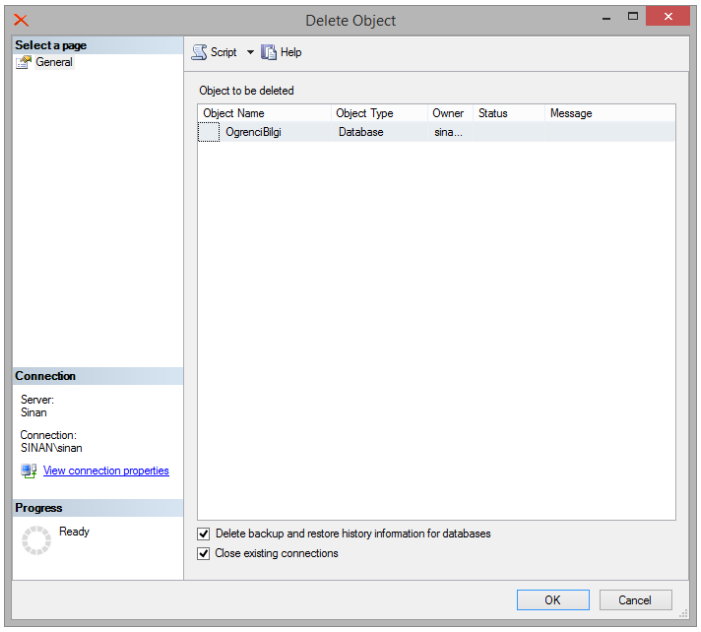




Şekil 12.3. Veri Tabanın Silinmesi

Veri tabanını silmek yani fiziksel olarak ortadan kaldırmak istediğinizde veri tabanına bağlı olan ve herhangi bir işlem yürüten bir kullanıcı olması durumunda veri tabanı ortadan kaldırılamayacaktır. Bunun için silme onay penceresinde, "Close existing connections" (var olan bağlantıları kapat) seçeneği aktif yapılmalıdır (Şekil 12.4). Bu onay kutucuğunun varsayılan olarak pasif olması, aktif olarak kullanılan bir veritabanının yanlışlıkla silinmesinin önüne geçmek içindir.

Veri tabanı ortadan kaldırılırken veri tabanı yedekleme ve geri yükleme geçmişini silmek için de “Delete backup and restore history information for databases” seçeneği aktif edilmelidir (Şekil 12.4).

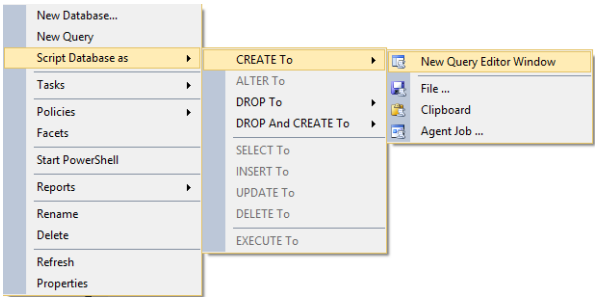


Şekil 12.4. Veri Tabanın Silinmesinin Onaylanması

Veri tabanının oluşturulması ve ortadan kaldırılması, sihirbaz kullanılarak yapılabildiği gibi kod penceresinden (“Query Editor”) de yapılabilmektedir (Şekil 12.5). İlgili TSQL kodları kullanıcı tarafından manüel olarak yazılabileceği gibi mevcut veri tabanının silinmesi (drop) ve yeni bir veri tabanının oluşturulması (create) için gerekli TSQL kodlarına, ilgili veri tabanı ismine sağ tıklandığında açılan menüden “Script database as” seçeneği altından erişilebilmektedir.

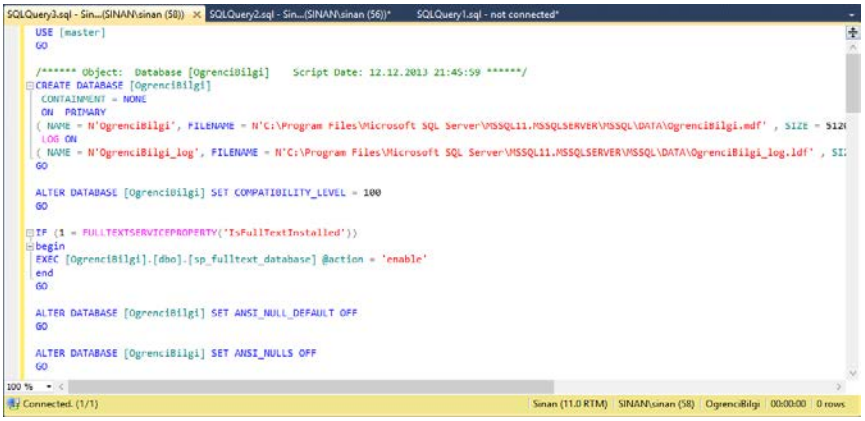
Veri tabanının oluşturulması ve ortadan kaldırılması, sihirbaz kullanılarak yapılabildiği gibi kod penceresinden (“Query Editor”) de yapılabilmektedir.

“Create To” menüsü tıklandığında yeni veri tabanının oluşturulması için gerekli kodlar, “Drop To” menüsü tıklandığında tabanının silinmesi için gerekli kodlar ve “Drop And Create To” menüsü tıklandığında ise ilgili veri tabanının ortadan kaldırılması ve yeniden oluşturulması için gerekli kodlar şablon hâlinde otomatik olarak oluşturulabilmektedir. Bu kodların, sorgu penceresinde görüntülenmesi için “New Query Editor Window”, dosyaya kaydedilebilmesi için “File”, kodların panoya aktarılabilmesi için “Clipboard” ve ilgili kodların zamanlanmış görev tanımının yapılabilmesi içinse “Agent Job” seçeneği seçilebilmektedir.



Şekil 12.5. Veri tabanı oluşturma sorgusunun, sorgu editörü penceresine aktarılması

İlgili veri tabanı üzerinde Şekil 12.5.teki gibi “Create To” ve “New Query Editor Window” uygulandığında (örnekte “OgrenciBilgi” veri tabanı üzerindeyken bu işlemler yapılmış) Şekil 12.6.daki gibi TSQL kodları otomatik olarak oluşturulmaktadır. Mevcut veri tabanı ortadan kaldırılmadan veya ismi değiştirilmeden bu kod olduğu gibi çalıştırılırsa “Database 'OgrenciBilgi' already exists. Choose a different database name.” yani “OgrenciBilgi veri tabanı mevcuttur. Başka bir veri tabanı adı seçin” şeklinde bir hata mesajı görüntülenecektir.



Şekil 12.6. Veri Tabanı Oluşturma Sorgusu, Sorgu Editörü Penceresinde

Sorgulama İşlemleri

Sorgulama işlemleri SQL Sunucu’da iki şekilde yapılabilmektedir. “Query Editor” ekranında manüel olarak T-SQL cümlecikleri yazılabileceği gibi “Design Query in Editor” ekranında sihirbaz kullanarak da sorgular oluşturulabilmektedir.

Sorgu editörü (Query Editor)

Veri tabanı aktif iken “Ctrl+N” tuş kombinasyonu ile veya standart araç çubuğunda bulunan “New Query” düğmesine tıklandığında sorgu editörü açılır ve bu ekranda aktif veri tabanı için sorgular yazılabilir (Gözüdeli, 2010).

Sorgu editörü ekranı, T-SQL sorgularının yazıldığı ve çalıştırıldığı ekrandır. Sorgunun çalıştırılması için, sorgu ekranı aktif iken “Alt+X” tuş kombinasyonu ve “Ctrl+E” tuş kombinasyonu kullanılabilir veya F5 tuşuna basılabilir. Sorgu komutlarının çalıştırılması için ayrıca araç düğmeleri içinde bulunan “Execute” (çalıştır) düğmesine de tıklanabilmektedir. Eğer sorgu editörü ekranında bulunan tüm sorgular değil de belirli bir kod bloğunun çalıştırılması istenirse ilgili kodlar seçildikten sonra yukarıda saydığımız yöntemlerden biri kullanılabilir.

Sorgu çalıştırıldıktan sonra dönen sonuçlar, “results” (sonuçlar) sekmesinde görüntülenmektedir. “Results” sekmesinin altında bulunan durum çubuğunda ise yürütme işleminin ne kadar sürede tamamlandığı ve sorgu sonunda kaç satır döndüğü bilgisi görüntülenmektedir.

Sorgu editöründe T-SQL kodları yazılırken SQL Sunucu yazım denetimi yapar ve yazım kurallarına uymayan ifadelerin altını çizer. SQL Sunucu kod yazımına ipuçları ile yardımcı olması için de “intellisense” (otomatik tamamlama) seçeneğinin aktif olması gerekmektedir.

Sorgu editörüne dosyadan veya panodan yapıştırma işlemi ile toplu sorgu gerçekleştirebilirsiniz.

SQL Sunucu Yönetim Stüdyosunda komutların otomatik olarak tamamlanması için “Query” sekmesinde “IntelliSense Enabled” seçili olmalıdır.

Sorgu editöründe yazılan kodların ".sql" uzantısıyla bilgisayara kaydedilmesi için "Ctrl+S" tuş kombinasyonu kullanılabilir. Kayıtlı dosyaları açılması için "Ctrl+O", yeni sorgu sayfası oluşturulması için de "Ctrl+N" uygulanabilir.

Sorgu yazımı sırasında sık kullanılan komutlar "SQL Editor" araç çubuğunda bir araya getirilmiştir. Bu araç çubuğu (Şekil 12.7) üzerindeki düğmeleri ve işlevlerini tek tek açıklayalım:

Şekil 12.7. "SQL Editor" Araç Çubuğu

Yukarıdaki şekilde SQL Editor araç çubuğu üzerindeki düğmeler numaralandırılarak gösterilmektedir. Şekilde numaralandırıldığı sıra ile araç düğmelerinin açıklamaları ise aşağıdaki gibidir:

- Sunucuya bağlanmak için kullanılır. Ayrıca "File" menüsü altındaki "Connect Object Explorer" seçeneği ile de bağlanılabilir.
- Aktif sunucu bağlantısının koparılmasını sağlar.
- Sorgu ekranında yazılan kodların hangi veri tabanı üzerinde çalıştırılacağını belirler. Kutucuk içinde, üzerinde çalışılan veri tabanı adı görüntülenmektedir. Yanındaki oka tıklandıktan sonra veri tabanı değiştirilebileceği gibi bunun için "Ctrl+U" tuş kombinasyonu da kullanılabilir.
- T-SQL ifadesinin çalıştırılmasını ("execute") sağlar.
- T-SQL ifadesinin hatalarının ayıklanması ("debug") için kodun, adım adım çalıştırılmasını sağlar.
- Çalışan bir T-SQL ifadesinin iptal edilmesini sağlar. Her hangi bir sorgu çalışırken aktif olur. Yanlışlıkla çalıştırılan bir sorgu bu şekilde iptal edilirse sorgu hiç çalıştırılmamış gibi olur. Örneğin, büyük bir tablodaki tüm kayıtları silecek sorgunun yanlışlıkla çalıştırıldığını varsayalım. Bu durumda sorgu henüz bitmeden iptal edilirse hiçbir kayıt silinmemiş olur.
- T-SQL ifadesinin, yazım kurallarına uygunluğunu test eder ("parse").
- Display Estimated Execution Plan: T-SQL ifadesinin en uygun yürütme yöntemini hesaplayarak bir yürütme planı oluşturmaktadır.
- Query Option: Aktif sorgu penceresinin özellerini değiştirmek için kullanılır.
- Intellisense: otomatik tamamlama özelliği aktif veya pasif yapılabilmektedir.
- Include Actual Execution Plan: T-SQL ifadesi çalıştırıldığında gerçek yürütme planının da görüntülenmesi sağlanır.
- Include Client Statistics: sorgu istatistiği görüntülenir.
- Result to Text: T-SQL sorgu sonucunun "text" (düz yazı) formatına aktarılmasını sağlar.
- Result to Grid: T-SQL sorgu sonucunun tablo (grid) şeklinde

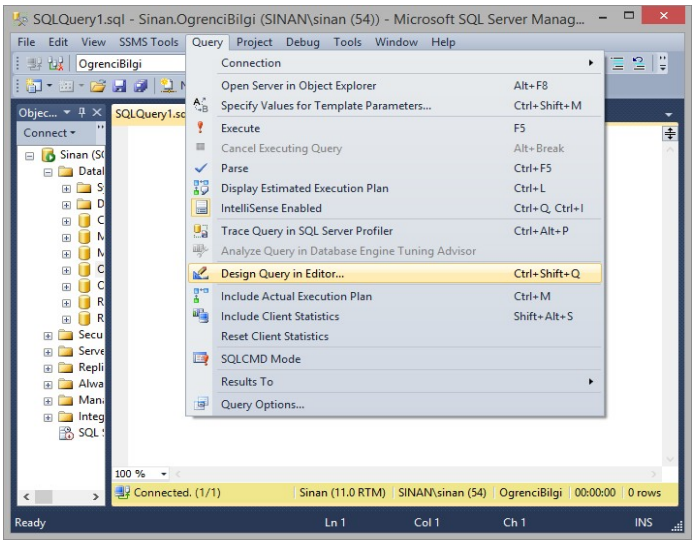
görüntülenmesini sağlar.

- Result to File: T-SQL sorgu sonucunun ".rpt" uzantılı bir dosyaya kaydedilmesini sağlar.
- Comment out Selected Lines: Seçilen satırların açıklama satırlarına dönüştürülmesi sağlanır. T-SQL'de bir satırda "—" (iki tire)'den sonra gelen ifadeler ve kod içinde "/*" ile "*/" deyimleri arasındaki ifadeler açıklama ifadeleri olarak kabul edilmektedir. Eğer tek satır açıklama satırı olacaksa
- "--" kullanılırke ardışık birden fazla satır açıklama satırına dönüştürülecekse "/*" ile "*/" ikilisi kullanılabilir.
- Uncomment Selected Lines: Açıklama satırı yapılan ifadeleri geri almak için kullanılır. Yani eğer satırın başında iki tire ("--") varsa bu iki tire silinir.
- Increase Indent (girintiyi artır): Satır başı girintisini artırmaya yarar.
- Decrease Indent (girintiyi azalt): Satır başı girintisini azaltmaya yarar.

Editörde sorgu tasarımı (Design query in editor)

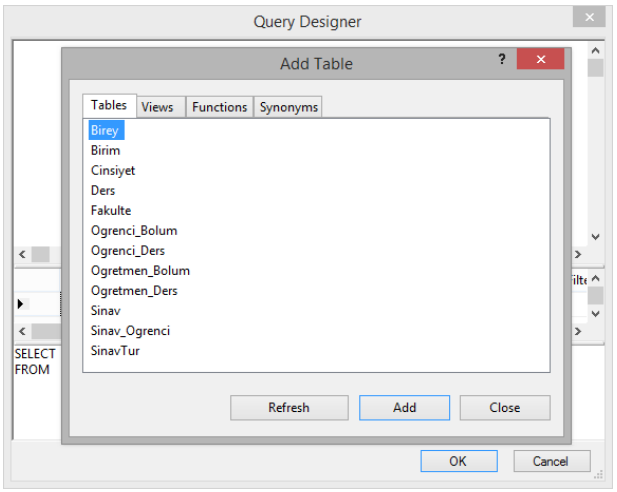
Sorgu tasarımı ekranı, grafik kullanıcı arayüzü (GUI- Graphical User Interface) olduğu için bu ekran ile çoklu tablo veya görünüm (view) ile karmaşık sorgular rahatlıkla oluşturulabilmektedir. Sorgu tasarımı ekranına geçmek için "Query" menüsünden (Şekil 12.8) veya "Query Editor" ekranında sağ tıklandığında açılan menüden "Design Query in Editor" seçilebilir veya "Ctrl+Shift+Q" tuş kombinasyonu kullanılabilir.

Editörde sorgu tasarımı açmak için Ctrl+Shift+Q tuş kombinasyonunu kullanabilirsiniz.



Şekil 12.8. Sorgu Tasarımı Ekranının Açılması

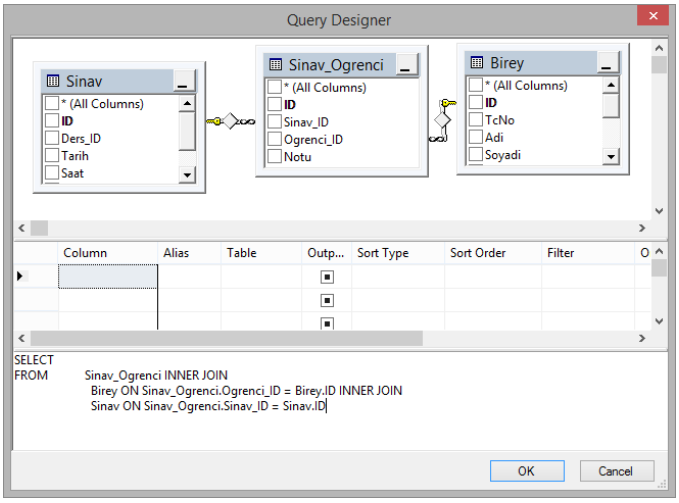
Yukarıda sayılan seçeneklerden herhangi biri uygulandığında Şekil 12.9.daki gibi Sorgu Tasarımı sihirbazı açılacaktır.



Şekil 12.9. Sorgu Tasarımı Ekranında, Sorguya Tablo Eklenmesi

Sorgu tasarımı ekranında öncelikle sorguda kullanılmak üzere tablo (table), görünüm (view), fonksiyon (function) veya sinonim (synonym) eklenebilmektedir. "Add Table" (tablo ekle) ekranında mevcut tablo isimlerinden birine çift tıklanarak veya ilgili tablo ismi seçildikten sonra ekranın sağ altında bulunan "Add" (ekle) düğmesine tıklanarak sorguda kullanılmak üzere tablolar hafızaya alınmaktadır. Böylece ilgili tablolara erişmek ve kayıtları sorgulamak için gerekli TSQL kodları otomatik olarak oluşturulmaktadır.

Örnek olması için "Sinav", "Sinav_Ogrenci" ve "Birey" tabloları yukarıda anlatıldığı gibi seçildiğinde ekran görünümü Şekil 12.10.daki gibi olmaktadır.



Şekil 12.10. Sorgu Tasarımı Ekranının Tablo Eklendikten Sonraki Görüntüsü

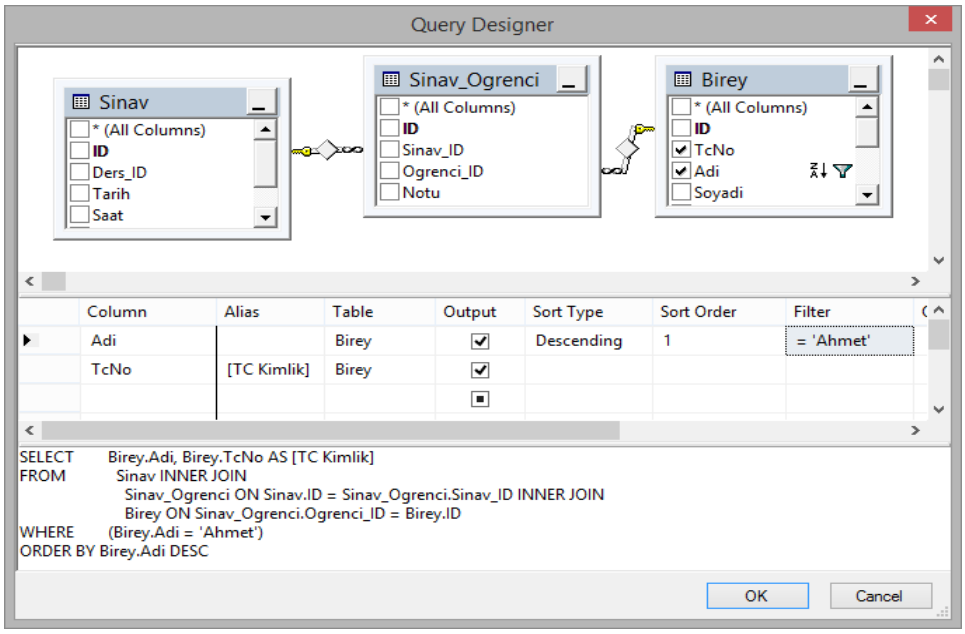
Diyagram sekmesi seçilen veri tabanı nesneleri ve aralarındaki ilişkileri göstermektedir.

Ekranın üst bölmesinde diyagram (diagram), orta bölmesinde kriter (criteria) ve alt bölmesinde SQL sekmesi bulunmaktadır. Diyagram sekmesinde, seçilen veri tabanı nesneleri (tablo, görünümi, fonksiyon ve sinonim) ve aralarındaki ilişkiler (bire bir, bire çok, çoka çok) gösterilmektedir. Diyagram sekmesinde ayrıca ilgili tablolardaki kolon adları yanında bulunan kutucuklar işaretlenerek sorgu ekranına aktarılması ve dolayısıyla sorgu sonucunda görüntülenmesi sağlanmış olur.

Diyagram sekmesinde seçilen kolon adları ve bilgileri otomatik olarak kriter

(criteria) sekmesine ve SQL sekmesine düzenlenmektedir. Sorguya dâhil edilecek kolon ile ilgili bilgiler, kriter sekmesinde de manüel olarak yazılabilmektedir. Bu sekmede kolon adı "Column" sütununa yazıldıktan sonra, görüntülenmesi istenen kolon adına verilecek takma ad, "Alias" sütununa, kolonun bulunduğu tablo adı "Table" sütununa yazılır. Kolonun görüntülenip görüntülenmeyeceği "Output" sütununda, kolon adının sıralamaya dâhil edilip edilmeyeceği "Sort Order" sütununda ve ilgili kolon sıralamaya dâhil edilecekse sıralama türü "Sort Type" sütununda tanımlanabilmektedir. Son olarak "Filter" (filtreleme) sütunundan ilgili kolona filtre tanımlanabilmektedir.

Örneğimizde, "Birey" tablosundaki "Adi" ve "TcNo" alanları seçilerek "TcNo" alanına "TC Kimlik" takma adı verilmektedir (Şekil 12.11). "Adi", "Ahmet" olan kayıtların görüntülenebilmesi için "Filter" alanının nasıl düzenlendiğine dikkat ediniz. Kayıtların, "Adi" alanına göre azalan şekilde sıralanması için de "Sort Order" sütun değeri 1 yapılarak "Sort Type" değeri "Descending" (azalan) olarak belirlenmiştir.



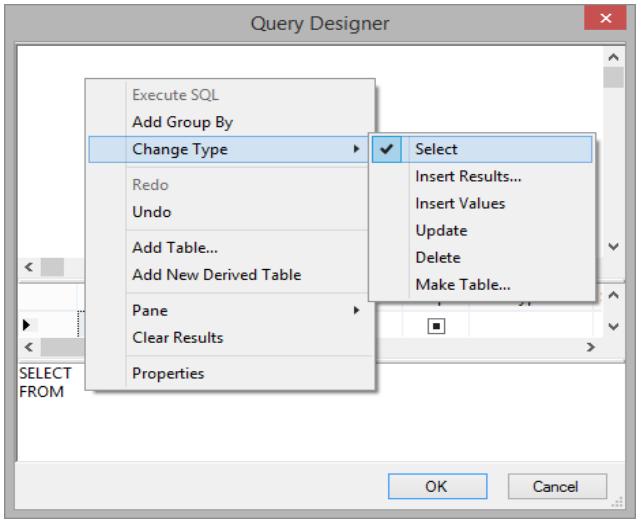
Şekil 12.11. Sorgu Tasarımı Ekranından Kriterlerin Tanımlanması

Kriter(criteria) bölmesindeki düzenlemeler otomatik olarak SQL sekmesine aktarılır.

Kriter (criteria) bölmesindeki düzenlemeler de otomatik olarak SQL sekmesine aktarılmaktadır. Sorgudaki gerekli düzenlemeler, SQL sekmesinden de manüel olarak yapılabilmektedir. Sorgu tasarımı bittikten sonra da SQL dil kurallarına uygunluğu, SQL sekmesine sağ tıklandığında açılan menüden "Verify SQL Syntax" yani "SQL Sözdizimini Doğrula"seçilerek test edilebilir.

Sorgu tasarımı ekranında, tasarlanacak olan sorgu türünü değiştirmek için, sorgu ekranında sağ tıklanarak "Change type" seçilir (Şekil 12.12). Burada birinci seçenek olan "Select" tipinde tasarım anlatıldı. "Insert Results…" seçildiğinde ise sorgu sonucu bir tabloya eklenmektedir. "Insert Values" seçeneğinde ise tabloya girilecek değerler tek tek belirtilmektedir. Dolayısıyla bu seçenekte bir tablo üzerinde işlem yapılabilmektedir. "Update" seçeneği ile tablodaki verilerin

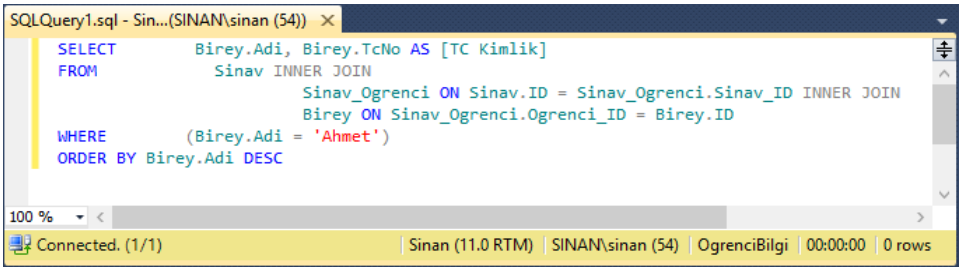
güncellenebileceği sorgu oluşturulabilirken "Delete" seçeneği ile de tablodaki verilerin silinebilmesi için sorgu oluşturulabilmektedir. Son olarak "Make Table…" seçeneği ile sorgu sonucu bir tabloya dönüştürülmektedir. Yani "Select… Into… " yapısında TSQL kodu oluşturulmaktadır.



Şekil 12.12. Sorgu Tasarımı Ekranında Sorgu Tipinin Değiştirilmesi

Şekil 12.11.deki gibi sorgu tasarımı yapıldıktan sonra Sorgu Tasarımı ekranında tamam (OK) düğmesine tıklanarak pencere kapatılır. Tasarlanan TSQL sorgusu, otomatik olarak sorgu editörüne aktarılacaktır (Şekil 12.13). Artık bu ekranda sorgu yeniden düzenlenebileceği gibi, sorgunun çalıştırılması ve hata ayıklanması gibi işlemler yapılabilecektir.

Sunucu üzerinde oluşan yüklerin analizi ve performansı için istemci istatistiği kullanılır.



Şekil 12.13. Sorgu Tasarımı Ekranında Oluşturulan Sorgunun, Sorgu Editörüne Aktarılması

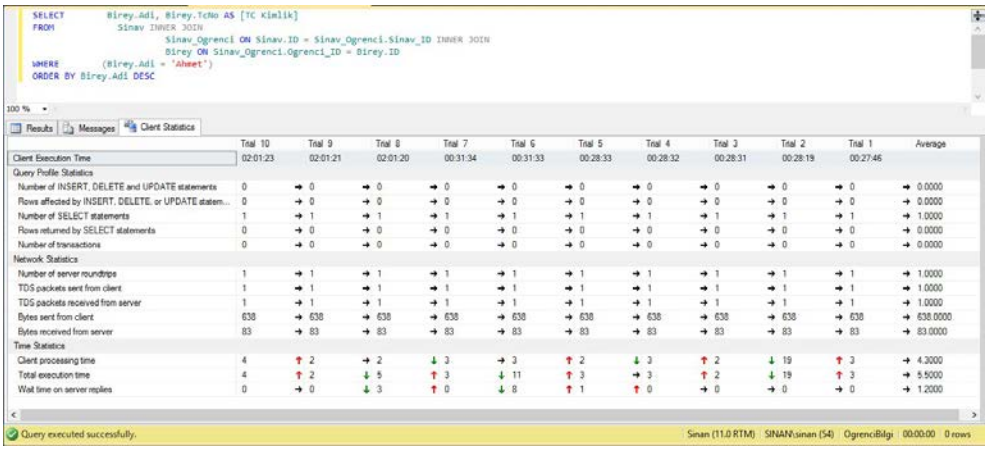
İstemci İstatistiği

İstemci istatistikleri (client statistics), SQL Sunucu'da çalıştırılan sorguyla ilgili performans bilgilerini vermektedir. Böylece bir işlemi yapmak için yazılan farklı T-SQL kodlarından hangisinin daha performanslı olduğu incelenebilmektedir (Elbahadır, 2012).

İstemci istatistiği seçeneği açıkken aynı sorgu her çalıştırıldığında yürütme süresi tekrar hesaplanmakta ve farklı sonuçlar bulabilmektedir. Bunun sebebi, işlemci yük durumu ve ağ trafiğinin yoğunluğudur. Şekil 12.14.teki örnekte bir sorgu ardışık olarak 10 defa çalıştırılmış ve zaman istatistiğinde farklılık gözlenmiştir. İstatistiki değerin artması yukarı doğru kırmızı ok ile gösterilirken azalması ise aşağı doğru yeşil ok ile gösterilmektedir. Değerin aynı kalması sağa doğru siyah ok ile işaret edilmektedir.

Bireysel Etkinlik

- Yazdığınız sorgunun 20 defa çalıştırılması ile oluşan istemci istatistikleri ile iki farklı sorgunun çalıştırılması sonucunda oluşan istatistikleri karşılaştırınız.



Şekil 12.14. Bir Sorgunun 10 Defa Çalıştırılması Ile Oluşturulan Istemci Istatistikleri

Total Execution Time sunucu üzerinde sorgunun toplam yürütme süresini gösterir.

Yukarıdaki örnekte istemci istatistiği en son çalıştırılan 10 sorgu için istatistik çıkarmaktadır. Her deneme (trial) bir sütun ile temsil edilirken en sağdaki sütunda ortalama (average) değerler görüntülenmektedir. En soldaki sütunda ise değişken isimleri yer almaktadır. Bu değişkenlerin anlamları ise şöyledir:

- "Client Execution Time": İstemci yürütme süresini saniye cinsinden gösterir.
- "Query Profile Statistics": Sorgu görüntü istatistiği grubudur.
- "Number of INSERT, DELETE and UPDATE statements": Sorgu içinde geçen "INSERT", "DELETE" ve "UPDATE" cümleciklerinin sayısını gösterir.
- "Rows affected by INSERT, DELETE or UPDATE statements": "INSERT", "DELETE" veya "UPDATE" işlemlerinde etkilenen satır sayısını gösterir.
- "Number of SELECT statements": SELECT cümleciği sayısını gösterir.
- "Number of transactions": İşlem sayısını gösterir.
- "Network Statistics": Ağ istatistiği grubudur.
- "Number of server roundtrips": Sunucuya gidiş geliş sayısını gösterir.
- "TDS packets sent from client": İstemciden gönderilen TDS paket sayısını gösterir.
- "TDS packets received from server": Sunucudan alınan TDS paket sayısını gösterir.
- "Bytes sent from clients": İstemciden gönderilen byte sayısını gösterir.
- "Bytes received from server": Sunucudan alınan byte sayısını gösterir.

- "Time Statistics": Süre istatistikleri grubudur.
- "Total execution time": Toplam yürütme süresini gösterir.
- "Wait time on server replies": Sunucu cevaplarının bekleme süresini gösterir.

Şimdi basit bir örnek ile sorgu performanslarını test edelim. "Birey" ve "Ogrenci_Bolum" tablolarını önce klasik birleştirme yöntemiyle sorgulayalım. Daha sonra dâhili birleştirme (inner join) kullanarak sorguyu yeniden yazıp çalıştıralım. İki sorgunun istemci istatistik değerleri Şekil 12.15.teki gibidir. Dâhili birleştirmenin klasik birleştirmeden daha yavaş çalıştığı, istemci tarafından daha fazla bilgi gönderildiği ve diğer bilgilerin ise benzer olduğu gözlenebilmektedir.

Results | Messages | Client Statistics

	Trial 2	Trial 1	Average
Client Execution Time	02:50:25	02:49:56	
Query Profile Statistics			
Number of INSERT, DELETE and UPDATE statements	0 →	0 →	0.0000
Rows affected by INSERT, DELETE, or UPDATE statem...	0 →	0 →	0.0000
Number of SELECT statements	1 →	1 →	1.0000
Rows returned by SELECT statements	8 →	8 →	8.0000
Number of transactions	0 →	0 →	0.0000
Network Statistics			
Number of server roundtrips	1 →	1 →	1.0000
TDS packets sent from client	1 →	1 →	1.0000
TDS packets received from server	1 →	1 →	1.0000
Bytes sent from client	194 ↑	182 →	188.0000
Bytes received from server	1350 →	1350 →	1350.0000
Time Statistics			
Client processing time	3 ↓	4 →	3.5000
Total execution time	76 ↑	5 →	40.5000
Wait time on server replies	73 ↑	1 →	37.0000

Şekil 12.15. İki Farklı Sorgunun Çalıştırılması Ile Oluşturulan Istemci Istatistikleri

Yürütme Planları

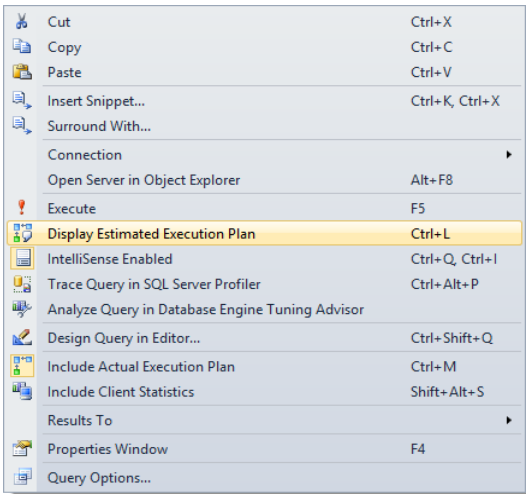
Yürütme planı, sorgu iyileştirme (query optimizer) aracı tarafından hesaplanan ve bir sorgunun en ideal şekilde çalışması için önerilen yoldur. Diğer bir ifadeyle yürütme planı, sorgunun nasıl çalışacağını göstermektedir. Uzun zaman alan sorguların yürütme planları incelenerek sorgular iyileştirilebilmektedir.

Yürütme planı, sorgu iyileştirme (query optimizer) aracı tarafından hesaplanan ve bir sorgunun en ideal şekilde çalışması için önerilen yoldur.

Tahmini yürütme planı ve gerçek yürütme planı olmak üzere iki tür yürütme planı bulunmaktadır.

Tahmini yürütme planı (Estimated execution plan)

Tahmini yürütme planı (Estimated execution plan) adından da anlaşılacağı üzere sorgu çalıştırılmadan oluşturulan tahmini plandır. Tahmini yürütme planını görüntülemek için "SQL Editor" araç çubuğunda bulunan "Display Estimated Execution Plan" (Tahmini Yürütme Planını Görüntüle) düğmesine tıklanabilir, sorgu editöründe sağ tıklandığında açılan menüde "Display Estimated Execution Plan" seçilebilir (Şekil 12.16) veya klavyeden "Ctrl+L" tuş kombinasyonu kullanılabilir.

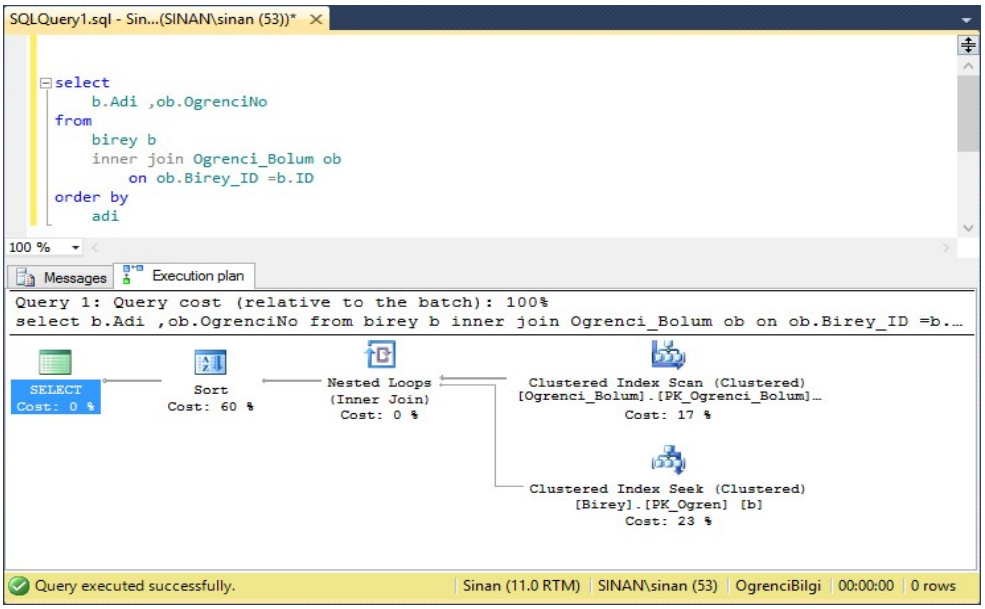


Şekil 12.16. Tahmini Yürütme Planının Açılması

Sorgunun tahmini yürütme planı ekranın alt kısmında "Execution plan" sekmesinde Şekil 12.17.deki gibi görüntülenecektir. Bu sekmede, veriye erişilirken dizin (index) kullanılıp kullanılmadığı, kullanıldıysa hangi dizinlerin kullanıldığı gibi bilgiler görüntülenmektedir. Ayrıca her bir işlemin, toplam yürütme süresinin yüzde kaçını temsil ettiği de bu sekmede görüntülenen bilgiler arasındadır.

Yürütme planları okunurken sağdan sola doğru okunmaktadır.

Yürütme planları okunurken de sağdan sola doğru okunmaktadır. Bu okuma düzeni, yapılan işlemlerin sırası dolayısıyla böyledir. Örnek sorgumuz çalıştırılırken öncelikle kümelenmiş dizinler (clustered index) kullanılarak dâhili birleştirme (inner join) işleminin yapıldığı görülmektedir. Birleştirme işlemi sonrasında derlenen kayıtlar, sıralandıktan (sort) sonra da son işlem olarak kayıtlar görüntülenmektedir (select). Örnek sorgu için en uzun sürenin %60'lık yürütme süresiyle sıralama (sort) işine ait olduğu bu sekmede görülmektedir (Şekil 12.17). "clustred index scan" (kümelenmiş dizin tarama) ve "clustred index seek" (kümelenmiş dizin arama) işlemleri ise sıralamadan sonra işlem maliyetini (Cost) omuzlayan işlemlerdir.



Şekil 12.17. Tahmini Yürütme Planının Görüntülenmesi

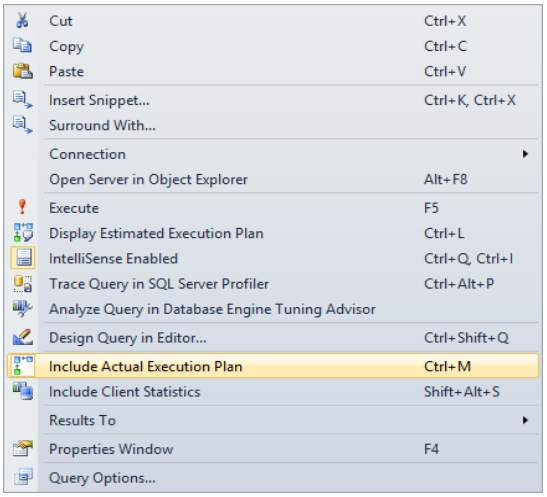
Tahmini yürütme planı oluşturulurken sorgu çalıştırılmadığı için sonuçlar (resuts) sekmesi görüntülenmemektedir.

Gerçek yürütme planı (Actual execution plan)

SQL Sunucu tarafından sorgu çalıştırılırken oluşturulan plandır. Bu yürütme planı daha sonra kullanılmak üzere "plan ön belleğine" (plan cache) kaydedilir. Böylece sorgu her çalıştırıldığında tekrar yürütme planı oluşturulmamış olur. Mevcut yürütme planı kullanıldığı için de performans artışı sağlanmış olur.

Gerçek yürütme planı daha sonra kullanılmak üzere plan ön belleğine (plan cache) kaydedilir.

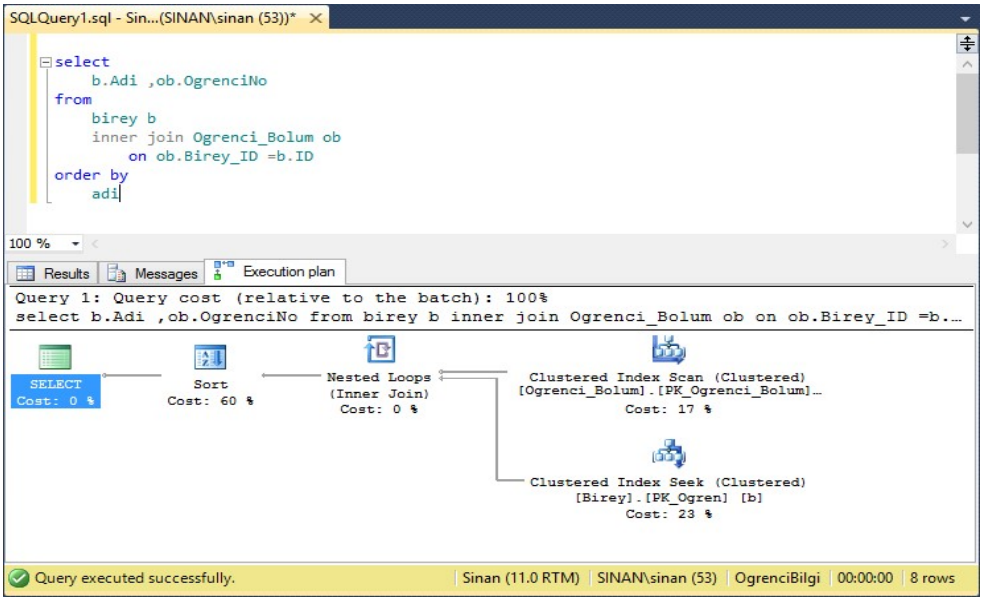
Gerçek yürütme planını görüntülemek için "SQL Editor" araç çubuğunda bulunan "Include Actual Execution Plan" düğmesine tıklanabilir, sorgu editöründe sağ tıklandığında açılan menüden "Include Actual Execution Plan" seçilebilir (Şekil 12.18) veya "Ctrl+M" tuşlarına birlikte basılabilir.



Şekil 12.18. Gerçek Yürütme Planının Açılması

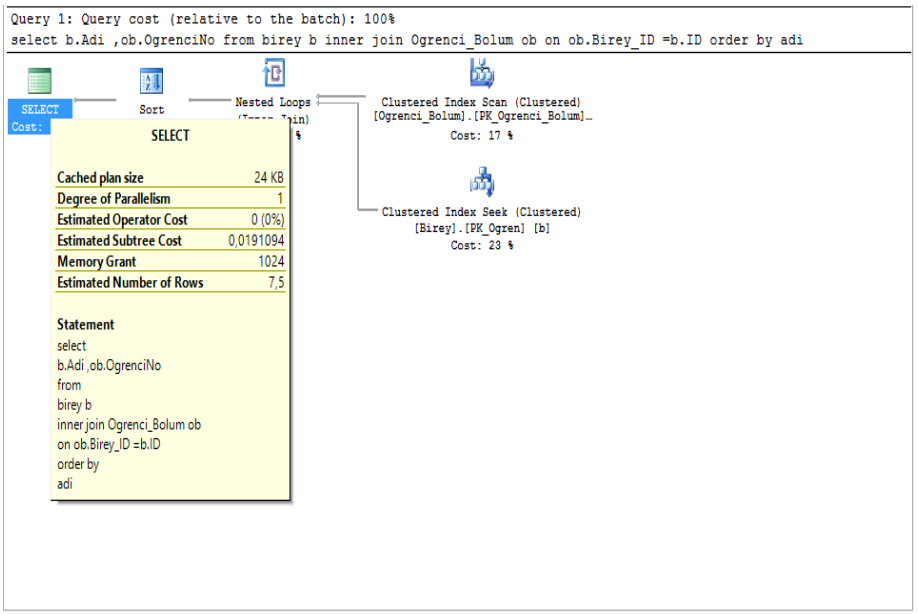
Tahmini yürütme planı ile gerçek yürütme planı her zaman aynı sonuçları vermeyebilir.

Sorgunun gerçek yürütme planı Şekil 12.19.daki gibi görüntülenecektir. Gerçek yürütme planı, örnek sorgu için tahmini yürütme planıyla aynı sonuçları vermiştir. Ancak her zaman aynı planlar oluşmayabilir.



Şekil 12.19. Gerçek Yürütme Planının Görüntülenmesi

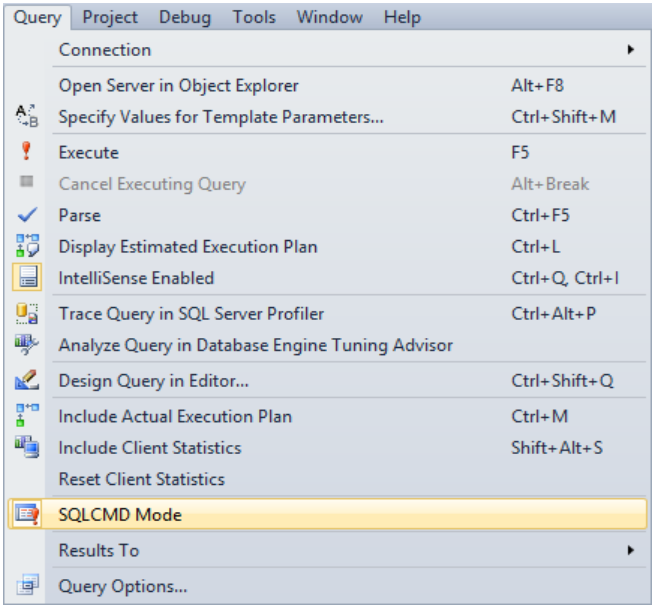
Yürütme planı içindeki ikonlar üzerinde fare imleci ile bir süre beklendiğinde Şekil 12.20.de görüldüğü gibi detaylı bilgilerin yer aldığı küçük bir ekran açılmaktadır.



Şekil 12.20. Gerçek Yürütme Planında Detayların Görüntülenmesi

VERİ TABANI SUNUCUSUNUN KOMUT SATIRINDAN VERİ TABANI İŞLEMLERİ YAPMAK

Sql Server de dos ekranı gibi, kendine özgü bir Komut İstemi Motoruna (Command Prompt Engine) sahiptir. SQL Server Management Studio ortamında komut istemi komutlarını çalıştırabilmek için de "Query" menüsü altında bulunan "SQLCMD Mode" seçilmelidir (Şekil 12.21). Ancak bu durumda otomatik tamamlama (intellisense) özelliğinin pasif olduğu bilinmelidir.



Şekil 12.21. SQLCMD Modunun Aktif Edilmesi

Sqlcmd komutları yazılırken ön ek olarak ":" deyimi ile başlar ve bir satırda sadece bir komut kullanılabilir.

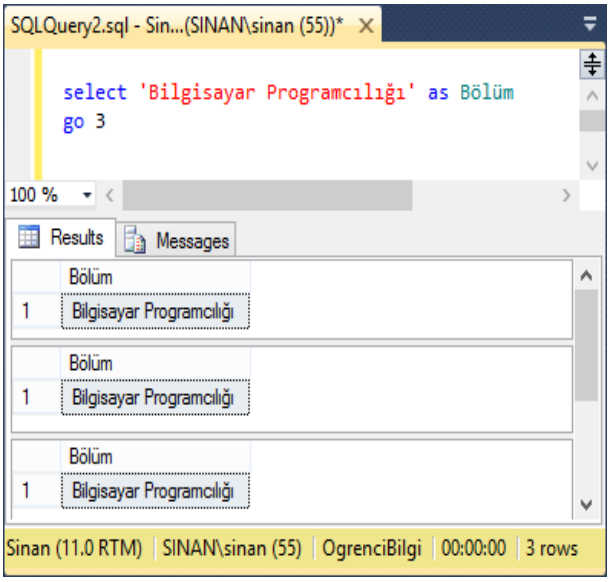
"Sqlcmd" komutları, T-SQL komutlarından ":" öneki ile ayrılmaktadır. Yani "Sqlcmd" komutları ":" deyimi ile başlamaktadır ve bir satırda sadece bir komut kullanılabilmektedir. İşletim sistemi komutları önünde "!!" deyimi bulunmalıdır.

Komut satırı komutlarından biri "Go" (git) komutudur. Bu komut tek başına kullanılabileceği gibi "!!:" önekiyle de kullanılabilir. "Go" komutunun çalışma şekli aşağıdaki gibidir.

Go komutu söz dizimi:

GO [*sayı*]

"Go" komutu "sayı" parametresi ile kullanıldığında kendinden önceki sorgu bloğunun kaç defa çalıştırılacağını ifade etmektedir. Şekil 12.22.de basit bir "Select" sorgusunun 3 defa çalıştırılmasına örnek verilmiştir.



Şekil 12.22. Sorgunun 3 Defa Çalıştırılması

Örnek

- SELECT 'Bilgisayar Programcılığı' AS Bölüm
- GO 3

Komut satırı komutlarından biri olan "Out" (dışarı) komutu ile de sorgu sonuçları bir dosyaya yazdırılabilmektedir. "Out" komutundan sonra gelen ilk parametre ile bu dosya adı ve konumu bildirilebilmektedir.

Söz dizimi:

OUT *dosyaAdı* |stderr|stdout

"Error" komutu, sorguda hata olması durumunda, ilgili hata mesajını bir dosyaya yazdırmaktadır. Bu komutun çalışma şekli de "Out" komutunun çalışma şekline benzemektedir. Komuttan sonra ilgili dosya adı ve konumu belirtilmektedir.

Out komutu ile sorgu sonuçlarını bir dosyaya yazdırmaktadır.

Söz dizimi:

ERROR *dosyaAdı* |stderr|stdout

"Quit" komutu ise sorgunun çalıştırılmasını sonlandırmaktadır. Bu komut ise herhangi bir parametre almadan kullanılmaktadır.

Söz dizimi:

QUIT

Bireysel Etkinlik

- OgrenciBilgi adında bir veri tabanı oluşturun.
- Veri tabanı oluşturma kodlarını içeren bir script dosyası oluşturun.
- Tablolar için oluşturulacak script dosyalarının türlerini araştırın(insert to, update to gibi)
- İki tabloyu önce dahili birleştirme(inner join) ile sonra da klasik birleştirme ile birleştiren kodları yazın. Bu iki kodun istemci istatistiklerini ve gerçek yürütme planlarını inceleyin.

Özet

- SQL Sunucu Yönetim Stüdyosu (SQL Server Management Studio), geliştiricilerin ve veri tabanı yöneticilerinin SQL Sunucu'ya erişimini sağlamak ve yönetmek için zengin grafiksel araçlar ve T-SQL komut yazım ortamını barındırır.
- "Object Explorer" penceresinde "Databases" klasörüne sağ tıklanarak veya var olan veri tabanı isimlerinden herhangi birine sağ tıklanarak açılan menüden "New Database" seçeneği seçilerek veri tabanı oluşturulabilir.
- İlgili veri tabanı ismine sağ tıklandığında açılan menüden, veri tabanının adının ("Rename") ve özelliklerinin ("Properties") değiştirilebilmesi veri tabanının güncellenebilmesi ("Refresh") veya ortadan kaldırılması ("Delete") mümkündür.
- Veri tabanının oluşturulması ve ortadan kaldırılması kod penceresinden de yapılabilmektedir.
- Sorgulama işlemleri SQL Sunucu'da iki şekilde yapılabilmektedir. "Query Editor" ekranında manüel olarak T-SQL cümlecikleri yazılabileceği gibi "Design Query in Editor" ekranında sihirbaz kullanarak da sorgular oluşturulabilmektedir.
- Veri tabanı aktif iken "Ctrl+N" tuş kombinasyonu ile veya standart araç çubuğunda bulunan "New Query" düğmesine tıklandığında sorgu editörü açılır ve bu ekranda aktif veri tabanı için sorgular yazılabilir.
- Sorgunun çalıştırılması için, sorgu ekranı aktif iken "Alt+X" tuş kombinasyonu ve "Ctrl+E" tuş kombinasyonu kullanılabilir veya F5 tuşuna basılabilir. Sorgu komutlarının çalıştırılması için ayrıca araç düğmeleri içinde bulunan "Execute" (çalıştır) düğmesine de tıklanabilmektedir.
- Sorgu çalıştırıldıktan sonra dönen sonuçlar, "results"(sonuçlar) sekmesinde görüntülenmektedir.
- Sorgu editöründe yazılan kodların ".sql" uzantısıyla bilgisayara kaydedilmesi için "Ctrl+S" tuş kombinasyonu kullanılabilir. Kayıtlı dosyaları açılması için "Ctrl+O"; yeni sorgu sayfası oluşturulması için de "Ctrl+N" uygulanabilir.
- Sorgu tasarlayıcısı ekranına geçmek için "Query" menüsünden veya "Query Editor" ekranında sağ tıklandığında açılan menüden "Design Query in Editor" seçilebilir veya "Ctrl+Shift+Q" tuş kombinasyonu kullanılabilir.
- Sorgu tasarlayıcısı ekranının üst bölmesinde diyagram(diagram), orta bölmesinde kriter(criteria) ve alt bölmesinde SQL sekmesi bulunmaktadır. Diyagram sekmesinde seçilen veri tabanı nesneleri ve aralarındaki ilişkiler gösterilmektedir.
- Kriter(criteria) sekmesinden sorguya dâhil edilecek kolon adı(column), kolon adına verilecek takma ad (alias), kolonun bulunduğu tablo adı (table), kolonun görüntülenip görüntülenmeyeceği (output), kolon adının sıralamaya dâhil edilip edilmeyeceği (sort order), sıralamaya dâhil edilecekse sıralama türü (sort type) ve filtreleme (filter) tanımlanabilmektedir.
- İstemci istatistikleri (client statistics), SQL Sunucu'da çalıştırılan sorguyla ilgili performans bilgilerini vermektedir. Böylece bir işlemi yapmak için yazılan farklı T-SQL kodlarından hangisinin daha performanslı olduğu incelenebilmektedir.
- Yürütme planı, sorgu iyileştirme (query optimizer) aracı tarafından hesaplanan ve bir sorgunun en ideal şekilde çalışması için önerilen yoldur. Tahmini yürütme planı ve gerçek yürütme planı olmak üzere iki tür yürütme planı bulunmaktadır.

DEĞERLENDİRME SORULARI

1. Bir sorgunun 5 defa çalıştırılmasını sağlamak için hangi komut kullanılmalıdır?
   a) GO 5
   b) COUNT 5
   c) NUMBER 5
   d) QUERY 5
   e) EXEC 5

2. Aşağıdaki işlemlerden hangisi veri tabanı üzerinde yapılamaz?
   a) Silme(delete)
   b) Adını değiştirme(rename)
   c) Sorgu oluşturma(new query)
   d) Veri tabanı oluşturma kodlarını görüntüleme(Script database as)
   e) Kayıt ekleme(insert into)

3. Aşağıdaki işlemlerden hangisi SQL Editor araç çubuğu üzerindeki düğmeler ile yapılamaz?
   a) Tahmini yürütme planını görüntüleme(display estimated execution plan)
   b) Komut satırı dosyalarının aktif edilmesi(SQLCMD Mode)
   c) Sorgunun çalıştırılması(execute)
   d) Çalışan bir sorgunun iptal edilmesi(cancel execution query)
   e) Aktif veri tabanının değiştirilmesi

4. Aşağıdaki tuş kombinasyonlarından hangisinin açıklaması yanlıştır?
   a) “Alt+X”: Sorgunun çalıştırılması
   b) “Ctrl+N”: Yeni sorgu sayfasının açılması
   c) “Ctrl+M”: Gerçek yürütme planının dâhil edilmesi
   d) “Ctrl+E”: Tahmini yürütme planının görüntülenmesi
   e) “Ctrl+U”: Aktif veri tabanının değiştirilmesi

5. İstemci istatistiği verilerinden hangisi sorgunun çalışma süresini yani istemci yürütme süresini saniye cinsinden göstermektedir?
   a) “Client Execution Time”
   b) “Network Statistics”
   c) “Wait time on server replies”
   d) “Total execution time”
   e) “Query Profile Statistics”

6. Sorgu çalıştırıldıktan sonra oluşan ve plan ön belleğine kaydedilen yürütme planının adı nedir?
   a) Tahmini yürütme planı (estimated execution plan)
   b) Sorgu yürütme planı (query execution plan)
   c) Gerçek yürütme planı (actual execution plan)
   d) Ön bellek yürütme planı (cache execution plan)
   e) Yürütme planı (execution plan)

7. Aşağıdakilerden hangisi veri tabanını oluşturan kodun aktarıldığı ortamlardan biri değildir?
   a) Dosya (file)
   b) Pano (clipboard)
   c) Izgara (grid)
   d) Zamanlanmış görev (agent job)
   e) Yeni sorgu editör penceresi (new query editor window)

8. Sorgu tasarım ekranından seçme (select) sorgusu hazırlanırken aşağıdaki işlemlerden hangisi yapılamaz?
   a) Tabloların seçilmesi
   b) Kolonların seçilmesi
   c) Sıralama
   d) Gruplama
   e) Filtreleme

9. Sorgu tasarım ekranında sorgu sonucu ile bir tablo oluşturulmak istenirse hangi sorgu tipi seçilmelidir?
   a) Tablo oluştur (create table)
   b) Tablo yap (make table)
   c) Sonuçları kaydet (insert results)
   d) Değerleri kaydet (insert values)
   e) Seç (select)

10. Sorgu çalıştırıldığında sorgu editörünün altında görüntülenen sekmelerden hangisinin açıklaması yanlıştır?
    a) Results: Sonuçlar
    b) Messages: Mesajlar
    c) Client Statistics: İstemci istatistikleri
    d) Execution plan: Gerçek yürütme planı
    e) Execution plan: Yürütme planı

Cevap Anahtarı

1.a, 2.e, 3.b, 4.d, 5.a, 6.c, 7.c, 8.d, 9.b, 10.d

YARARLANILAN KAYNAKLAR

Adar, İ., 2012. SQL Server 2012 Programlama ve Yönetim. İstanbul, 720

Gözüdeli, Y., 2010. Yazılımcılar için SQL SERVER 2008 R2 ve Veri tabanı Programlama. Ankara, 671

Elbahadır, H., 2012. T-SQL SQL SERVER 2012. İstanbul, 294

VERİ TABANI GÜVENLİĞİNİ SAĞLAMAK

İÇİNDEKİLER

- Yetkiler
- Roller
- Kullanıcı yetkilerinin düzenlenmesi
- Kullanıcı oluşturma
- Kullanıcı silme
- Kimlik doğrulama

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
- Bu üniteyi çalıştıktan sonra;
- Kullanıcı türlerini kullanıcıların veri tabanına erişim yetkilerini kavrayabilecek,
- Erişim yetkilerini tanımlayabilecek ve kısıtlayabilecek,
- Kullanıcı ekleyebilecek, silebilecek ve tanımlayabilecek,
- Kimlik doğrulama modları hakkında bilgi edinebileceksiniz.

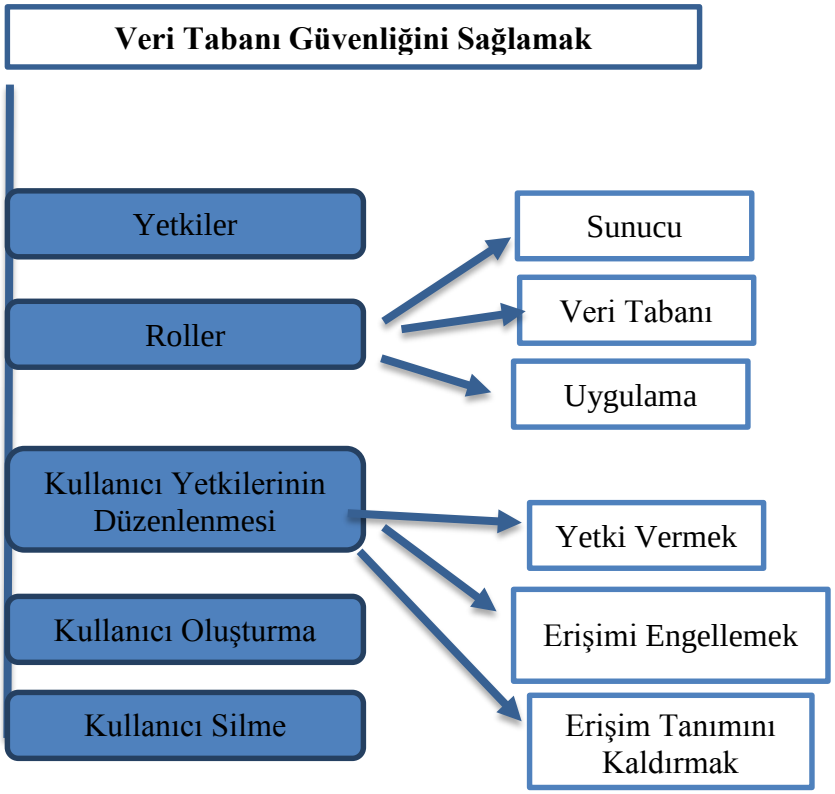
Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi Ahmet Kamil KABAKUŞ

ÜNİTE

13


Veri Tabanı Güvenliğini Sağlamak
Yetkiler
Roller
Kullanıcı Yetkilerinin Düzenlenmesi
Kullanıcı Oluşturma
Kullanıcı Silme
Sunucu
Veri Tabanı
Uygulama
Yetki Vermek
Erişimi Engellemek
Erişim Tanımını Kaldırmak

GİRİŞ

Önceki ünitelerde genel olarak T-SQL dili ile programlamanın kuralları öğretilerek çeşitli örnekler ile konunun pekiştirilmesi sağlanmıştı. Bir önceki ünitede T-SQL sorgularının yazılabildiği SQL Sunucu Yönetim Stüdyosu ve sorgu tasarlayıcısı ("Query Designer") ortamları tanıtılmıştı. Sorgu editörü ekranının ve sql editor araç çubuğunun kullanımına ve gerekli kısayol kombinasyonlarına değinilmişti. Sorgu performanslarının analiz edilmesini sağlayan istemci istatistikleri (client statistics) aracının kullanımı incelenmişti. Sorgu iyileştirme (query optimizer) aracı tarafından hesaplanan ve bir sorgunun en ideal şekilde çalışması için önerilen yürütme planlarının nasıl oluşturulabileceği gösterilmişti. Tahmini ve gerçek yürütme planlarının ne olduğu açıklandıktan sonra nasıl görüntüleneceği ve detaylı olarak nasıl okunabileceği incelenmişti. Bir önceki ünitede ayrıca komut satırından veri tabanı işlemlerinin nasıl yapılabileceği anlatılarak go, out, error ve quit komutlarının kullanımlarına değinilmişti.

Bu ünitede ise öncelikle her bir veri tabanı nesne türüne göre farklılaşabilen yetki türleri anlatılacaktır. Sonrasında her bir kullanıcı için yetkileri düzenlemek yerine, roller seviyesinde yetki kontrolünün nasıl yapılacağından bahsedilecektir. Kullanıcılar da çeşitli yetkiler ile veri tabanı nesnelerine erişecekken bu roller üzerinden tanımlanacaktır. Bu bağlamda sunucu, veri tabanı ve uygulama olmak üzere her bir rol grubu altındaki roller açıklanacaktır.

Sonrasında kullanıcı yetkilerinin düzenlenmesi başlığı altında kullanıcılara yetki verilmesi, kullanıcı erişimlerinin kısıtlanması/engellenmesi ve erişim tanımlarının kaldırılması konuları ayrıntılı olarak işlenecektir. Kullanıcı yetkileri düzenlenirken her bir yetki ve veri tabanı nesnesi ayrı ayrı belirtilebileceği gibi tüm (All) izinler ile ilgili işlem yapılacağı da belirtilebilmektedir. Benzer bir biçimde her bir kullanıcı için ayrıca yetki tanımlaması yapılabileceği gibi tüm kullanıcılar (Public) için tek bir SQL kodu ile kullanıcı yetkilendirmesi yapılabilmektedir.

Bu ünite kapsamında ayrıca kullanıcı oluşturma ve silme işlemlerinin T-SQL ile ve sihirbaz kullanılarak nasıl yapılacağı anlatılarak kullanıcı türünün, varsayılan dilin ve varsayılan veri tabanının nasıl belirlenebileceği gösterilecektir. Son olarak kimlik doğrulama başlığı altında Windows ve karma (mixed) yetkilendirme modlarının ne oldukları anlatılarak yetkilendirme modunun nasıl değiştirilebileceği, nasıl şifre tanımlanabileceği vs. anlatılacaktır.

Her bir veri tabanı nesne türü için yetkiler farklılaşabilmektedir. Örneğin tabloya "Insert" yetkisi tanımlanabilirken saklı yordama (stored procedure) "Execute" yetkisi tanımlanmaktadır.

YETKİLER

SQL Server'de kullanıcıların, veri tabanlarına erişimlerini düzenlemek için bütün yetkiler tanımlanabilmektedir ve veri tabanı nesnelerinin türlerine göre yetkiler, farklılaşabilmektedir. Böylece bütün kullanıcının bütün veri tabanları üzerinde sınırsız yetkiler ile donatılmasının yol açabileceği problemler önlenmiş olmaktadır. Kullanıcı, admin (veri tabanı yöneticisi), tarafından kendisine tahsis edilen veri tabanları içinde sadece yetkili olduğu veri tabanı nesneleri üzerinde admin tarafından tanımlı yetkiler kapsamında erişim sağlayabilmektedir.

Farklı veri tabanı nesnelerinde farklı işlemler yapılabileceği için yetkiler de farklılık arz etmektedir. Örneğin, tablo (table), görünüm (view) ve tablo döndüren fonksiyonlar için kullanıcıya "select" (kayıt seçme/görüntüleme), "insert" (kayıt ekleme), "update" (kayıt güncelleme), "delete" (kayıt silme) ve "references" (referans verme) yetkileri verilebilirken saklı yordam (stored procedure) için sadece execute (çalıştırma/yürütme) yetkisi verilebilmektedir. Veri tabanı içinse kullanıcıya "backup database" (veri tabanını yedekleme), "backup log" (veri tabanı için günlük yedekleme), "create database" (veri tabanı oluşturma), "create default" (tablodaki herhangi bir sütuna varsayılan değer atayabilme), "create function" (veri tabanında fonksiyon oluşturabilme), "create rule" (veri bütünlüğünü sağlamak için tablodaki herhangi bir sütuna kural koyabilme), "create table" (veri tabanı içinde tablo oluşturabilme) ve "create view" (veri tabanı içinde görünüm oluşturubilme) yetkileri atanabilmektedir (Tablo 13.1).

Tablo 13.1. Veri Tabanı İçin Tanımlanabilen Yetkiler

Nesne Türü	Yetkiler
Görünüm	DELETE, INSERT, REFERENCES, SELECT ve UPDATE
Saklı Yordam	EXECUTE
Skaler Fonksiyon	EXECUTE ve REFERENCES
Tablo	DELETE, INSERT, REFERENCES, SELECT ve UPDATE
Tablo Döndüren Fonksiyon	DELETE, INSERT, REFERENCES, SELECT ve UPDATE
Veri Tabanı	BACKUP DATABASE, BACKUP LOG, CREATE DATABASE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE TABLE ve CREATE VIEW.

ROLLER

Roller, yetki ve erişim tanımlamalarını gruplamaktadır. Böylece her bir kullanıcı için ve her bir veri tabanı nesnesi için ayrı ayrı yetkileri düzenlemek yerine, daha üst seviyede denetlemek daha kolay ve düzenli olmaktadır. Kullanıcılar, rollere dâhil edildiklerinde, dâhil edildikleri rollerin yetkileriyle donatılmaktadır.

Roller; sunucu rolleri, veri tabanı rolleri ve uygulama rolleri olmak üzere üçe ayrılmaktadır.

Sunucu Rolleri

Sunucu (server) rolleri, veri tabanı sunucusunun çalışması ile ilgili yetkileri içermektedir. Bu roller, Dbcreator, Diskadmin, Processadmin, Securityadmin, Serveradmin, Setupadmin, Sysadmin ve Bulkadmin olmak üzere 8 tanedir. "Dbcreator" (veri tabanı oluşturucusu) rolüne sahip olan bir kullanıcı, veri tabanı oluşturma, silme, değiştirme yetkisine sahip iken "Setupadmin" (kurulum yöneticisi) ise farklı veri tabanları kullanarak işlem yapma yetkisi vermektedir.

"Diskadmin" (disk yöneticisi) rolü ise disk üzerinde bulunan dosyaları yönetme yetkilerini içerirken ve "Processadmin" (süreç yöneticisi), SQL Server üzerinde çalışan işlemcileri kontrol etme yetkilerini içerirken "Bulkadmin" (çoklu

kayıt yöneticisi) ise çoklu kayıt ekleme komutlarını çalıştırabilme yetkilerini içermektedir.

"Serveradmin" (sunucu yöneticisi) rolü, yapısal ayarlamaları, sunucunun durdurulması ve yeniden başlatılması gibi ayarlamaları yapabilme yetkisi verirken "Securityadmin" (güvenlik yöneticisi) ise kullanıcıların sunucu bazında yetkilerini tanımlama, yönetme ve şifrelerini sıfırlama gibi yetkiler içermektedir.

"Sysadmin" (sistem yöneticisi) ise yukarıda bahsedilen tüm rolleri de içine alarak SQL Server üzerinde en yüksek yetkiler vermektedir. Yani, "Sysadmin" rolüne sahip bir kullanıcı her türlü işlemi yapabilme yetkisindedir.

Tablo 13.2. Sunucu (Server) Rolleri (Adar, 2012)

Rol Adı	Açıklama
Dbcreator	Veri tabanı oluşturucusu, veri tabanı oluşturulması ve değiştirilmesi işlemlerini yönetir.
Diskadmin	Disk yöneticisi, veri tabanı dosyalarını yönetir.
Processadmin	Süreç yöneticisi, SQL Server üzerinde çalışan süreçleri (process) kontrol eder.
Securityadmin	Güvenlik yöneticisi, kullanıcıları ve kullanıcı yetkilerini tanımlar.
Serveradmin	Sunucu yöneticisi, SQL Server ayarlamalarını yapar.
Setupadmin	Kurulum yöneticisi, bağlı sunucu (linked server) ekleyebilir ve silebilir.
Sysadmin	Sistem yöneticisi, SQL Server'daki en yetkili roldür, her türlü işlemi gerçekleştirebilir. SQL Server'da bulunan "sa" kullanıcısı bu gruba üyedir.
Bulkadmin	Toplu ekleme yöneticisi, toplu ekleme (insert) komutlarını çalıştırabilir.

Db, "database" (veri tabanı) kelimelerinin kısaltmasıdır.

Veri Tabanı Rolleri

Veri tabanı rolleri, veri tabanı seviyesindeki yetkilerle ilgili tanımlı rollerdir. Her bir veri tabanı için ayrıca tanımlanması gerekir. DataBase kelimelerinin kısaltması olan "db" ön ekiyle başlayan bu roller, her bir veri tabanı için ayrıca tanımlanmaktadır.

Veri tabanı rolleri, "db_accessadmin", "db_backupoperator", "db_datareader", "db_datawriter", "db_ddladmin", "db_denydatareader", "db_denydatawriter", "db_owner", "db_none" ve "db_securityadmin" olmak üzere 10 tanedir. En geniş yetkilerle donatılan rol, "db_owner" (sahip) rolü iken "db_none" (hiç), rolünde ise hiçbir yetki bulunmamaktadır.

"db_accessadmin" (erişim yöneticisi) rolündeki kullanıcı, diğer kullanıcılara veri tabanına erişim yetkisi tanımlamaktadır. "db_backupoperator" (yedekleme operatörü) rolündeki kullanıcıda veri tabanını alma yetkisi bulunmaktadır.

"db_datareader" (veri okuyucusu), veri tablolarında veya görünümlerinde sadece okuma (select) yapabilirken "db_datawriter" ise veriler üzerinde değişiklik (veri ekleme, düzenleme ve silme) yapabilmektedir. "db_denydatareader" (veri

okuyucuyu mahrum etme) rolü "db_datareader" rolünün tam tersini yapar ve kullanıcının ilgili veri tabanındaki verileri okumasını engellerken "db_denydatawriter" (veri okuyucuyu mahrum etme) rolü ise "db_datawriter" rolünün tam tersini yapar ve kullanıcının ilgili veri tabanındaki veriler üzerinde değişiklik yapabilmesini engeller.

Tablo 13.3. Veri Tabanı Rolleri (Gözüdeli, 2010)

Rol Adı	Açıklama
db_accessadmin	Veri tabanı erişim yöneticisi, Windows kullanıcılarının ve SQL Server hesaplarının veri tabanına erişimlerini tanımlar.
db_backupoperator	Veri tabanının yedeğini alabilir.
db_datareader	Veri tabanında veri okuyucusudur, kullanıcı tabloları üzerinde SELECT komutunu
db_datawriter	Veri tabanında veri yazıcısıdır, kullanıcı tabloları üzerinde DELETE, INSERT ve UPDATE komutlarını çalıştırabilir.
db_ddladmin	Veri tabanı DDL yöneticisi, veri tabanı üzerinde herhangi bir DDL (Data Definition Language) komut çalıştırabilir.
db_denydatareader	Veri tabanı engelli veri okuyucusu, kullanıcı tabloları üzerinde SELECT komutunu
db_denydatawriter	Veri tabanı engelli veri yazıcısı, kullanıcı tablolarında DELETE, INSERT ve UPDATE komutlarını çalıştıramaz.
db_owner	Veri tabanının sahibidir. Her türlü ayarlamayı
db_none	Boş
db_securityadmin	Veri tabanı güvenlik yöneticisi, erişim yetkilerini tanımlar.

"db_owner", veri tabanının sahibini temsil eder.

"db_securityadmin", güvenlik yöneticisi rolüdür. "db_ddladmin" ise veri tabanı DDL (Data Definition Language – Veri Tanımlama Dili) yöneticisidir ve DDL komutları çalıştırabilmektedir.

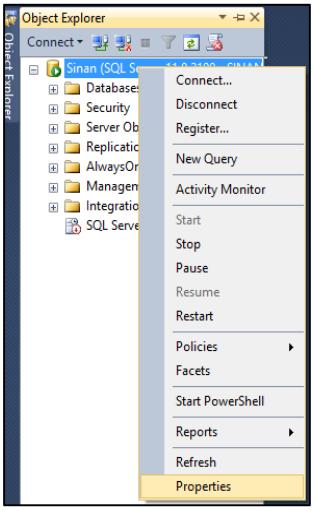
Uygulama (Application) Rolleri

Uygulama rolleri, herhangi bir uygulamaya özgü yetkilerin tanımlanabildiği yani o uygulama için özelleştirilmiş veri tabanı rolleridir.

Uygulama rolleri, herhangi bir uygulamaya özgü yetkilerin tanımlanabildiği yani o uygulama için özelleştirilmiş veri tabanı rolleridir.

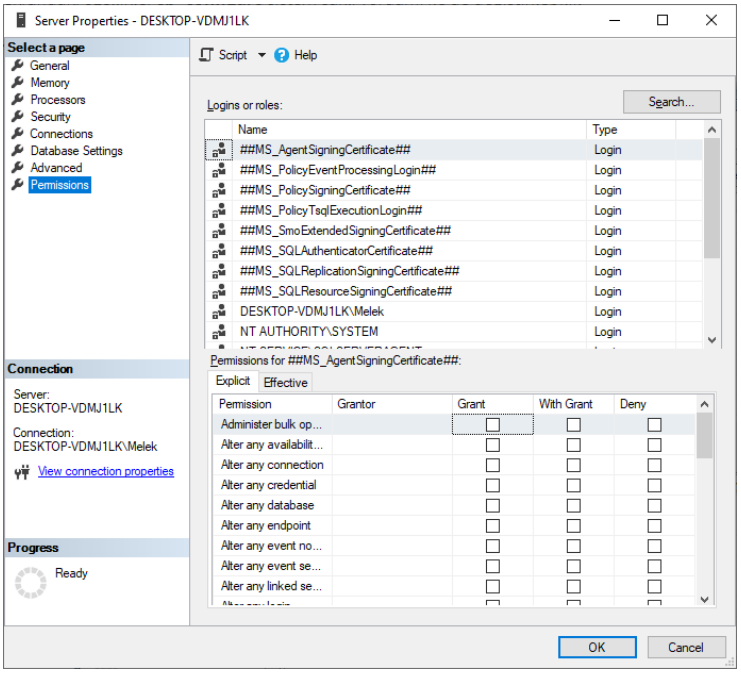
KULLANICI YETKİLERİNİN DÜZENLENMESİ

Kullanıcı yetkilerinin düzenlenmesinin öncelikle sihirbaz kullanılarak nasıp yapılacağı anlatılacaktır. Sunucu ismine sağ tıklandığında açılan menüden "Properties" (özellikler) seçeneği tıklanır (Şekil 13.1).



Şekil 13.1. Sunucu Özelliklerinin Açılması

Açılan pencerede "Permissions" (izinler) sekmesi Şekil 13.2.deki gibi seçilerek kullanıcı yetkileri düzenlenebilmektedir.



Şekil 13.2. Sunucu Özellikleri – İzinler Sekmesi

İzinler sekmesinin üst bölümünde mevcut loginler/roller seçildikten sonra ilgili login'e veya role ilişkin yetkiler düzenlenebilmektedir (Şekil 13.2). Her bir yetkinin yanında "Grant" (izin ver), "With Grant" ve "Deny" (engelle) kutucukları bulunmaktadır. "Grant" ve "Deny" kutucukları aynı anda işaretlenemezken "With Grant" kutucuğu işaretlendiğinde otomatik olarak "Grant" seçeneği de işaretlenmektedir.

Takip eden bölümlerde "Grant", "Deny" ve "With Grant" ("with grant option" olarak geçecek) kavramları detaylı olarak işlenirken yetkilerin TSQL kodları ile nasıl düzenleneceği yani kullanıcıya yetki verme, kullanıcı erişimini engelleme ve erişim tanımını kaldırma/geri alma işlemleri yapan TSQL komutları işlenecektir.

Yetki Vermek (Grant)

Veri tabanı kullanıcılarına T-SQL ile yetki tanımlamak için "Grant" ifadesi kullanılmaktadır.

Söz Dizimi:

```
GRANT ALL|izinler
[ON nesneler]
TO PUBLIC|kullanıcılar
[WITH GRANT OPTION]
```

Grant ifadesi ile birlikte "All" (hepsi) deyimi kullanılırsa bütün izinler verilmiş olur. Örnek olarak "Ali" adlı kullanıcıya tablo ve görünüm (view) oluşturma izinlerini verelim.

İzin tanımlanacak nesne ismi belirtilmezse tüm nesneler üzerinde tanımlanır.

Örnek

```
•GRANT CREATE TABLE, CREATE VIEW TO Ali
```

Örnek

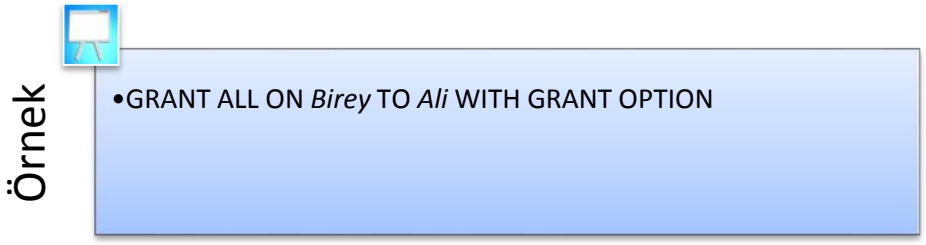
```
•GRANT SELECT ON Birey TO PUBLIC
```

Kullanıcı adı yerine "Public" (herkes) ifadesinin kullanılması, veri tabanına erişebilen tüm kullanıcılar için tanımlama yapmayı mümkün kılar. Örnek olarak tüm kullanıcılara "Birey" tablosundaki verileri okuma (select) yetkisi verelim.

"On" ifadesinin kullanımı seçimliktir, yani zorunlu değildir. "On" ifadesinden sonra hangi veri tabanı nesnesi üzerinde yetki verileceği tanımlanmaktadır. Kullanılmadığı durumda ise yetki tanımlaması, tüm nesneler üzerinde geçerli olmaktadır.

"Deny" komutu kullanıcının erişimini engellemektedir.

"With grant option" seçeneği, bir kullanıcıya verilen yetkiyi, bu kullanıcının, başkalarına verebilmesini sağlamaktadır (Elbahadır, 2012). Örnek olarak "Ali" kullanıcısı "Birey" tablosu üzerinde bütün yetkilerle donatalım ve "Ali" kullanıcısı da başkalarına "Birey" tablosu için yetki tanımlayabilsin. Bu örnekte "with grant option" ifadesi olmasaydı "Ali" kullanıcısı, başka kullanıcılara "Birey" tablosu için erişim yetkisi tanımlayamayacaktı.



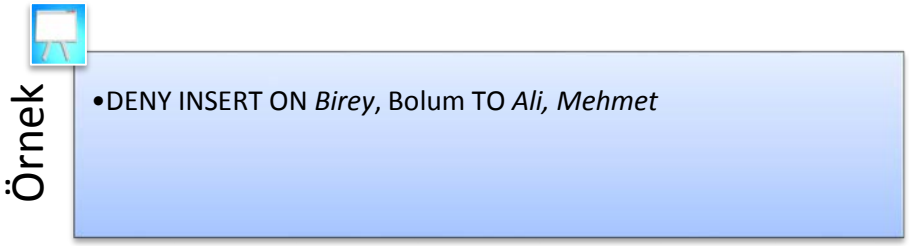
Erişimi Engellemek (Deny)

Veri tabanı kullanıcısının T-SQL kodu ile erişeme engellenebilmesi için "Deny" ifadesi kullanılmaktadır. Kullanıcılar, "Grant" komutuyla yetkilendirilmekteydi, "Deny" komutu ise kullanıcının erişimini engellemektedir. "Deny" komutunun kullanımı, "Grant" komutunun kullanımına benzemektedir. Ancak "with grant option" imkânı, "Deny" komutunda bulunmamaktadır.

Söz Dizimi:

```
DENY ALL|izinler
[ON nesneler]
TO PUBLIC|kullanıcılar
```

"Deny" ifadesi ile birlikte "All" deyimi kullanılırsa bütün izinler iptal edilmiş olur. Örnek olarak "Ali" ve "Mehmet" adlı kullanıcılardan "Birey" ve "Bolum" tablolarına veri ekleme (insert) engeli getirelim.



Erişim Tanımını Kaldırmak

"Grant" ile verilen yetkiler ve "deny" ile yapılan engellemeler, "revoke" (geri almak) ifadesi ile eski haline getirilebilmektedir. Yani tüm yetki değişiklikleri geri alınmaktadır.

Söz Dizimi:

```
REVOKE ALL|izinler
[ON nesneler]
TO|FROM PUBLIC|kullanıcılar
[CASCADE]
```

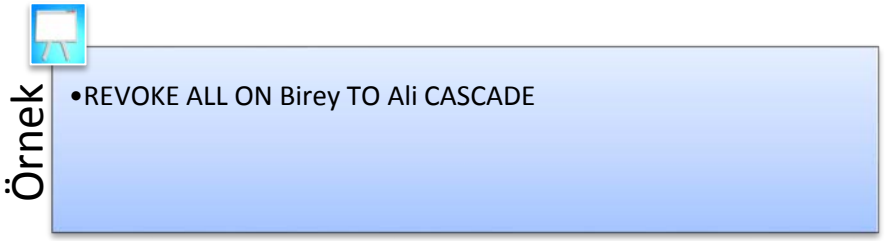
Örnek olarak "Mehmet" kullanıcısı için "Birey" tablosuna daha önce verilmiş

olan veri ekleme yetkisini iptal edelim yani geri alalım.

Erişim tanımı kaldırılırken "to" ve "from" ifadeleri seçimliktir yani ikisinden biri tercihen kullanılabilir. Ancak boş geçilemez yani "to" ve "from" ifadelerinin ikisinden birinin kullanılması zorunludur. "Cascade" ifadesi ile de kullanıcının yetki düzenlemeleri iptal edildikten sonra, bu kullanıcı tarafından başka kullanıcılara yetki tanımlanmışsa o yetkilerin de topluca iptal edilmesi sağlanmaktadır. Yani daha önce "grant" komutu altında anlatıldığı üzere "with grant option" seçeneği ile kullanıcılara kendilerine verilen yetkiyi başka kullanıcılar için de kullanabilme imkânı dolayısıyla verilen tüm alt yetkiler iptal edilmektedir.

"CASCADE" ifadesi ile kullanıcı tarafından başka kullanıcılara tanımlanan yetkiler de iptal edilmektedir.

Bir örnekle "cascade" parametresinin kullanımını somutlaştıralım. "Ali" kullanıcısına daha önce "Birey" tablosu için tüm yetkiler verilmişti ve başka kullanıcılara da yetki verebilmesi ("with grant option" ile) sağlanmıştı. Şimdi de birey tablosu için "Ali" kullanıcısına verilen tüm yetkileri kaldırırken "Ali" kullanıcısının başka kullanıcılara verdiği olası tüm yetkileri de kaldıralım.



KULLANICI OLUŞTURMA

Veri tabanına erişecek kullanıcının oluşturulması, "CREATE USER" deyimi ile yapılmaktadır. Bu deyimden sonra kullanıcı adı verilerek tanımlama gerçekleştirilmiş olur. Deyimin kullanımında bulunan "For login" ifadesi ve sonrası ise seçimliktir. Kullanıldığında hangi yerel kullanıcı hesabı kaydı için bir kullanıcı tanımlanacağı belirtilebilmektedir.

Söz Dizimi:

CREATE USER *Kullanıcı_Adı* [FOR LOGIN *Login_Adı*]

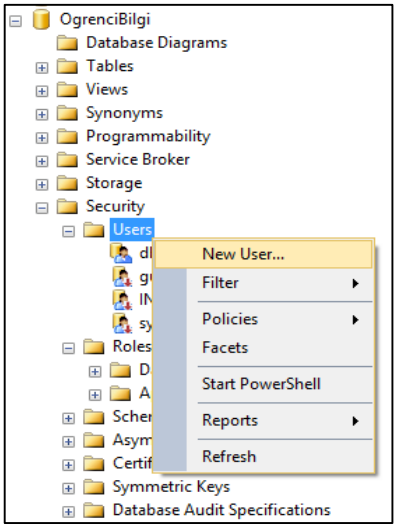
TSQL kodu kullanarak kullanıcı tanımlamaya örnek olması için "Melis" adında bir kullanıcı oluşturalım:

Örnek

•CREATE USER *Melis*

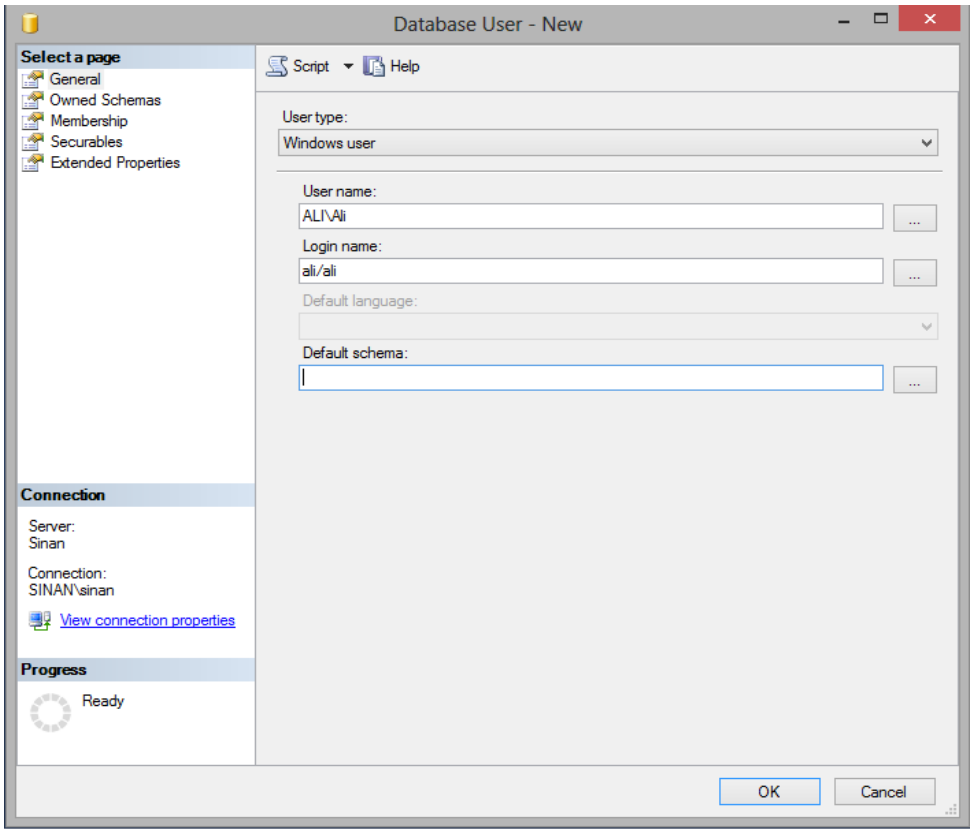
Şimdi de SQL Server Management Studio ortamında sihirbaz kullanarak bir kullanıcının nasıl oluşturulabileceğini görelim. Bunun için ilgili veri tabanı altında bulunan "Security" (güvenlik) klasörü içindeki "Users" (kullanıcılar) klasörüne Şekil 13.3.teki gibi sağ tıklanır ve "New User" (yeni kullanıcı) seçeneği seçilir.

Veri tabanı kullanıcıları, veri tabanı altında "Security" klasörü altında "Users" klasörü altındadır.



Şekil 13.3. Yeni Kullanıcı Ekleme

Şekil 13.4.teki gibi kullanıcı ekleme penceresi gelecektir. Buradan "user type" (kullanıcı tipi), "user name" (kullanıcı adı), "login name" (yerel kullanıcı hesabı adı) ve "default schema" (varsayılan şema) belirlendikten sonra "Ok" (tamam)'a tıklanarak yeni kullanıcı kaydı oluşturulur.



Şekil 13.4. Yeni Kullanıcı Kaydının Oluşturulması

Kullanıcı tanımlarken gerekli yetkiler tanımlanabilmektedir.

Herhangi bir domain veya yerel kullanıcı hesabının SQL Server ile ilişkilendirilmesi işlemi T-SQL ile şu şekilde yapılabilmektedir:

Söz Dizimi:

```
CREATE LOGIN Login_Adı
FROM WINDOWS
WITH DEFAULT_DATABASE= VeriTabanı_Adı,
DEFAULT_LANGUAGE = dil
```

Örnek

```
•CREATE LOGIN Ali
•FROM WINDOWS
•WITH DEFAULT_DATABASE= OgrenciBilgi,
•DEFAULT_LANGUAGE = Turkish
```

"From windows" ifadesi ile tanımlanan kullanıcının Windows kullanıcısı olduğu "default_database" (varsayılan veri tabanı) ifadesi ile kullanıcının, sunucuya giriş yaptığında otomatik olarak bağlanacağı veri tabanı tanımlanabilmektedir. "Default_language" (varsayılan dil) parametresi ile de kullanıcının varsayılan dil ayarı belirlenebilmektedir.

KULLANICI SİLME

Mevcut kullanıcı adının SQL Server'dan kaldırılabilmesi için "drop user" deyiminin kullanımı yeterlidir. "drop user" deyiminden sonra silinecek kullanıcının adının yazılması yeterlidir.

Söz Dizimi:

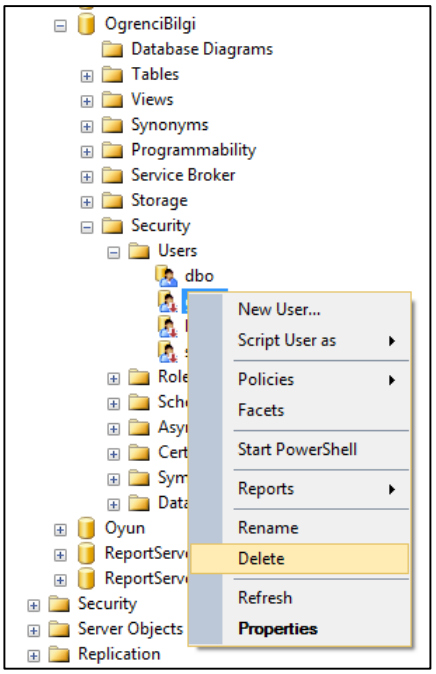
```
DROP USER KullanıcıAdı
```

Örnek olması için de daha önce tanımladığımız "Ali" isimli kullanıcıyı silelim.

Örnek

```
•DROP USER Ali
```

SQL Server Management Studio ortamında kullanıcı adı üzerine Şekil 13.5.te görüldüğü gibi sağ tıklandığında açılan menüden "Delete" seçilerek de ilgili kullanıcı silinebilmektedir. İki kullanım arasında bir fark bulunmamaktadır.



Şekil 13.5. Yeni Kullanıcı Adının Belirlenmesi

Bu menü altından ayrıca kullanıcı adının (Rename) ve özelliklerinin (Properties) de değiştirilebileceğini, yeni kullanıcı (New User) eklenebileceğini de yeri gelmişken söyleyelim.

Windows yetkilendirme modu ve karma(mixed) mod olmak üzere iki kimlik tanımlama modu bulunmaktadır.

KİMLİK DOĞRULAMA (AUTHENTICATION)

SQL Server'de kimlik doğrulama, Windows yetkilendirme modu ve karma (mixed) mod olmak üzere iki şekilde yapılmaktadır. Varsayılan (default) mod ise Windows yetkilendirme modudur.

Windows üzerinde yetkilendirilmiş kullanıcılar için Windows yetkilendirme modu seçilirken karma mod seçildiğinde hem Windows yetkilendirme modunun hem de SQL Server kullanıcılarının tanımlanması mümkün olmaktadır. Diğer bir deyişle Windows mod seçildiğinde sadece işletim sisteminde tanımlı olan kullanıcı adı ve şifresiyle sisteme giriş yapılabilir. Yani kullanıcı adı ve şifre bilgileri SQL Server'de kayıtlı tutulmakta ve kullanıcı doğrulaması buradan da sağlanabilmektedir. Karma mod seçildiğinde ise hem işletim sisteminde kayıtlı kullanıcı adı ve şifresiyle giriş yapılabilirken hem de SQL Server'de tanımlı kullanıcı adı ve şifresiyle sisteme giriş yapılabilmektedir.

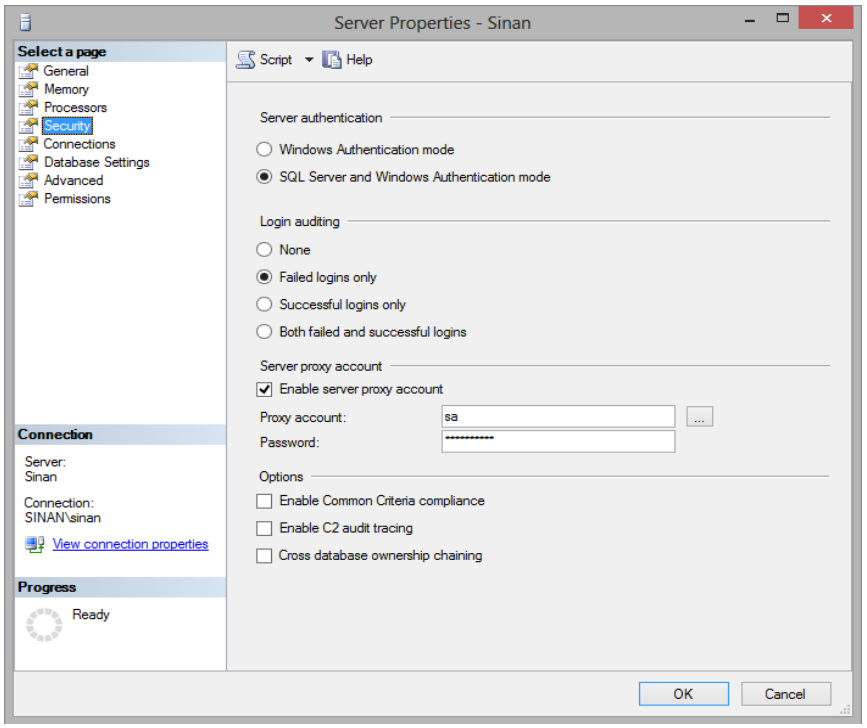
Veri tabanı sunusu yerel bilgisayarda ise uygulama yazılımı ve veri tabanı aynı bilgisayarda ise Windows yetkilendirme modunun seçilmesi daha uygundur. Ancak eğer bir çalışma grubu içindeyseniz ve SQL Server'a uzak bilgisayarlardan erişim sağlanacaksa veya SQL Server'de tanımlı veri tabanına bir internet uygulaması üzerinden erişilmesi planlanıyorsa karma mod kullanılması daha uygundur.

Eğer çok gerekli değilse Windows kimlik doğrulaması önerilmektedir. Çünkü Windows kimlik doğrulaması, daha sağlam şifreleme kullanmaktadır. SQL Server oturumlar kullanıldığında ise kullanıcı adları ve parolalar her ne kadar şifrelenerek taşınsa da ağ üzerinden geçirildiği için daha az güvenliklidir.

Yetkilendirme modu, kurulum esnasında seçilebildiği gibi daha sonradan da değiştirilebilmektedir. Kurulum sonrasında yetkilendirme modunu düzenlemek için Şekil 13.1.deki gibi sunucu ismine sağ tıklandığında açılan menüden "Properties" (özellikler) seçeneği tıklanır. Açılan pencerede "Security" (güvenlik) sekmesi Şekil 13.6.daki gibi seçilerek sunucu yetkilendirme "Server authentication" bölümü altından düzenlenebilmektedir.

Sunucu özellikleri altında bulunan "Permissions" sekmesi ile de sunucuya erişim izinleri tanımlanabilmektedir.

"Login Auditing" (giriş denetimi) altından kullanıcı login işlemi ile ilgili hangi bilgilerin log kayıtlarının tutulacağı düzenlenebilmektedir. Varsayılan olarak "Failed logins only" (sadece hatalı girişler) seçeneği seçilidir. Hiçbir girişin log kaydının tutulmaması için "None" seçilebilirken başarılı ve başarısız tüm girişlerin kaydedilmesi için de "Both failed and succesful logins" seçilebilmektedir. Bunlar dışında bir de "Succesful logins only" (sadece başarılı girişler) seçeneği bulunmaktadır ki bu seçenek ile başarısız girişlerin log kayıtları tutulmamaktadır.



Şekil 13.6. Sunucu Güvenlik Özelliklerinin Düzenlenmesi

"Connection" sekmesi altından SQL Server'ın uzaktan erişime izin verip vermeyeceği belirlenebilmektedir.

Sunucu özellikleri penceresinde bulunan diğer sekmeler altından da ne tür işlemlerin yapılabileceğini özetle bahsedelim. "General" (genel) sekmesi altından sunucunun adı, SQL Server ve işlemci versiyon bilgileri, dil bilgisi, toplam bellek boyutu, işlemci sayısı ve kök dizini gibi bilgiler görüntülenmektedir. Bu sekmede herhangi bir düzenleme yapılmasına izin verilmemektedir.

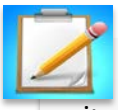
"Memory" (bellek) sekmesi altından işletim sisteminin izin verdiği kapasite sınırları içinde kalmak kaydıyla minimum ve maksimum bellek miktarı ayarlanabilirken "processors" (işlemciler) sekmesi altında SQL Server'ın kullanacağı işlemciler için temel yapılandırma ayarları bulunmaktadır.

"Connection" (bağlantı) sekmesi altından SQL Server'ın uzaktan erişime izin verip vermeyeceği belirlenebilmekte, uzaktan erişime izin verecekse de aynı anda yapılabilecek bağlantı sayısı sınırlaması düzenlenebilmektedir.

"Database Settings" (veri tabanı özellikleri) sekmesi altından ise veri tabanı dosyası, log dosyası ve yedekleme dosyalarının nerede tutulacağı belirlenirken yedeklemeyle ilgili diğer ayarlamalar da yapılabilmektedir.

"Advanced" (gelişmiş) sekmesi altından SQL Server ile ilgili ileri düzey ayarlamalar yapılabilmektedir. "Permission" sekmesi ise daha önce "kullanıcı yetkilerinin düzenlenmesi" bölümünde anlatılmıştı.

Bireysel Etkinlik



- İki arkadaşınızın, veri tabanına erişebilmesi için SQL Server tarafında arkadaşlarınızın adıyla birer kullanıcı oluşturun.
- Bir arkadaşınıza, tablolardaki verileri okuma (Select) ve yazma (insert, update) yetkilerini verin. Başka kullanıcılara da bu yetkileri verebilsin.
- Bu arkadaşınız, diğer arkadaşınız için tablolara okuma (select) yetkisi versin.
- Bu arkadaşlarınızın tüm yetkilerini bir T-SQL komutu ile kaldırın.

Özet

- SQL Server'de kullanıcıların veri tabanlarına erişimlerini düzenlemek için bütün yetkiler tanımlanabilmektedir ve veri tabanı nesnelerinin türlerine göre yetkiler, farklılaşabilmektedir.
- Roller yetki ve erişim tanımlamalarını gruplar. Böylece her bir kullanıcı için yetkileri düzenlemek yerine, roller seviyesinde denetlenmek daha kolay ve düzenli olmaktadır. Kullanıcılar, dâhil oldukları rollerin yetkileriyle donatılır.
- Roller, sunucu rolleri, veri tabanı rolleri ve uygulama rolleri olmak üzere üçe ayrılmaktadır.
- Sunucu rolleri, veri tabanı sunucusunun çalışması ile ilgili yetkileri içermektedir.
- Veri tabanı rolleri, veri tabanı seviyesindeki yetkilerle ilgili tanımlı rollerdir.
- Uygulama rolleri, herhangi bir uygulamaya özgü yetkilerin tanımlanabildiği yani o uygulama için özelleştirilmiş veri tabanı rolleridir.
- Veri tabanı kullanıcısına T-SQL ile yetki tanımlamak için "GRANT" ifadesi kullanılmaktadır.
- Grant ifadesi ile birlikte "ALL" deyimi kullanılırsa bütün izinler verilmiş olur.
- Kullanıcı adı yerine "PUBLIC" ifadesinin kullanılması, veri tabanına erişebilen tüm kullanıcılar için tanımlama yapmayı mümkün kılar.
- "WITH GRANT OPTION" seçeneği, bir kullanıcıya verilen yetkiyi, bu kullanıcının, başkalarına verebilmesini sağlamaktadır.
- Veri tabanı kullanıcısındaki bir yetkinin T-SQL kodu ile kaldırılması için "DENY" ifadesi kullanılmaktadır.
- Grant ifadesi ile birlikte "ALL" deyimi kullanılırsa bütün izinler iptal edilmiş olur.
- "GRANT" ile verilen yetkiler ve "DENY" ile yapılan engellemeler, "REVOKE" ifadesi ile eski hâline getirilebilmektedir. Yani tüm yetki değişiklikleri geri alınmaktadır.
- Veri tabanına erişecek kullanıcının tanımlanması, "CREATE USER" deyimi ile yapılmaktadır. SQL Server Management Studio ortamında sihirbaz kullanarak kullanıcı oluşturmak için de veri tabanı altında, "Security" klasörü altında yer alan "Users" klasörüne sağ tıklanır ve "New User" seçeneği seçilir.
- Veri tabanına erişecek kullanıcının silinmesi "DROP USER" deyimi ile yapılmaktadır. SQL Server Management Studio ortamında kullanıcı adı üzerine sağ tıklandığında açılan menüden "Delete" seçilerek de kullanıcı silinebilmektedir.
- SQL Server'da kimlik doğrulama, Windows yetkilendirme modu ve karma (mixed) mod olmak üzere iki şekilde yapılmaktadır. Windows üzerinde yetkilendirilmiş kullanıcılar için Windows yetkilendirme modu seçilmektedir. Karma mod ise hem Windows yetkilendirme modunun kullanıldığı hem SQL Server kullanıcılarının tanımlanabildiği moddur.
- Yetkilendirme modu, kurulum esnasında seçilebildiği gibi daha sonradan da değiştirilebilmektedir. Kurulum sonrasında düzenlemek için aşağıdaki gibi sunucu ismine sağ tıklanarak açılan menüden "Properties" (özellikler) seçeneği tıklanır.
- Açılan pencerede "Security" (güvenlik) sekmesi aşağıdaki resimdeki gibi seçilerek sunucu yetkilendirme (server authentication) değiştirilebilir ve gerekli şifre tanımlamaları yapılabilir.

DEĞERLENDİRME SORULARI

1. Aşağıdaki eşleştirmelerden hangisi yanlıştır?
   a) GRANT: Yetki vermek
   b) DENY: Engellemek
   c) REVOKE: Verilen yetkileri geri almak
   d) CREATE USER: Kullanıcı oluşturmak
   e) DELETE USER: Kullanıcı silmek

2. Aşağıdaki yetkilerden hangisi kullanıcıların veri tabanı tablolarına erişimleri için tanımlanamaz?
   a) DELETE
   b) INSERT
   c) SELECT
   d) EXECUTE
   e) REVOKE

3. "GRANT" komutu ile ilgili aşağıdaki bilgilerden hangisi yanlıştır?
   a) "ALL" deyimi bütün yetkileri vermek için kullanılır.
   b) "ON" deyiminin kullanımı zorunludur.
   c) "PUBLIC" deyimi bütün kullanıcılar yerine kullanılır.
   d) "WITH GRANT OPTION" deyimi ile yetkilendirilen kullanıcı başka kullanıcıları da yetkilendirebilir.
   e) "TO" deyiminden sonra kullanıcı veya kullanıcılar bildirilir.

4. Bütün kullanıcıların "Birey" tablosundaki kayıtları seçme (select) yetkisi için yapılan tanımlama aşağıdakilerden hangisidir?
   a) REVOKE SELECT ON Birey TO PUBLIC
   b) GRANT SELECT TO Birey ON ALL
   c) GRANT SELECT ON Birey TO PUBLIC
   d) GRANT SELECT ON Birey TO ALL
   e) GRANT SELECT Birey TO PUBLIC

5. Mehmet kullanıcısının "Birey" tablosundaki kayıtları silme (delete) yetkisinin engellenmesi için yapılan tanımlama aşağıdakilerden hangisidir?
   a) REVOKE DELETE ON Birey TO Mehmet
   b) DENY DELETE ON Birey TO Mehmet
   c) DENY DELETE ON Mehmet TO Birey
   d) GRANT DELETE ON Birey TO Mehmet
   e) GRANT DELETE ON Mehmet TO Birey

6. Mehmet kullanıcısının silinmesi için yapılan tanımlama aşağıdakilerden hangisidir?
   a) REVOKE ALL FROM Mehmet
   b) DENY ALL TO Mehmet
   c) DELETE Mehmet
   d) DELETE ON Mehmet
   e) DROP USER Mehmet

GRANT SELECT, INSERT, UPDATE ON Birey TO Ali, Mehmet WITH GRANT OPTION

7. Verilen yetkilendirme koduyla ilgili aşağıdaki bilgilerden hangisi yanlıştır ?
   a) Mehmet kullanıcısı "Birey" tablosuna kayıt ekleyebilir.
   b) Ali kullanıcısı "Birey" tablosundaki kayıtları güncelleyebilir.
   c) Ali kullanıcısı "Birey" tablosundaki kayıtları, başka kullanıcıların seçebilmesi için onlara yetki tanımlayabilir.
   d) Mehmet kullanıcısı "Birey" tablosundaki kayıtları okuyabilir ancak değiştiremez.
   e) Ali ve Mehmet kullanıcılarının "Birey" tablosuna erişim yetkileri tanımlanmaktadır.

8. Aşağıdakilerden hangisi sunucu rollerinden biri değildir?
   a) Diskadmin
   b) Db_datawriter
   c) Setupadmin
   d) Bulkadmin
   e) Securityadmin

9. Aşağıdaki rollerden hangisinin açıklaması yanlıştır?
   a) Diskadmin: Disk yöneticisi
   b) Serveradmin: Sunucu yöneticisi
   c) Sysadmin: Kurulum yöneticisi
   d) Bulkadmin: Toplu ekleme yöneticisi
   e) Securityadmin: Güvenlik yöneticisi

10. Aşağıdaki rollerden hangisi veri tabanı üzerinde her türlü ayarlamayı yapabilir?
   a) Db_accessadmin
   b) Db_owner
   c) Db_none
   d) Db_securityadmin
   e) Db_ddladmin

Cevap Anahtarı

1.e, 2.d, 3.b, 4.c, 5.b, 6.e, 7.d, 8.b, 9.c, 10.b

YARARLANILAN KAYNAKLAR

Adar, İ., 2012. SQL Server 2012 Programlama ve Yönetim. İstanbul, s.720.
Elbahadır, H., 2012. T-SQL SQL SERVER 2012. İstanbul, s.294.
Gözüdeli, Y., 2010. Yazılımcılar için SQL SERVER 2008 R2 ve Veritabanı Programlama. Ankara, s.671.

VERİ TABANINDA YEDEKLEME VE GERİ YÜKLEME

İÇİNDEKİLER

- İşlem Günlüğü
- Günlükleme Seçenekleri
- SQL Sunucu Otomatik Kurtarma
- Yedekleme Araçları
- Yedekleme Türleri
- SQL Sunucu'da Yedekleme
- Geri Yükleme Türleri
- SQL Sunucu'da Geri Yükleme

HEDEFLER

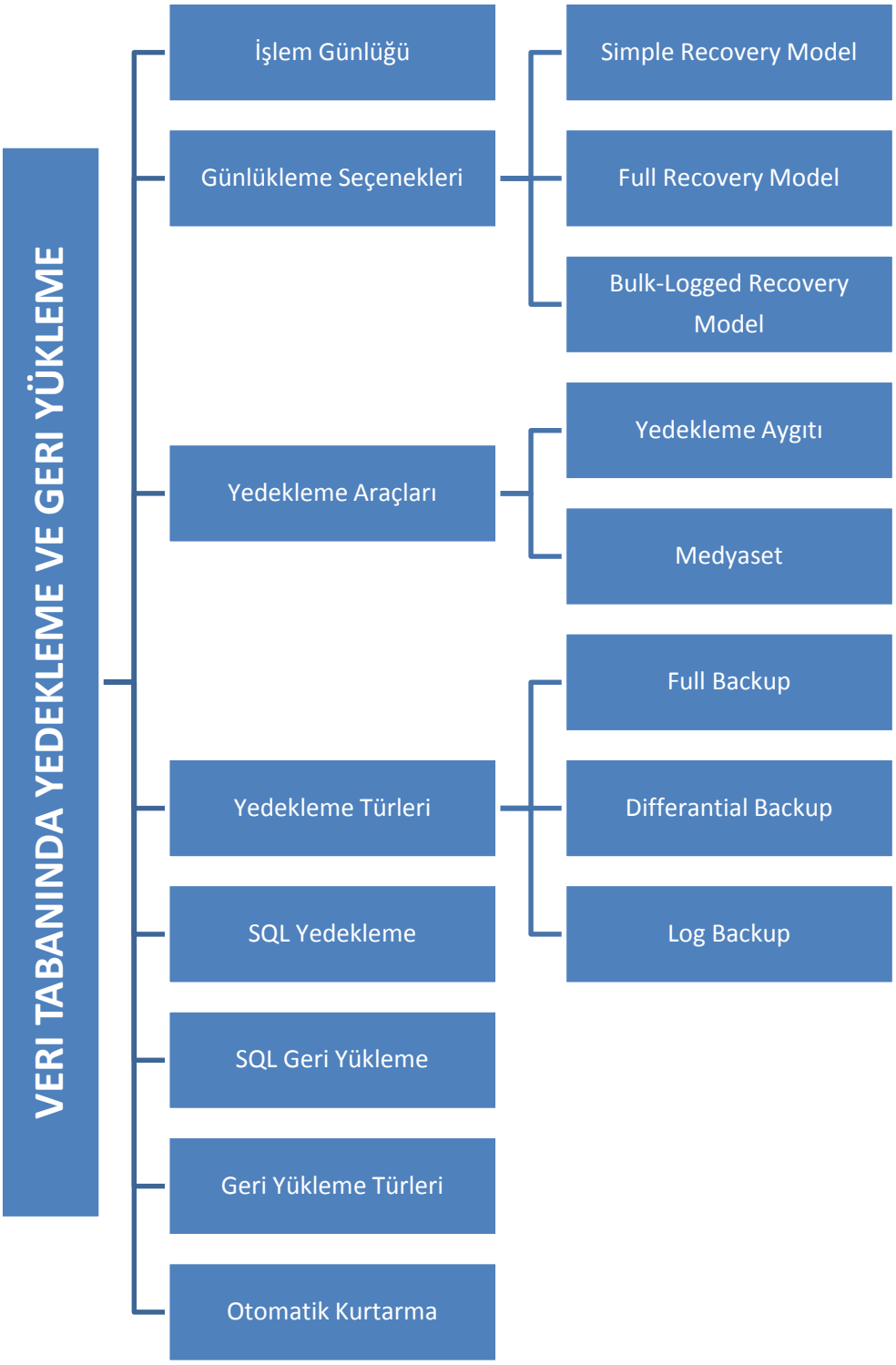
- Bu üniteyi çalıştıktan sonra;
  - Yedekleme konusunda temel bilgiler edinebilecek,
  - Planlı yedek alma ve manuel yedek alma konusunda bilgi sahibi olabilecek,
  - Geri yükleme konusunda temel bilgileri edinilebilecek,
  - Yedeği alınmış bir veri tabanını istediğiniz bir zamana geri döndürebileceksiniz.

Atatürk Üniversitesi
Açıköğretim Fakültesi

VERİ TABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi Ahmet Kamil KABAKUŞ

ÜNİTE
14


VERI TABANINDA YEDEKLEME VE GERI YÜKLEME
İşlem Günlüğü
Günlükleme Seçenekleri
Simple Recovery Model
Full Recovery Model
Bulk-Logged Recovery Model
Yedekleme Araçları
Yedekleme Aygıtı
Medyaset
Yedekleme Türleri
Full Backup
Differantial Backup
Log Backup
SQL Yedekleme
SQL Geri Yükleme
Geri Yükleme Türleri
Otomatik Kurtarma

GİRİŞ

Bundan önceki ünitelerde SQL Sunucu'nun etkin bir şekilde kullanılması, sorguların oluşturulması, analizi ve optimizasyonu, kullanıcı yetkileri ve SQL Sunucu Yönetim Stüdyosu'nun yönetimsel araçlarını etkin bir şekilde kullanma öğrenilmişti.

Bu ünitede öncelikle yedekleme ve geri yükleme hakkındaki temel bilgiler edinilecektir. Yedekleme ve geri yükleme günümüze IT sektörünü ve özellikle veri tabanı yöneticilerini çok yakından ilgilendiren bir konudur. Bilim ve teknolojinin ilerlemesi yedekleme ve geri yükleme noktasında farklı çözümler sunmaktadır.

Temel kavramların ardından yedek alma işlemini ayrıntıları ile öğrenip bir yedekleme görevini zamanlayarak düzenli yedekleme yapılmasını sağlama öğrenilecektir. Sunucular kesintisiz çalışmak için tasarlanmış olsa da meydana gelebilecek donanımsal hatalar sonucu sunucunun aniden kapanabilir. Böyle bir durumda otomatik kurtarma işlemi veri tabanı erişime açılmadan önce son kapanan işlem günlüğü noktasına kadar ki değişiklikleri yaparak veri tabanını kararlı bir hâle getirir.

Yedekleme işlemi çoğu proje için kullanıcıların inisiyatifine bırakılmayacak kadar önemlidir. SQL Agent servisinin yapılandırılarak SQL Server'in boşta kaldığı zamanlarda yedek alabilmesi için planları oluşturmayı ve bu planların uygulanabilmesi için SQL Agent servisinin başlatılması gerekliliği hakkındaki bilgileri edineceğiz.

Veri kayıpları işletmelere önemli ölçüde zarar vermekte ve para, zaman ve itibar kayıplarına neden olabilmektedir. Veri kayıplarının en büyük sebebi donanım kesintisi oluştururken fidye alma amaçlı yapılan siber saldırılar, doğal afetler, elektrik kesintileri ile dosya silme, yanlışlıkla hard diskin ya da sunucunun formatlanması gibi insan hataları da bu kayıpları yaratan diğer başlıca nedenler olarak sıralanmaktadır.

Yedekleme sürecinden sonra alınmış yedekleri geri döndürmek için gerekli temel bilgiyi edineceğiz ve son olarak yedeği alınmış bir veri tabanını yedekten istediğimiz bir zamana geri döndürmeyi öğreneceğiz. Yedekleme ve geri yükleme konusunda verilen örnekler için *SQL Server 2008 R2 Express Edition*, SQL Agent ve Maintance Plan oluşturma için verilen örneklerde ise *SQL Server 2008 R2 Enterprise Edition* kullanılmıştır.

Sistemin bulunduğu sürücü üzerine alınan yedeklerin hiçbir garantisi yoktur. Diskler arızalandığında aktif veriler ile birlikte yedeklerde kaybolacaktır.

VERİ TABANI YEDEKLEME VE GERİ YÜKLEME

Bu ünitede veri tabanı yedekleme *(Backup Database)* hakkındaki temel bilgileri edineceğiz. Ardından yedek alma işlemini ayrıntıları ile öğrenip bir yedekleme görevini zamanlayarak düzenli yedekleme yapılmasını sağlamayı öğreneceğiz. Daha sonra, alınmış yedekleri geri döndürmek için gerekli temel bilgiyi edineceğiz ve son olarak yedeği alınmış bir veri tabanını yedekten istediğimiz bir zamana geri döndürmeyi öğreneceğiz.

Verilerin saklandığı ortam olan veri tabanında meydana gelebilecek olas

veri kayıplarına karşı, önceden önlem almak gerekir. Alınan yedekler bu tür durumlarda hayati öneme sahip olacaktır. Alınmış bir yedeğin sisteme geri yüklenmesi de verilerin kaybolmaması için en az yedeklemek kadar önemlidir. Bu ünitede verileri yedeklemeyi ve yedeklenmiş verileri geri döndürmeyi öğreneceğiz.

Veri tabanı içerisinde tutulan veriler değerlidir. Tutulan veriler kaybedildiğinde tekrar oluşturmak çoğu zaman imkânsızdır ve kaybedilen veriler ciddi anlamda para ve zaman kaybına sebep olacaktır. Bu nedenle onları korumalı ve kaybetmemek için önlemler alınmalıdır. Veri kayıplarının dört temel nedeni vardır (Gözüdeli, 2010). Bunlar;

- Donanımsal Sorunlar
- Yazılımsal Sorunlar
- İnsan Kaynaklı Hatalar
- Doğal afetler olarak sıralanır.

Veri kayıplarını önlemenin en etkin yolu veri tabanının düzenli olarak yedeklenmesidir. Veri tabanının ne kadar sıklıkla ne türde ve nereye yedekleneceği veri tabanı sistemi ve veri tabanının kullanım şekliyle ilgilidir.

Yedeklemenin yapılacağı sürücü veya birim eldeki yedeklerin sağlıklı bir şekilde saklanması için önemlidir. Örneğin, sistemin bulunduğu sürücü üzerine alınan yedeklerin hiçbir garantisi yoktur. Çünkü disk üzerinde bir hata oluştuğunda aktif verilerle birlikte yedeklenen verilerde kaybolacaktır. Bu nedenle alınan yedeklerin harici depolama birimlerinde tutulması gerekmektedir. Veri tabanı sistemlerinin tamamında veri tabanın yedeklenmesi ve geri yüklenmesi çoğunlukla kullanılmaktadır. Bu işlem kullanılan veri tabanı sistemine göre farklılık göstermektedir.

Günümüzün teknolojik yenilikleri ve buna imkân sağlayan fiber optik alt yapı sayesinde hayati önem taşıyan veri tabanlarından senkron yedek alınıp saklanabilir ve daha uzaklara asenkron yedekler gönderilebilmektedir. Hatta aktif-aktif çalışabilen ve SQL cluster mimarisine uygun tasarlanan sistemlerde veri kayıplarını en aza indirerek gerektiğinde yük dengelemesi de yapılabilmektedir.

Uzun süre yedeklenmeyen hareket günlüğü, Trs-Log dosyasının çok fazla büyümesine neden olabilir. Log dosyasını yedeklemeniz hâlinde dosya boyutu küçülecektir.

SQL Sunucu'da veri kaybını en aza indirmek için ve kaybolan veriye tekrar ulaşabilmek için veri tabanlarını düzenli olarak yedeklemek gerekmektedir. Veri tabanını kurup üstünde hayati bir projeyi çalıştırmaya başladığınızda veri kaybetmeye neden olabilecek bir çok durumu göz önünde bulundurulmalıdır. Tutulan veriler kaybedildiğinde tekrar oluşturmak çoğu zaman imkânsızdır ve bu nedenle onları korumalı ve kaybetmemek için önlemler alınmalıdır. Veri tabanı için yedekleme planları oluşturma ve yedekleme bu önlemlerin başında gelmektedir.

İŞLEM GÜNLÜĞÜ

Bilişim sözlüklerinde Türkçe karşılığı İşlem Günlüğü (Transcaction Log) olan ve veri tabanlarında yapılan her işlemin kaydının tutulduğu günlük dosyasına *işlem günlüğü* denir. Veri tabanında yapılan her ekleme değiştirme işleminin kaydı tutularak sonradan olan bir hata esnasında geri sarım yapılması ve istenilen bir zamana geri dönülmesi için gerekli olan günlük (log) dosyasıdır. İşlem günlüğü

bölüm içerisinde, dipnot ve örneklerde Trs-Log olarak kısaltılacaktır.

SQL Sunucu'da açılan her bir veri tabanı için ayrı birer İşlem Günlüğü bilgisi tutulur ve işlem günlüğüne yansımamış hiçbir değişiklik veri dosyasına yansıtılmaz. Verilerin okunması dışında *INSERT, DELETE, UPDATE* ifadeleri ile veri tabanı şemalarına ait değişiklikler, bu kapsamdadır. Bu aşama şeklinden dolayı İşlem Günlüğü dosyasına ilk önce yazılan günlük *(Write-Ahead Logging - WAL)* denir.

Bölüm içerisinde ilerledikçe aldığımız yedekten istenilen bir ana dönülebilmesi için *İşlem Günlüğü* dosyasının ne kadar önemli olduğu anlaşılacaktır. SQL Sunucu'da yer alan veri tabanımızı herhangi bir zamana döndürmeniz gerekirse İşlem Günlüğü dosyalarınıza gereksinim duyarsınız. İşlem Günlüğü dosyaları, İşlem Günlüğü yedeğiniz veya yedeklemeye müsait dosyalarınız yoksa elinizde tam yedek (full-backup) olsa bile istediğiniz bir zamana verileri döndüremezsiniz. Aksi durumda verilerinizi sadece tam yedeğin alındığı zamana döndürebilirsiniz.

GÜNLÜKLEME SEÇENEKLERİ

Her bir veri tabanı için işlem günlüğü dosyasına günlüklerin seviyeleri ayarlanabilir. Basit (Simple), Toplu Oturum (Bulk Logged) ve Tam (Full) Kurtarma Modeli olmak üzere 3 farklı günlükleme modeli mevcuttur (Gözüdeli, 2010). Günlükleme modelini veri tabanının kritikliği belirler. SQL Sunucu'da oluşturulan yeni bir veri tabanı için varsayılan seviye *Tam Kurtarma Modeli* 'dir. Bu seviyede her değişim kayıt altına alınarak günlüklenir (Server, 2008). Basit Kurtarma Modeli'nde ise sadece verilerin tutarlılığını garanti edecek kadar günlük kayıt altına alınır.

Günlükleme dosyalarının büyüklüğüne göre kurtarma modelleri sıralandığında *Full, Bulk-logged, Simple* şekline büyükten küçüğe doğru sıralanır.

Basit Kurtarma Modeli (Simple Recovery Model)

Sadece okunan veri tabanları, test veri tabanları gibi değişimi ve veri kaybı önemli olmayan veya belli bir zamana dönme zorunluluğu gerektirmeyen veri tabanları için tercih edilebilir. Bu modda alınan bir veri tabanının günlük dosyası belli bir boyutun üstüne çıkmaz. Günlük bilgisini yedekleme gereksinimi yoktur.

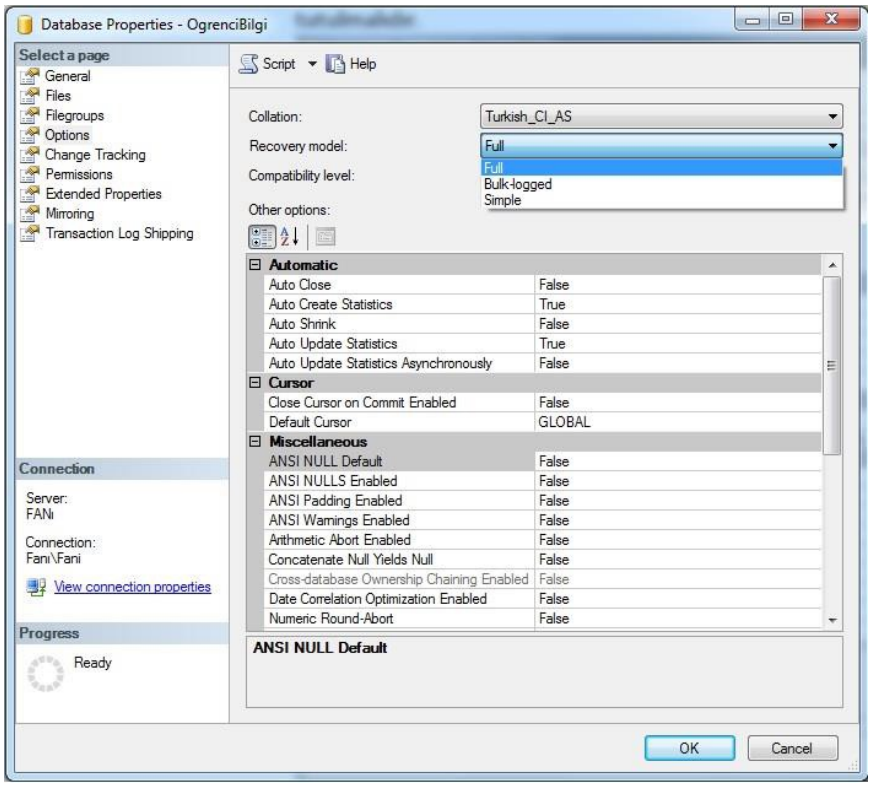
Günlükler sadece 'uygulandı' olarak işaretlenene kadar tutulur. Aktifliği biten günlükler silinir. Bakımı kolaydır. Ancak herhangi bir zamana dönmek mümkün değildir. Sadece *tam veri tabanı yedeği* (full database backup) alınan zamanlara dönülebilir. Kritik uygulamalar bu modda tutulmamaktadır.

Tam Kurtarma Modeli (Full Recovery Model)

Bütün değişimler günlüğe yansıtılır. Yedek alınana kadar günlükler bekler. İstenilen noktaya dönülebilir. Ancak uzun süre günlükler yedeklenmezse günlük dosyaları aşırı büyüyebilir. Bu modda yedek alınan işlem günlüğü yedekleri arasında kayıp yapılmaması çok önemlidir. Zincirin kırılması (aradan bir yedeğin kaybolması) hâlinde herhangi bir zamana dönmek imkânsızlaşacaktır. Bu modda, tam veri tabanı yedekleme *(full database backup)* işleminin yanı sıra özellikle çok önemli bir işlem SQL Sunucu'da tanımlanacak bir zamanlanmış görev ile günlüklerin belli aralıklarla yedeklenmesi yararlı olacaktır.

Toplu Oturum Kurtarma Modeli (Bulk-Logged Recovery Model)

Bu model, Basit ile Tam kurtama modelin arasında kalan bir modeldir. *SELECT INTO*, bcp ile yapılan yüklemeler, *CREATE- ALTER INDEX REBUILD-DROP INDEX* gibi toplu işlemler günlüğe düşmez. Şayet veri tabanı üstünde Aynalama (Mirroring) gibi bir günlük dosyası üstünden çalışan servis yoksa Tam Kurtarma ile Toplu Oturum arasında yukarıda sayılan durumlarda günlük dosyasının boyutunu fazla uzatmamak için geçiş yapılabilir. Bu türden bir servis varsa (Aynalama) veri tabanı *Tam Kurtarma Modeli* 'nde tutulmalıdır.



Şekil 14.1. Veri Tabanına Ait Günlükleme Seviyesi

SQL Sunucu Yönetim Stüdyosu üzerinden OgrenciBilgi veri tabanını sağ tıklayıp özellikler *"Properties"* tıklandığında, seçenekler *"Options"* kısmında Kurtarma Modeli *"Recovery model"* karşısında *Full, Bulk-logged ve Simple* olarak seçenekler karşımıza çıkmaktadır. Bu seçenekler veri tabanının günlükleme seviyeleri olarak da isimlendirilebilir (Burma, 2009).

T- SQL ifadeleri ile bir veri tabanının günlükleme seviyesi şu şekilde ayarlanabilir:

Örnek

```
•USE master
•ALTER DATABASE OgrenciBilgi SET RECOVERY FULL;
```

Sadece *tam kurtarma modeli* seçeneği aktif veri tabanlarımız, herhangi bir hata anında istediğiniz zamana döndürebilecek detayda günlük üretebilir. Üretilen günlükler yedeklenmediği sürece, tam kurtarma modunda çalışan bir veri tabanınızın olması, istediğiniz zamana dönme imkânını size sağlamaz.

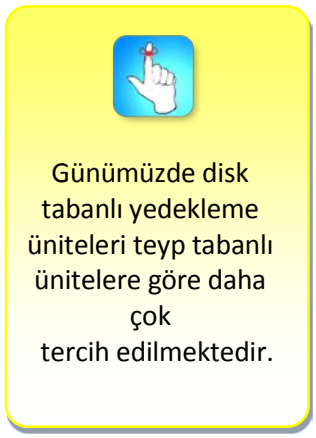
SQL SUNUCU OTOMATİK KURTARMA İŞLEMİ

Her ne kadar sunucular sürekli çalışmak için tasarlansa da elektrik kesilmesi, güç kaynağı bozulması, işlemcilerin yanması, RAM hataları gibi sorunlar sunucunun kapanmasına neden olabilir. İstem dışı bir kapanma sonucunda, SQL Server Buffer Cache ve Log Buffer gibi RAM bölgelerinde yer alan veriler kaybedilir. Bu durumda, SQL Sunucu'nun henüz İşlem Günlüğüne kaydedemedigi verilerle ilgili otomatik kurtarma işlemi *(SQL Server Automatic Recovery)* SQL Sunucu Servisinin çalışması ile devreye girer (Gözdüdeli, 2010). Bu işlem sayesinde veri tabanı erişime açılma öncesinde, SQL Sunucu tarafından harici bir müdahale gerekmeksizin son kapanan işlem noktasına kadarki değişiklikler yapılarak veri tabanı kararlı hâle getirilir.

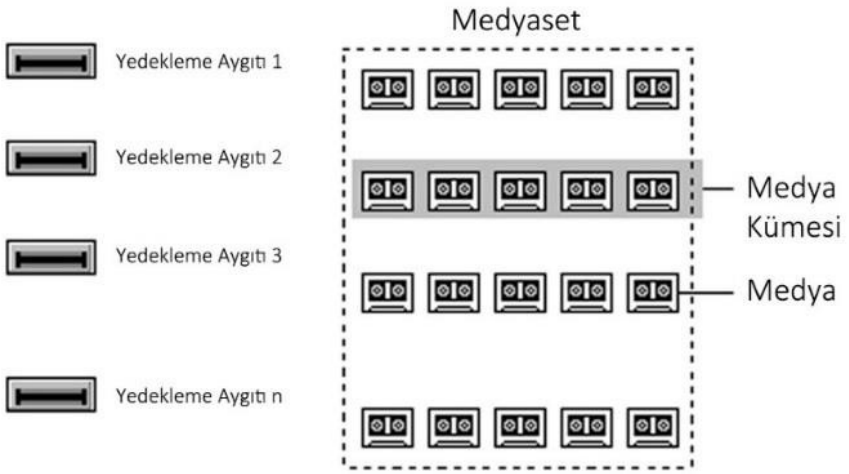
Otomatik Kurtarma, öncelikle Günlük dosyasına göre *COMMIT* olan ama diske yansıtılmamış işlemler varsa ileri sarma *(ROLLFORWARD)* işlemine tabi tutar ve günlükleri 'uygulandı' olarak işaretler. Bütün günlükler bittiğinde açık kalan işlemler varsa (Bir işlem *COMMIT* ile sonlanmamışsa açık kalmış demektir.) geri sarım *(ROLLBACK işlemi)* işlemine tabi tutulur. Böylelikle son biten sağlıklı işlem noktasına veri tabanı geri döndürülmüş olur ve kararlı bir noktaya getirilmiş olur.

YEDEKLEME ARAÇLARI

Her ne kadar SQL Sunucu Yönetim Stüdyosu veya benzeri araçlarla kolayca yedek almak mümkün olsa da SQL Sunucu'nun yedekleme mimarisini daha iyi anlayabilmek için öncelikle Yedekleme Aygıtı *(Backup Device)* ve Medyaset kavramlarını anlamakta fayda var.



Günümüzde disk tabanlı yedekleme üniteleri teyp tabanlı ünitelere göre daha çok tercih edilmektedir.



Şekil 14.2. Medyaset, Yedekleme Aygıtı Ve Medya

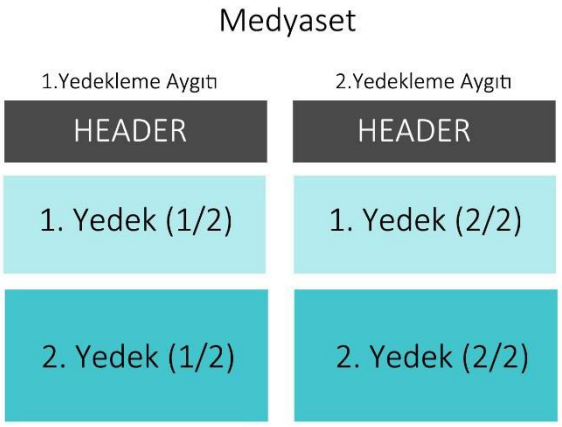
Yedekleme Aygıtı

İçerisine SQL Sunucu'nun yedekleri yazabileceği manyetik veri saklama ortamı ile bu ortamı SQL Sunucu açısından tanımlayan yapıya Yedekleme Aygıtı (Backup Device) denir (Gözüdeli, 2010). Yedekleme aygıtları yedekleri

kaydedebilmek için bir ortama *(Medya)* gereksinim duyar. Yedekler genellikle disk veya kaset *(tape)* türünden ortamlarda saklanır.

Medyaset

Medyaset (Media set), yedekleme aygıtlarını gruplayarak yönetmek için kullanılır. Bir veya daha fazla yedekleme aygıtı içerebilir. Ancak birden fazla yedekleme aygıtı içerirse tiplerinin tamamı aynı medya tipinden *(disk/tape)* olmalıdır. SQL Sunucu, yedekleme işlemi yaparken yedeği bir medyasette yer alan bütün aygıtlara paylaştırarak yazar. Yani yedek verileri medyasette birden fazla yedek aygıtı varsa hepsine dağıtılmış olarak kaydedilir.



Şekil 14.3. Yedekleme Aygıtlarına Dağıtılmış Yedekler

Bir medyaset birden fazla yedekleme aygıtını bir tek ortammış gibi kullanabilmeye yarayan yapıdır.

SQL Sunucu yedekleme yaparken yedekleme aygıtını, varsayılan olarak ilk yedekleme işleminde formatlayarak başlık (header) ekler. Sonra içerisine yedek ekler. Şayet aynı aygıta ikinci bir yedek alınırsa varsayılan olarak ikinci yedek aynı dosyanın devamına eklenir. Ancak yedekleri alt alta eklemeden var olanların üstüne yazmak da mümkündür.

Bir medyaset birden fazla yedekleme aygıtını bir tek ortammış gibi kullanabilmeye yarayan yapıdır. Şekilde 1. yedek ve 2. yedek iki farklı aygıta dağıtılarak yazılmıştır. Aygıtlardan ikisi birlikte okunamadığı sürece tüm yedek okunamaz.

Yansıtılmış Medya Set

Yansıtılmış medya set (Mirrored media set) bir yedek medya setinin birden fazla kopyaya saklanmasına olanak sağlayan bir yapıdır. Bir medya setinin yansıması (mirror) da medya seti ile aynı karakteristiğe sahip olmalıdır. Bu durumda, medya setlerden herhangi biri zarar görürse diğer yansımadaki medya seti ile bu zarara rağmen yedeklerden geri dönülebilir.

YEDEKLEME TÜRLERİ

SQL Sunucu'da Tam Veri Tabanı Yedeği *(Full Database Backup)*, Fark Yedeği *(Differential Backup)* ve Günlük Yedeği *(Log Backup)* olmak üzere üç tür yedekleme yapılabilir (Server, 2008).

Tam Veri Tabanı Yedeği (Full Database Backup)

Bu yedekleme türü veri tabanının tamamını yedeklemek için kullanılır. Bu tür yedeklemede uygulanmamış işlem günlüklerinin de yedeği alınacaktır. Diğer yedekleme türleri için başlangıç noktası olduğu için önemlidir. Yani, diğer yedekleme türlerini kullanabilmek için veri tabanı bir kez tam veri tabanı yedeği alınmış olması gerekmektedir. Yoğun işlem işlemlerinin olmadığı küçük veri tabanlarında ve kısa süreli veri kaybının önemli olmadığı veri tabanlarında tercih edilmektedir.

Tam veri tabanı yedeği diğer iki yedekleme türünün başlangıç noktasını oluşturacağı için ilk olarak Tam Veri tabanı yedeği alınmalıdır.

Fark Yedeği (Differential Backup)

En son alınan tam yedeklemeden sonra yapılan değişikliklerin alındığı yedekleme türüdür. Bu tür yedeklemeyi kullanabilmek için daha önce *Tam Veri Tabanı Yedeği* alınmış olmalıdır. Sadece veri tabanında meydana gelen değişiklikleri alacağı için bu yedekleme türü *Tam Veri Tabanı Yedekleme* türüne göre daha hızlı ancak geri yükleme işlemi daha uzundur. Bu yedekleme türünde *işlem günlüğü* bilgileri yedeklenmez.

Günlük Yedeği (Log Backup)

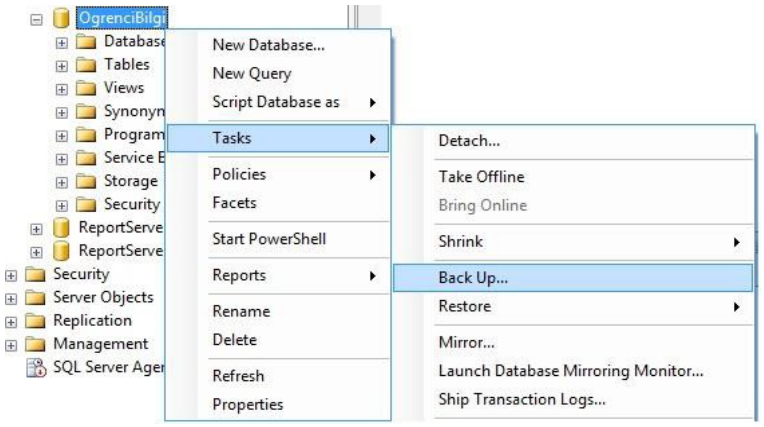
Sadece işlem günlükleri (transaction log) yedeğini alır. Bu yedekleme türünü kullanabilmek için daha önce *Tam Veri Tabanı Yedeği* alınmış olmalıdır. Bu yedekleme ile herhangi bir andaki noktaya geri dönüş yapılabilir. Veri kaybı riskinin fazla olduğu ve yoğun şekilde hareket işlemi gerçekleşen veri tabanlarında belirli aralıklarla bu tür yedeklemenin yapılması önemlidir. Veri tabanı kurtarma modu *Basit Kurtarma Modeli* olarak tanımlıysa bu tür yedekleme kullanılamaz. Bu tür yedekleme sadece *Tam* ve *Toplu Oturum* kurtarma modelinde kullanılabilir.

SQL Sunucu'nun son çalışma anına ait günlük bilgilerini yedeklemek suretiyle elde edilen günlük yedeğine Kuyruk günlük yedeği *(Tail Log Backup)* denir. Kuyruk günlük yedeği, herhangi bir hata anında belli bir zamana dönmek için gereklidir.

SQL SUNUCU'DA YEDEKLEME

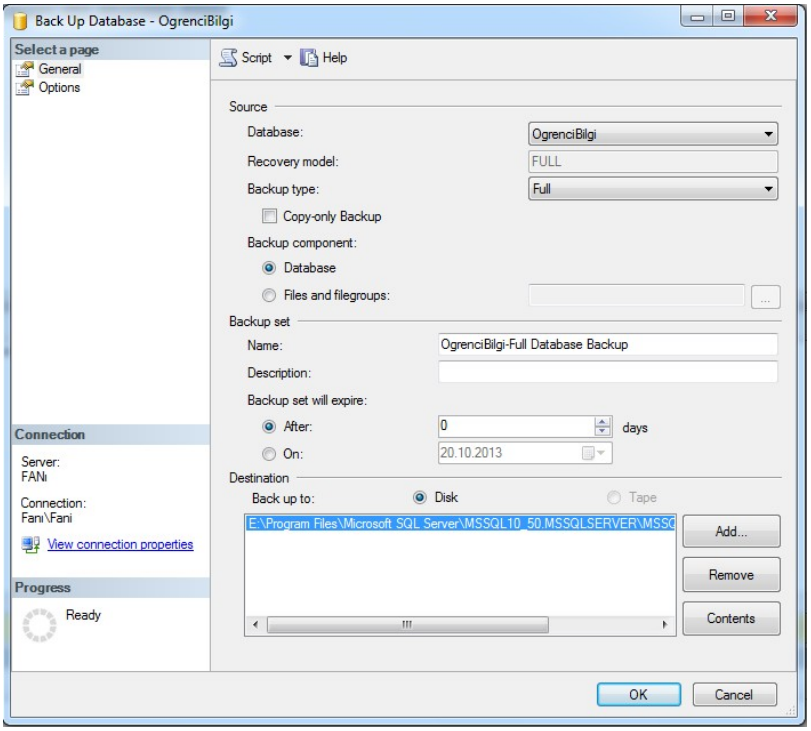
SQL Sunucu Yönetim Stüdyosu ile yedekleme yapmak için işlem basamakları aşağıda verilmiştir.

Yedekleme yapılacak veri tabanına sağ tıklayarak açılan menüden görevler *"Tasks" >* yedekleme *"Back Up"* seçeneği seçilir.



Şekil 14.4. Ogrencibilgi Veri Tabanı Yedekleme

Yedekleme *"Back up"* seçildiğinde aşağıdaki yedekleme penceresi açılacaktır.



Şekil 14.5. Ogrencibilgi Veri Tabanı Yedekleme/Genel Seçenekleri

Daha önce alınmış Tam Veri tabanı Yedeği olmadan fark yedeğinden geri dönmek mümkün olmaz.

Database: Yedeklemesi yapılacak veri tabanını seçmek için kullanılır.

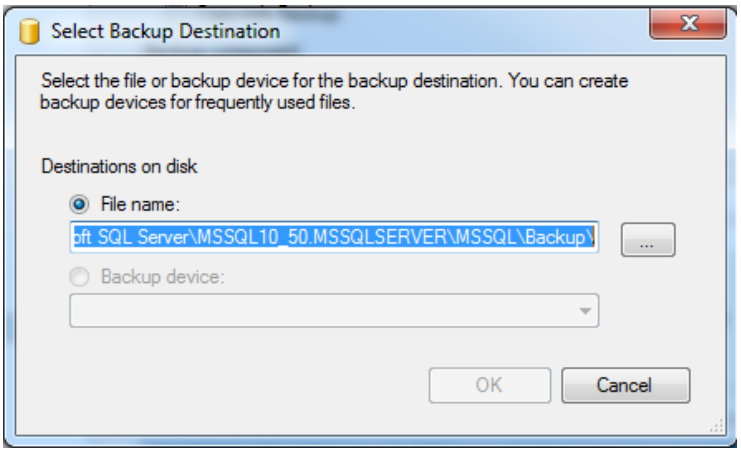
Backup Type: Hangi yedekleme türü kullanılacağını seçmek için kullanılır.

Name: Yapılacak yedeklemeye isim vermek için kullanılır.

Description: Yedekleme hakkında açıklama eklemek için kullanılır.

Backup set will expire: Alınan yedeğin kaç gün veya hangi tarihe kadar geçerli olacağını belirlemek için kullanılır.

Destination: Yedeklemenin yapılacağı yeri veya sürücüyü belirlemek için kullanılır. Eğer sunucu üzerinde sürücü tanımlaması yapıldı ise tanımlanan sürücüye yedekleme işlemi de gerçekleştirilebilir. Varsayılan olarak SQL Sunucu'nun yüklü olduğu klasör yedekleme için kullanılacaktır. Yani yedekleme konumu veya sürücü belirlemek için ekle *"Add"* butonuna tıklandığında aşağıdaki pencere açılacaktır.

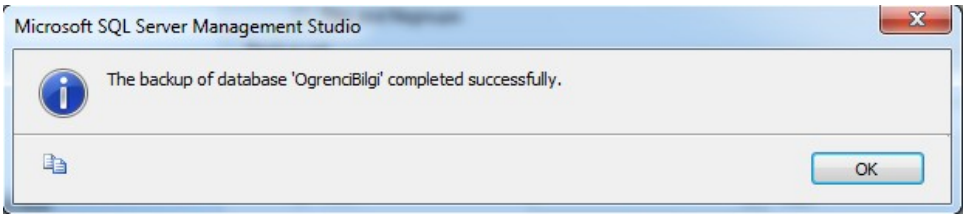


Şekil 14.6. Ogrencibilgi Veri Tabanı Yedekleme Hedefi

File Name: Yedeklemenin nereye ve hangi isimle yapılacağını belirlemek için kullanılır. Yan tarafında bulunan butona tıklanarak değişiklik yapılabilir.

Backup Device: Yedekleme daha önce oluşturulan bir sürücü üzerinde yapılacaksa buradan tanımlı olan sürücüler seçilebilir.

Ok butonuna tıklandığında yedekleme işlemi başlayacaktır ve yedekleme sonrasında aşağıdaki gibi pencere çıkacaktır.



Şekil 14.7. Ogrencibilgi Veri Tabanı Yedek Alma Işlemine Ait Bilgi Ekranı

Bir veri tabanının birden fazla dosyaya dağıtarak yedeklemek mümkündür. Ancak bu yedeklenen dosyalardan biri bile bozulursa yedek geri okunamayacaktır.

Yedekleme T-SQL

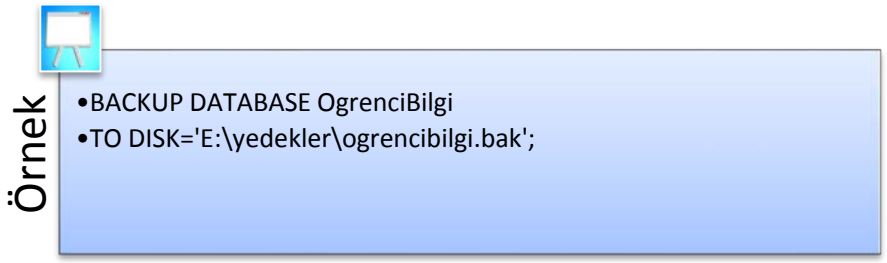
Herhangi bir veri tabanının yedeğini almak için şu genel ifade kullanılır:

BACKUP DATABASE veritabani

TO DISK ='disk dosya yolu'

[TO DISK ='ikinci aygit yolu',....] [WITH MEDIANAME='medyaset ismi'];

Yedek Almak Ve Yedekleri Listelemek



Yukarıdaki örnekte OgrenciBilgi veri tabanının tam veri tabanı yedeğini alalım.

Normalde, alınan her yedek, aynı dosyaya yazılsa bile dosyadaki yedeklerin devamına eklenir. Bu yedeklerin uzun süre, korunması için tercih edilebilir. Ancak bazen yer sıkıntısı gibi nedenlerle yedeğin eski yedekleri silerek alması gerekebilir. Bu türden bir yedeklemede eski yedek dosyanızı başka bir yere kopyalamış olmanız, sizi olası yedeksiz kalma sorunlarından kurtaracaktır.

Örnek

```
•BACKUP DATABASE OgrenciBilgi
•TO DISK='E:\yedekler\ogrencibilgi.bak' WITH INIT;
```

Bir sonraki alınan yedeğin bu yedek dosyasına eklenmek yerine var olan yedeğin üstüne yazması için yukarıda ki ifadeyi kullanabiliriz:

Bir veri tabanının birden fazla dosyaya dağıtarak yedeklemek mümkündür.

Örnek

```
•BACKUP DATABASE OgrenciBilgi
•TO DISK= 'E:\yedekler\ogrencibilgi1.bak ',
•DISK= 'G:\yedekler\ogrencibilgi2bak '
•WITH MEDIANAME = 'ogrencibilgi_disk_media' ;
```

OgrenciBilgi veri tabanının yedeğinin birden fazla dosyaya yazdırılmasını istersek yukarıdaki gibi bir ifade kullanabiliriz. Ancak bu yedeklenen dosyalardan biri bile bozulursa yedek geri okunamayacaktır. Birden fazla dosyaya yayılmış yedeklerin başka bir kopyasını aynı sayıda ve tipte dosyaya dağılmış olarak almak için aynalanmış medya set (Mirrored Media Set) yedekleme seçeneği kullanılır.

OgrenciBilgi veri tabanının iki diske yazılacak yedeğinin bir kopyasını daha oluşturmak istersek:

Örnek

```
•BACKUP DATABASEOgrenciBilgi
•TO DISK='E:\yedekler\ogrencibilgi1.bak',
•TO DISK='G:\yedekler\ogrencibilgi2.bak'
•MIRROR TO DISK='\\10.102.4.88\yedekler\ogrencibilgi1.bak',
•TO DISK='\\10.102.4.88\yedekler\ogrencibilgi2.bak'
•WITH MEDIANAME='ogrencibilgi_mirrored_medya_set';
```

Yukarıdaki örnekteki gibi bir ifade çalıştırılabilir.

Alınan bu yedekte aynalanan (mirror) dosyalardan herhangi biri bozulursa diğeri ile yedek okunabilir. Örneğimizde sadece iki kopya var ancak aynalanmış medya setler (mirrored media set) birden fazla kopya almaya izin verir.

Yedekleme İle Ilgili Katalog Bilgilerini Listelemek

Yedekleme ile ilgili bilgiler (yedek alınan ortamlar ve yedek alma zamanları gibi) msdb veri tabanında tutulur (Server, 2008). Yedekleme ile ilgili bilgiler içeren belli başlı tablolar şunlardır:

- Backupfile
- Backupfilegroup
- Backupset
- Backupmediaset
- Backupmediafamily

Alınan yedeklerin türü ve alınma tarihi gibi bilgileri görmek için aşağıdaki örnekteki sorgu kullanılabilir.

Birden fazla dosyaya dağıtılarak alınan yedekleri aynalaması yapılırsa dosyalardan biri bozulursa aynalanmış kısımdan bozulan kısım okunabilir.

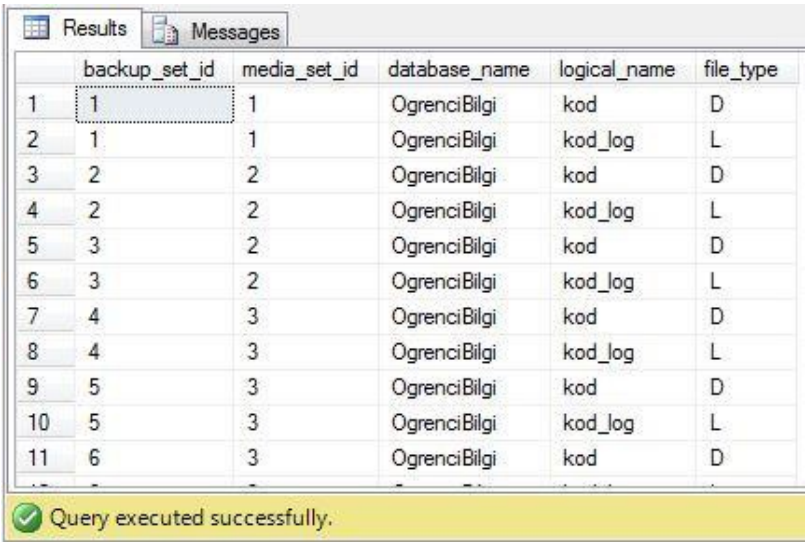
Çoğu proje için, düzenli veri tabanı yedeği alma işlemi, bir veri tabanı yöneticisinin insiyatifine bırakılmayacak kadar önemlidir.

Örnek

```
•SELECT b.backup_set_id, b.media_set_id, b.database_name,
 f.logical_name, f.file_type
•FROM msdb.dbo.backupset b LEFT JOIN msdb.dbo.backupfile f
 •ON b.backup_set_id = f.backup_set_id
```

Tablo 14.1. Yedek Tipleri Ve Anlamları

File_type Sütunu	Anlamı
D	Data Yedeği, Tam Veri Tabanı Yedeği
F	Fulltext, dosya, dosya grubu yedeği
I	Fark yedeği
L	Log yedeği



Results | Messages

	backup_set_id	media_set_id	database_name	logical_name	file_type
1	1	1	OgrenciBilgi	kod	D
2	1	1	OgrenciBilgi	kod_log	L
3	2	2	OgrenciBilgi	kod	D
4	2	2	OgrenciBilgi	kod_log	L
5	3	2	OgrenciBilgi	kod	D
6	3	2	OgrenciBilgi	kod_log	L
7	4	3	OgrenciBilgi	kod	D
8	4	3	OgrenciBilgi	kod_log	L
9	5	3	OgrenciBilgi	kod	D
10	5	3	OgrenciBilgi	kod_log	L
11	6	3	OgrenciBilgi	kod	D

Query executed successfully.

Şekil 14.8. Ogrencibilgi Veri Tabanı Yedeklerin Listelenmesi

Yedekleme Stratejileri

SQL Sunucu'da hangi aralıklarla yedek alınacağı, yedeklerin ne türden ortamlarda saklanacağı, fiziksel olarak nerede tutulacağı gibi konuların olası hasar veya felaket senaryolarında verileri yedekten geri yüklemeye olanak sağlayacak şekilde yapılandırılmasına yedekleme stratejisi denir.

Orta ölçekli bir şirket için verinin büyüklüğü ve kritikliğine göre haftada 1 tam yedek, çarşamba günleri fark yedek, her saat başı günlük yedeği almak, herhangi bir zamana verileri geri döndürmek için yeterli olacaktır. Genellikle haftada bir alınan tam yedekler, veri tabanı yükünün en az olduğu pazar günleri gibi boş günlere planlanarak sistemin rahat çalışması sağlanabilir. Ayrıca, tam yedekler için de yine yük seviyesinin en az olacağı gece yarısı gibi zaman dilimleri seçilebilir. Elbetteki bir veri tabanı yöneticisinin geceleri işe gidip yedek alması düşünülemez. Bu iş için SQL Sunucu Yönetim Stüdyosu ile erişilebilen Bakım Planı (Maintenance Plan) arayüzü kullanılarak SQL Sunucu Aracısı (SQL Server Agent) servisi tarafından otomatik yedek alınması planlanabilir.

Ayrıca, SQL Sunucu sistem veri tabanlarının da yedeğini almak gerekebilir. Sistem veri tabanlarından master ve msdb veri tabanlarının sık sık yedeklemek gerekir. Model veri tabanında bir değişiklik yapılması halinde yedeklemek yeterli olacaktır. Resources veri tabanı sadece okunan bir veri tabanı olduğundan yedeklemeye gerek duyulmaz.

Tamamen yedekleme kapsamında değerlendirilmese de kritik verilerle çalışan bir uygulama için yangın, deprem, sel baskını gibi durumları da içeren felaket hâlinde yedekten dönme senaryoları için, yedeklerin farklı bir fiziksel lokasyon da saklanması da faydalı olacaktır. Bu durumda en basit uygulama aynalanmış medya set ile farklı noktalara doğrudan yedek alınabileceği gibi yedekler farklı lokasyonlara ağ üstünden elle de kopyalanabilir.

SQL Sunucu'nun boşta kaldığı tahmin edilen gece yarıları, hafta sonları gibi zamanlarda yedek alabilmesi için SQL Sunucu Aracısı (SQL Server Agent) adı verilen servisi yapılandırabilirsiniz.

Ayrıca, SQL Sunucu'nun günlük aktarma *"log shipping"* adı verilen özelliği ile de yedekler uzak bir bölgeye otomatik olarak aktarılıp tekrardan bir veri tabanına uygulanarak veri tabanın belli bir süre fark ile yaklaşık aynısı oluşturulabilir.

Bunların dışında üçüncü parti yazılımlar ve disk üreticilerinin sağlamış olduğu imkânlar ile aktif-aktif veya aktif-pasif yedeklemeler yapılabileceği gibi sanallaştırma alt yapısının sağlamış olduğu imkânlar yedekleme stratejileri oluşturulurken değerlendirilebilir ve herhangi bir felaket anında sistemin yani veri tabanının kesintisiz bir şekilde çalışması sağlanabilir.

GERI YÜKLEME TÜRLERİ

Geri yükleme işlemi, herhangi bir sorunla karşılaşıldığında alınan yedeklerin veri tabanına geri yüklenerek tekrar kullanılabilmesini sağlar.

Geri yükleme SQL Sunucu Yönetim Stüdyosu veya T-SQL ile erçekleştirilebilir. Geri yükleme işlemi öncesinde alınan yedekler hakkında bilgi almak için aşağıdaki T-SQL komutları kullanılabilir (Burma, 2009).

Restore Headeronly

Alınan yedeğin veri tabanı adı, yedek dosyası adı, yedekleme ortamı, yedekleme türü (Full, differential, transaction log), yedekleme tarihi, yedek boyutu gibi bilgilerini gösterir.

```
RESTORE HEADERONLY FROM disk ='E:\yedekler\ogrencibilgi.bak';
```

Results | Messages

	BackupName	BackupDescription	BackupType	ExpirationDate	Compressed	Position	DeviceType
1	NULL	NULL	1	NULL	0	1	2
2	OgrenciBilgi-Full Database Backup	NULL	1	NULL	0	2	2

Şekil 14.9. Ogrencibilgi Veri Tabanına Ait Yedeklerin Headeronly Bilgilerinin Listelenmesi

Restore Filelistonly

Yedek dosyasının hangi veri tabanı ve transaction log dosyasına ait olduğu bilgisi döndürür.

```
RESTORE FILELISTONLY FROM disk ='E:\yedekler\ogrencibilgi.bak';
```

Results | Messages

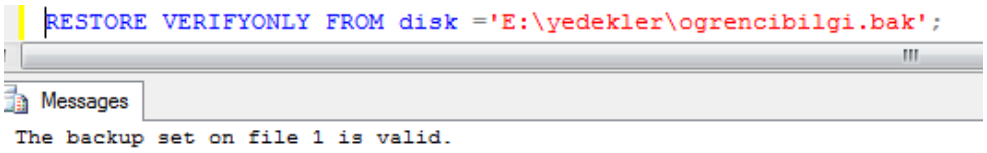
	LogicalName	PhysicalName	Type	FileGroupName	Size	MaxSize
1	kod	E:\Program Files\Microsoft SQL Server\MSSQL10_50....	D	PRIMARY	3145728	35184372080640
2	kod_log	E:\Program Files\Microsoft SQL Server\MSSQL10_50....	L	NULL	1048576	2199023255552

Şekil 14.10. Ogrencibilgi Veri Tabanına Ait Yedeklerin Log Ve Datalarının Listelenmesi

Restore Verifyonly

Yedek dosyasından herhangi bir sorun olup olmadığını kontrol etmek için kullanılır. Eğer yedek dosyasında herhangi bir sorun yoksa “The backup set on file

… is valid” mesajı dönecektir.



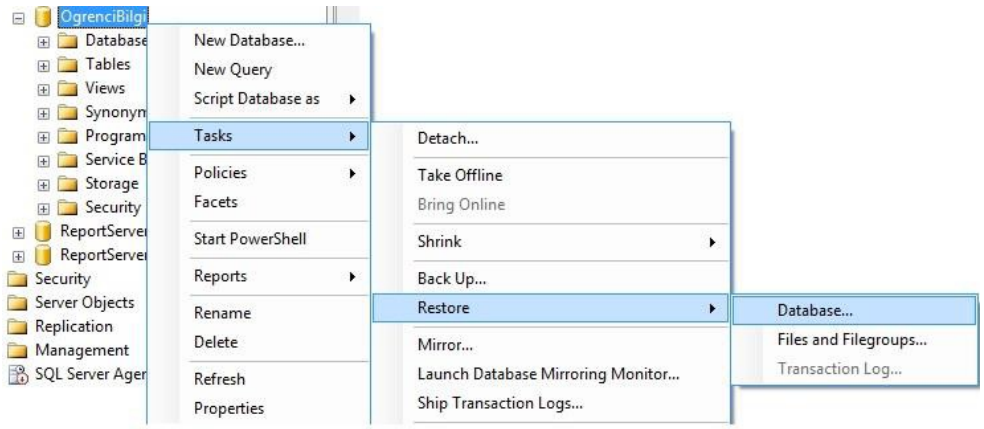
Şekil 14.11. Ogrencibilgi Veri Tabanına Ait Yedeklerinde Sorun Olup Olmamasının Sorgulanması

SQL SUNUCU'DA GERI YÜKLEME

Geri yükleme işlemini SQL Sunucu Yönetim Stüdyosu ile yapmak için aşağıdaki işlem basamakları izlenmelidir (Gözüdeli, 2010):

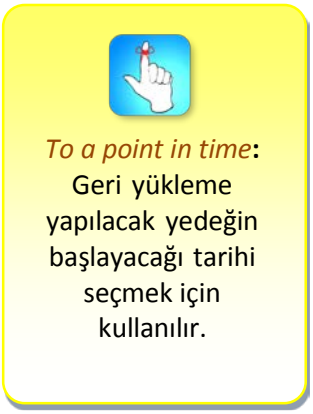
Geri yükleme yapılacak veri tabanına sağ tıklayarak Görevler *"Tasks"* – Geri yükle

"Restore" – Veri Tabanı *"Database"* seçeneği seçilir.

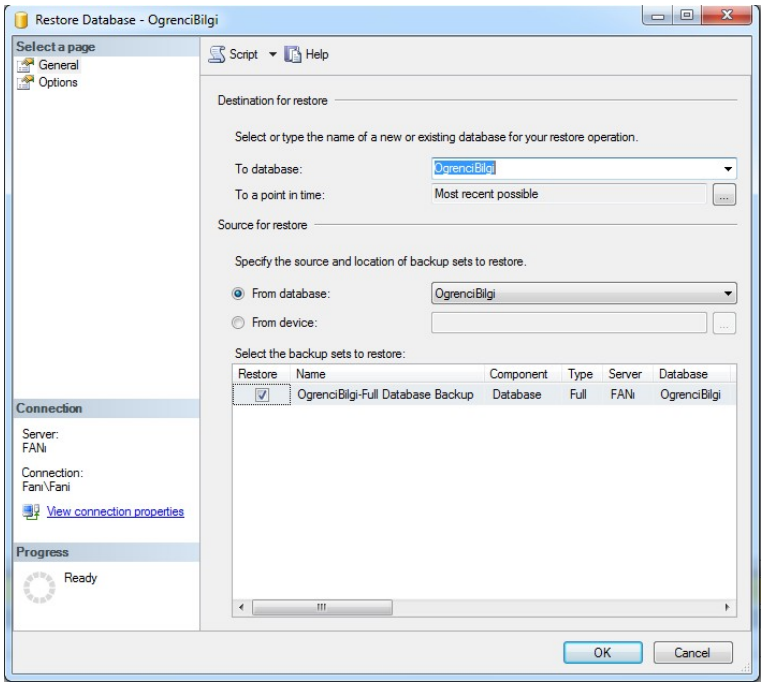


Şekil 14.12. Ogrencibilgi Veri Tabanına Ait Yedeklerinde Sorun Olup Olmamasının Sorgulanması

Geri yükleme seçildikten sonra aşağıdaki Veri Tabanını Geri Yükle *"Restore Database"* penceresi açılacaktır. Bu pencere üzerinden geri yüklemenin hangi veri tabanına yapılacağı gibi seçenekler vardır (Server, 2008).



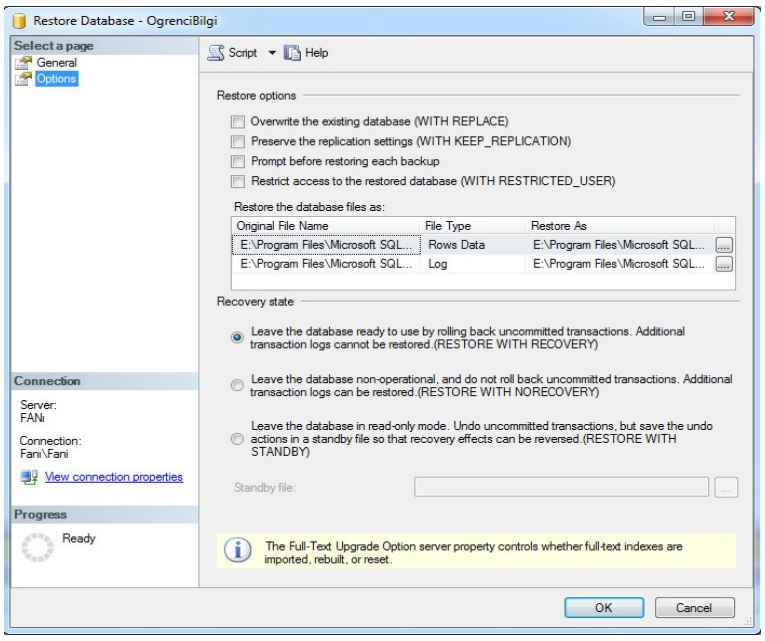
To a point in time: Geri yükleme yapılacak yedeğin başlayacağı tarihi seçmek için kullanılır.



Şekil 14.13. Ogrencibilgi Veri Tabanına Ait Yedeklerinde Sorun Olup Olmamasının Sorgulanması

To Database: Geri yüklemenin yapılacağı veri tabanını seçmek için kullanılır.

To a point in time: Geri yükleme yapılacak yedeğin başlayacağı tarihi seçmek için kullanılır. Pencere açıldığında "*Most recent possible*" yani olası en yeni yedek seçeneği seçilir. Pencerenin alt kısmında (Select the backup sets to restore) veri tabanı için alınan yedekler listelenecektir. " *To a point time*" kısmından herhangi bir tarih seçilirse seçilen tarihe uygun yedekler alttaki liste içerisinde listelenir.
From Database: Geri yüklemenin veri tabanından yani veri tabanı için alınan yedeklerden yapılmasını sağlar. Bu seçenek seçili olduğunda "Select the backup sets to restore" bölümünde uygun yedekler listelenecektir.



Şekil 14.14. Ogrencibilgi Veri Tabanına Ait Yedeklerinde Sorun Olup Olmamasının

Sorgulanması

From device: Geri yükleme, tanımlı olan bir sürücüden veya bilgisayarın herhangi bir yerindeki dosyadan yapılacaksa bu seçenek seçilerek yanındaki butondan yedek dosyası veya tanımlı sürücü seçilmelidir. Gerekli seçim yapıldığında aşağıdaki listede uygun yedek dosyaları listelenecektir.

"Restore Database" penceresinin *"Options"* bölümünde geri yükleme ile ilgili seçenekler vardır.

Bu pencere üzerinde, geri yüklenen yedeğin mevcut veri tabanının üzerine yazılıp yazılamayacağını belirlemek için *"Overwrite the existing database"* seçeneği, veri tabanı ve transaction log dosyaları için orijinal ve geri yüklenecek isimleri vardır.

SQL Sunucu'nun data dosyasının dosya uzantısı *.mdf, log dosyasının uzantısı ise *.ldf 'dir. Backup dosya uzantısı *.bak'dır.

Alt kısımda bulunan üç seçenek "*WITH RECOVERY*", *"WITH NORECOVERY*" ve *"WITH STANDBY*" geri yükleme seçeneklerini göstermektedir.

WITH RECOVERY: Geri yükleme sonrasında veri tabanı kullanıma hazır bırakılır. Bu seçenek ile geri yükleme yapılırsa differential ve transaction log geri yükleme işlemi gerçekleştirilemez.

WITH NORECOVERY: Geri yükleme sonrasında veri tabanı kullanıma kapatılır. Bu durumda kullanıcılar veri tabanına erişemez. Full Backup işleminin geri yüklemesinden sonra differential veya transaction log geri yüklenecekse bu seçenek seçilmelidir.

WITH STANDBY: NORECOVERY ile yapı olarak aynı özelliğe sahiptir. Yani geri yüklemeden sonra differential veya transaction log geri yükleme yapılacaksa kullanılır ve veri tabanı kilitler. NORECOVERY'den farkı kullanıcılar veri tabanına salt okunur olarak erişebilir ve SELECT ifadelerini kullanabilir.

Geri Yükleme T-SQL

Veri tabanı yedeklerini geri yüklemek için *RESTORE DATABASE,* transaction logları geri yüklemek için *RESTORE LOG T-SQL* komutları kullanılır (Burma, 2009).

Kullanım şekli;

RESTORE DATABASE // LOG veritabani_adi FROM DISK=' yedek_dosyası' WITH seçenekler,

Geri yükleme işlemi yapılırken *WITH* ile *RECOVERY*, *NORECOVERY*, *STANDBY* ve *REPLACE* (Geri yüklenecek veri tabanını mevcut veri tabanının üzerine yazmak için kullanılır.) seçenekleri kullanılabilir.

Bu seçenekler "SQL Sunucu Yönetim Stüdyosu ile geri yükleme" kısmında anlatılmıştı.

Aşağıdaki daha önce alınmış tam veri tabanı yedeği geri yüklenmektedir.

Örnek

```
•RESTORE DATABASE OgrenciBilgi
•FROM DISK='E:\yedekler\ogrencibilgi.bak';
```

Aşağıdaki örnekte ilk olarak tam veri tabanı yedeği REPLACE ve NORECOVERY parametreleri ile geri yüklenmektedir. Daha sonra da mevcut fark yedek geri yüklenmektedir.

İşlem günlüğü yedekleri geri yüklenirken STOPAT parametresi kullanılarak belirli bir tarihe kadar alan işlemlerin geri yüklenmesi sağlanabilir.

Örnek

```
•RESTORE DATABASE OgrenciBilgi
•FROM DISK='E:\yedekler\ogrencibilgi.bak';
•WITH REPLACE, NORECOVERY
```

Tam veri tabanı yedeğini geri yükleyelim.

Fark yedeği geri yükleyelim.

Örnek

```
•RESTORE DATABASE OgrenciBilgi
•FROM DISK='E:\yedekler\ogrencibilgidiff.bak';
•WITH NORECOVERY
```

Son yedek geri yüklenirken de NORECOVERY parametresi kullanıldığı için veri tabanı kullanıma kapalıdır. Veri tabanını tekrar kullanıma açmak için aşağıdaki ifade yazılmalıdır.

RESTORE DATABASE OgrenciBilgi WITH NORECOVERY

İşlem günlüğü yedekleri geri yüklenirken STOPAT parametresi kullanılarak belirli bir tarihe kadar alan işlemlerin geri yüklenmesi sağlanabilir.

Aşağıdaki örnek, işlem günlüklerinin 23 Eylül 2013 02:00:20'ye kadar içerdiği işlemleri geri yükleyecektir.

Örnek

```
•RESTORE LOG OgrenciBilgi
•FROM DISK='E:\yedekler\ogrencibilgi.bak';
•WITH STOPAT = '2013-09-23T02:00:20 '
```

Bireysel Etkinlik

- OgrenciBilgi veri tabanının tam yedeğini bu OgrenciBilgi_Backup medya sete alın.
- Sonra veri tabanınıza yeni tablo ekleyin ve bu tablo üzerinde INSERT, UPDATE işlemleri gerçekleştirin.
- Sonra OgrenciBilgi veri tabanının fark yedeğini alın.
- Geri yükleme işlemi yapmadan önce bir görüntü oluşturun.
- Sonra OgrenciBilgi Veri tabanını geri yükleyin.

Özet

- Yedekleme ve geri yükleme günümüze IT sektörünü ve özellikle veri tabanı yöneticilerini çok yakından ilgilendiren bir konudur. Veri kayıpları işletmelere önemli ölçüde zarar vermekte ve para, zaman ve itibar kayıplarına neden olabilmektedir. Günümüzde veri kaybına uğrayan işletmelerin %60'ı 6 ay içerisinde iflas seviyesine geldiği araştırma sonuçlarına göre gözlemlenmektedir.
- Veri kayıplarının en büyük sebebi donanım kesintisi oluştururken fidye alma amaçlı yapılan siber saldırılar, doğal afetler, elektrik kesintileri ile dosya silme, yanlışlıkla hard diskin ya da sunucunun formatlanması gibi insan hataları da bu kayıpları yaratan diğer başlıca nedenler olarak sıralanmaktadır. İşletmeler karşılaşılabilecek risklere göre önlem almak için yedekleme ve geri yükleme planları yapar. Bu sayede kriz anı meydana geldiğinde verileri kurtarmak için önceden tanımlanmış olan prosedürleri takip etmeniz yeterli olacaktır. İş sürekliliği ve felaket kurtarma gibi projeler içerinde yine veri tabanı yedekleme ve geri yükleme senaryoları yer almaktadır.
- Yedekleme iş gücünün sürekli hareketli olduğu IT sektöründe personelin projelere alışması sürecinde çok önemli yer tutar. Veri kayıpları insan faktörünün yanı sıra donanımsal hatalar, yazılımsal hatalar ve doğal afetlerden de kaynaklanabilir. Çoğu zaman verilerin yeniden oluşturulması imkânsızdır ve iş akışını olumsuz etkilemektedir. Bu bölümde veri tabanı loglama seçenekleri ile birlikte full, differential ve log backup konuları üzerinde durduk. Yedekleme seviyenize göre veri kayıplarınızı en aza indirebilirsiniz. SQL sunucunun aniden kapanması durumunda verilerinizi kurtarmak için ihtiyaç duyacağınız transaction log (işlem günlüğü) dosyasıdır. Bu dosya SQL server otomatik kurtarma tarafından yada yedek indirime işlemi tarafından kullanılmak için tutulur.
- Otomatik sistem kurtarma, her ne kadar sunucular kesintisiz çalışmak için tasarlanmış olsa da meydana gelebilecek donanımsal hatalar sonucu sunucunun aniden kapanması durumu ile karşı karşıya kalınılabilir. Otomatik kurtarma işlemi veri tabanı erişime açılmadan önce son kapanan işlem günlüğü noktasına kadar ki değişiklikleri yaparak veriş tabanını kararlı bir hale getirir. Yedekleme işlemi çoğu proje için kullanıcıların inisiyatifine bırakılmayacak kadar önemlidir. SQL Agent servisinin yapılandırılarak SQL Server'in boşta kaldığı zamanlarda yedek alabilmesi için planları oluşturmayı ve bu planların uygulanabilmesi için SQL Agent servisinin başlatılması gerektiğinin üzerinde durduk. Özellikle büyük işletmelerde yedeklemeyi takip etmek sistemler büyüdükçe zorlaşır. Yedeklerin yönetilmesi kataloglanması ve arşivlenmesi yani yedek yönetimi daha farklı bir boyutta konuyu ele almaktadır.
- Alınan yedeklerin listelenmesi tutarlılıklarının kontrol edilmesi ve veri tabanının mantıksal hatalara karşı korunması yedekleme senaryoları ile sağlanmaktadır. Günümüzde yedekleme konusunda çağa ışık tutan teknolojiler ve disk üreticilerinin ortaya çıkarmış olduğu çözümler yedekleme konusunda hataya yer bırakmayan bir yapı kazandırmaktadır. Kilometrelerce uzağa senkron veri transfer eden yapılar ve bu verilerin tutarlılığını kontrol eden algoritmalar. Artık herkes bilginin ne kadar ne değerli olduğunun farkında bundan dolayıdır veri kayıplarını en aza indirecek mümkünse sıfırlayacak ürünler, sistemler ve yazılımlar geliştirilmektedir. Geri yükleme yapmadan önce tekrar bir yedek almak sizi olası veri kayıplarından kurtaracaktır. Yedekleme planları sizin yerinize günlük, haftalık ve aylık yedekler almakta ve sizleri bilgilendirmektedir.

DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi veri kayıplarının temel nedeni değildir?
   a) Donanımsal sorunlar
   b) Doğal afetler
   c) Yazılımsal sorunlar
   d) Fiziksel kaynaklar
   e) İnsan kaynaklı hatalar

2. Aşağıdakilerden hangisi yedek dosyasında bir sorun olup olmadığını kontrol etmek için kullanılan parametredir?
   a) RESTORE HEADERONLY
   b) RESTORE FILELISTONLY
   c) RESTORE VERIFYONLY
   d) RESTORE CHECKPNLY
   e) RESTORE SEARCHONLY

3. Microsoft SQL Sunucu Yönetim Stüdyosu üzerinden yedek almak için Backup Device nerden tanımlanır?
   a) Task / Backup
   b) Properties / Options
   c) Backup / New Backup Device
   d) System / Backup Devices
   e) Server Objects / Backup

   I. Simple Recovery Model
   II. Bulk-Logged Recovery Model
   III. Full Recovery Model

4. Verilen veri tabanı günlükleme seçenekleri aşağıdakilerin hangisinde kayıt edildiği günlüklere göre büyükten küçüğe doğru sıralanmıştır?
   a) I, II ve III
   b) I, III ve II
   c) III, II ve I
   d) II, III ve I
   e) III, I ve II

5. Microsoft SQL Sunucu'da sırasıyla Data ve Log dosyası uzantısı aşağıdakilerden hangisidir?
   a) mdf - bak
   b) bak -bcp
   c) ldf - mdf
   d) df - lf
   e) mdf – ldf

I. Veri tabanını servisleri yeniden başlatılır.
II. Veri tabanının Full yedeği alınır.
III. Açık olan hareketler kapatılır ve yedeği alınır.

6. Yukarıdakilerden hangisi ya da hangileri veri tabanında İşlem Günlüğü dosyasının büyüklüğünü azaltmak için yapılmalıdır?
   a) Yalnız I
   b) Yalnız II
   c) I ve II
   d) II ve III
   e) I, II ve III

Fark yedeği (differential) şeklinde alınan bir veri tabanına ait yedekler; T (Full yedek), T1, T2, T3 ve T4' tür (T1-T2-T3-T4 Sıra ile Fark yedekleri).

7. Buna göre kullanıcının T3 anına geri dönebilmesi için aşağıdaki yedeklerden hangisi ya da hangilerine ihtiyaç duyar?
   a) T3
   b) T ve T3
   c) T1 ve T4
   d) T, T1 ve T3
   e) T, T1, T2 ve T3

I. Yedekler sadece bir dosyada tutulur ve dağıtılamaz.
II. Mirrored Media Set yedekleri birden fazla kopyaya saklar.
III. Yedekleme aygıtlarını Mirror Set gruplayarak yönetir.

8. Yukarıdakilerden hangisi ya da hangileri Microsoft SQL Sunucu yedekleme ile ilgili doğru kavramdır?
   a) Yalnız I
   b) Yalnız II
   c) Yalnız III
   d) I ve II
   e) I ve III

9. Microsoft SQL Sunucu veri tabanında geri yükleme işlemi sonrasında veri tabanını kilitleyip sadece kullanıcıların salt okunur olarak erişip SELECT sorgusu çekmesine izin veren RESTORE parametresi aşağıdakilerden hangisidir?
   a) WITH RECOVERY
   b) WITH WAIT
   c) WITH NORECOVERY
   d) WITH STOPAT
   e) WITH STANDBY

10. MS-SQL Server'da fark yedeği (differential) hangi harf ile ifade edilir?
    a) D
    b) F
    c) L
    d) I
    e) N

Cevap Anahtarı
1.d, 2.c, 3.e, 4.c, 5.e, 6.e, 7.e, 8.b, 9.e, 10.d

YARARLANILAN KAYNAKLAR

Burma, Z. A. (2009). Veri Tabani Yönetim Sistemleri, 1. Cilt, 2. Baskı, Seçkin Yayıncılık.

Gözüdeli, Y. (2010). Yazılımcılar İçin Sql Server 2008 R2 & Veritabanı Programlama, 1. Cilt, 5. Baskı, Seçkin Yayıncılık.

Server, S. Q. L. (2008). Microsoft SQL Server 2008 R2 Books Online, http://technet. microsoft.com/en- US/sqlserver/ff398089