DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI Veri Tabanı Temel Kavramları
ÜNİTE NO 1
YAZAR Öğr.Gör.Dr. ERKAN BAYRAM

VERİ TABANI TEMEL KAVRAMLARI

Bu ünitede veri tabanı sistemlerinin kuramsal temelleri, temel kavramları ve uygulama alanları bütüncül bir yaklaşımla ele alınmıştır. Dijital dönüşüm sürecinin hız kazandığı günümüzde veri, kurumların stratejik karar süreçlerini yönlendiren en önemli kaynaklardan biri hâline gelmiştir. Ancak verinin değerli olması, yalnızca toplanmasına değil; düzenli biçimde saklanmasına, güvenli şekilde korunmasına, tutarlı olarak güncellenmesine ve anlamlı bilgiye dönüştürülmesine bağlıdır. Bu gereksinimler, veri tabanı sistemlerinin neden vazgeçilmez olduğunu ortaya koymaktadır. Genel olarak bu ünite, veri tabanı sistemlerinin kavramsal altyapısını sağlamayı amaçlamaktadır. Veri modellerinden anahtar kavramlara, dosya sistemlerinden DBMS mimarisine kadar temel bileşenler sistematik biçimde ele alınmıştır. Bu çerçevede öğrencilerin, ilerleyen ünitelerde SQL sorgulama, veri tabanı tasarımı ve performans optimizasyonu konularını daha bilinçli ve analitik bir perspektifle ele alabilecek düzeye ulaşmaları hedeflenmiştir. Veri tabanı temel kavramlarının anlaşılması hem akademik gelişim hem de profesyonel bilişim uygulamaları açısından kritik bir kazanım olarak değerlendirilmektedir.

Veri Tabanı Tanımı

Ünitenin bu bölümünde veri tabanı tanımı açıklanmış ve veri tabanlarının birbiriyle ilişkili verilerin mantıksal bütünlük içinde depolandığı sistematik yapılar olduğu vurgulanmıştır. Veri tabanı, yalnızca bir depolama alanı değil; aynı zamanda veri bütünlüğünü, erişilebilirliğini ve güvenliğini sağlayan bir yönetim altyapısıdır. Bu bağlamda veri yönetiminin kurumsal yapılardaki önemi ve bilgi üretim sürecindeki rolü ortaya konmuştur.

Veri Modelleri

Bu bölümde veri modelleri incelenmiş ve veri modellemenin soyutlama süreci üzerinde durulmuştur. Hiyerarşik ve ağ modelleri tarihsel gelişim açısından değerlendirilmiş; ancak günümüzde en yaygın kullanılan modelin ilişkisel veri modeli olduğu belirtilmiştir. Ayrıca veri soyutlama kavramı açıklanarak, Üç Şema (Three-Schema) Mimarisi çerçevesinde dış, kavramsal ve iç şema düzeyleri tanımlanmıştır. Bu yapı sayesinde veri bağımsızlığı sağlanmakta; fiziksel değişikliklerin uygulamaları etkilemesi önlenmektedir.

Veri Tabanı Veri Türleri

Bu bölümde veri tabanı veri türleri ayrıntılı olarak ele alınmıştır. Sayısal veri türleri, karakter veri türleri, tarih ve zaman veri türleri ile özel veri tipleri incelenmiştir. Doğru veri türü seçiminin hem depolama maliyeti hem de performans açısından kritik öneme sahip olduğu vurgulanmıştır. Özellikle SQL Server 2019 ortamında veri türü seçiminin indeksleme, sorgu optimizasyonu ve bellek kullanımı üzerinde doğrudan etkili olduğu belirtilmiştir.

İlişkisel Veri Tabanları

Bu bölümde ilişkisel veri tabanları konusu işlenmiş ve tabloların iki boyutlu yapısı açıklanmıştır. İlişkisel modelde verilerin satırlar (kayıtlar) ve sütunlar (alanlar) şeklinde düzenlendiği; tablolar arası ilişkilerin anahtarlar aracılığıyla kurulduğu ifade edilmiştir. Bu modelin matematiksel temeli olan küme teorisi ve ilişkiler kavramına değinilmiş; SQL dilinin ilişkisel model üzerindeki rolü açıklanmıştır. İlişkisel veri tabanlarının esnek sorgulama imkânı sunması, veri bütünlüğünü koruması ve ölçeklenebilir olması temel avantajları arasında gösterilmiştir.

Tablo, Kayıt, Alan Kavramları

Bu bölümde tablo, kayıt ve alan kavramları ayrıntılı biçimde ele alınmıştır. Her tablonun belirli alanlardan oluştuğu ve her kaydın bu alanlara ait değerleri içerdiği belirtilmiştir. Bu yapının veri organizasyonunun temelini oluşturduğu vurgulanmıştır.

Birincil Anahtar ve Yabancı Anahtar Mantığı

Bu bölümde ise Birincil Anahtar (Primary Key) ve Yabancı Anahtar (Foreign Key) mantığı açıklanmıştır. Birincil anahtarın her kaydı benzersiz biçimde tanımladığı; yabancı anahtarın ise tablolar arası referans bütünlüğünü sağladığı belirtilmiştir. Ayrıca bire-bir, bire-çok ve çoktan-çoğa ilişkiler örneklerle açıklanmış; çoktan-çoğa ilişkilerde ara tablo kullanımına değinilmiştir.

Dosya Tabanlı Sistemler, Veri Tabanı Sistemleri

Bu bölümde dosya tabanlı sistemler ile veri tabanı sistemleri karşılaştırılmıştır. Dosya tabanlı yaklaşımların veri tekrarına, tutarsızlığa ve güvenlik zafiyetlerine yol açtığı belirtilmiştir. Buna karşılık DBMS tabanlı sistemlerin merkezi veri yönetimi, eşzamanlılık kontrolü, veri bütünlüğü ve

yedekleme mekanizmaları sunduğu açıklanmıştır. Bu karşılaştırma, modern bilgi sistemlerinde DBMS kullanımının neden zorunlu hâle geldiğini ortaya koymuştur.

DBMS Nedir? Neden Kullanılır?

Bu bölümde DBMS (Veri Tabanı Yönetim Sistemi) kavramı detaylandırılmıştır. DBMS'in verilerin tanımlanması, saklanması, işlenmesi ve korunmasını sağlayan yazılım sistemi olduğu belirtilmiştir. DBMS'in sağladığı temel avantajlar arasında veri tekrarının azaltılması, veri tutarlılığının sağlanması, güvenlik mekanizmaları, çok kullanıcılı çalışma desteği, ACID prensipleri ile transaction güvenliği, performans optimizasyonu ve yedekleme/kurtarma özellikleri yer almaktadır. Bu özelliklerin özellikle kurumsal sistemlerde hayati öneme sahip olduğu vurgulanmıştır

Gerçek Hayattan Kullanım Örnekleri

Son bölümde gerçek hayattan kullanım örnekleri incelenmiştir. Eğitim kurumlarında öğrenci otomasyon sistemleri; bankacılıkta para transferi ve hesap yönetimi; e-ticarette sipariş ve stok kontrolü; sağlık sektöründe hasta bilgi sistemleri; kamu ve kurumsal yapılarda personel ve bütçe yönetimi gibi uygulama alanları değerlendirilmiştir. Bu örnekler, veri tabanı sistemlerinin yalnızca teorik değil; pratik ve hayati uygulama alanlarına sahip olduğunu göstermiştir.

DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI Veri Modelleme ve ER Diyagramları
ÜNİTE NO 2
YAZAR Öğr.Gör.Dr. ERKAN BAYRAM

VERİ MODELLEME ve ER DİYAGRAMLARI

Bu ünitede veri modelleme süreci, ER (Varlık–İlişki) modeli ve ilişkisel veri tabanı tasarımının kavramsal temelleri ayrıntılı biçimde ele alınmıştır. Her alt başlık, veri tabanı tasarım sürecinin belirli bir aşamasını temsil etmektedir.

İlişkisel Veri Tabanı Tasarımı

Bu bölümde ilişkisel veri tabanı tasarım sürecinin temel aşamaları açıklanmıştır. Veri tabanı tasarımının kavramsal, mantıksal ve fiziksel olmak üzere üç ana düzeyde gerçekleştirildiği vurgulanmıştır. Kavramsal tasarım aşamasında sistem gereksinimleri analiz edilmekte ve varlıklar ile ilişkiler belirlenmektedir. Mantıksal tasarım aşamasında bu yapı ilişkisel modele dönüştürülerek tablolar, anahtarlar ve kısıtlar tanımlanmaktadır. Fiziksel tasarım aşamasında ise MSSQL Server 2019 ortamında performans, indeksleme ve depolama stratejileri belirlenmektedir. Sağlam bir tasarım sürecinin veri tutarlılığı, performans ve ölçeklenebilirlik açısından kritik öneme sahip olduğu belirtilmiştir.

İlişkisel Veri Tabanının Kavramsal Tasarımı

Bu bölümde kavramsal tasarımın önemi ele alınmıştır. Kavramsal tasarım, teknik ayrıntılardan bağımsız olarak sistemin veri gereksinimlerinin modellenmesini sağlar. ER modeli bu aşamada kullanılan temel araçtır. Kavramsal tasarım sayesinde sistem gereksinimleri görsel ve anlaşılır bir yapı ile ifade edilir. Bu süreç, yazılım geliştirme sürecinde iletişim hatalarını azaltır ve mantıksal tasarım aşamasına sağlam bir temel oluşturur.

Varlık (Entity) ve Özellik (Attribute)

Bu bölümde veri modellemenin temel kavramları olan varlık ve özellik açıklanmıştır. Varlık, hakkında veri saklanması gereken gerçek dünya nesnesini temsil ederken; özellik, bu varlığı tanımlayan nitelikleri ifade eder. Özelliklerin basit, bileşik, tek değerli, çok değerli ve türetilmiş türleri ele alınmıştır. Anahtar özellik kavramı, her varlık örneğinin benzersiz biçimde tanımlanmasını sağlamaktadır. Bu bölümde doğru varlık ve özellik seçiminin normalizasyon ve veri bütünlüğü açısından taşıdığı önem vurgulanmıştır.

İlişki Türleri (1–1, 1–N, N–N)

Varlıklar arasındaki bağlantılar ilişki olarak adlandırılır ve farklı kardinalitelere sahip olabilir. Bire-bir (1:1) ilişkiler, her iki varlığın birer örneğinin birbirine karşılık geldiği durumları ifade eder ve genellikle iki varlığı tek tabloda birleştirmenin daha uygun olduğu anlamına gelir. Bire-çok (1:N) ilişkiler, bir varlığın birçok örneği ile ilişkilendirildiği ama karşı tarafın yalnızca bir örneğe sahip olduğu durumları tanımlar; bu tür ilişkilerde "çok" tarafının tablosuna, "bir" tarafının birincil anahtarı yabancı anahtar olarak eklenir. Çoka-çok (N:N) ilişkilerde her iki tarafta da çoklu eşleşmeler bulunur. Bu durumda doğrudan bağlantı mümkün değildir; araya ara tablo (bağlantı tablosu) eklenerek her iki varlığın anahtarları bu tabloda saklanır. Kardinalite ve katılım kuralları, her ilişkinin kaç örnek içerebileceğini ve bu ilişkinin zorunlu mu yoksa isteğe bağlı mı olduğunu belirler. Doğru kardinalite ve katılım tanımı, veri bütünlüğünü korur ve sorgu performansını optimize eder.

Varlık İlişki Modeli

ER (Entity-Relationship) modeli, veri tabanı tasarımını görsel bir biçimde ifade etmek için kullanılan bir yöntemdir. Bu modelde sistemdeki varlıklar dikdörtgenler, nitelikler oval şekiller ve ilişkiler elmas sembollerle gösterilir. Kardinalite ve katılım kuralları, varlıkları birbirine bağlayan çizgilerin uçlarına yazılan sayılar veya karga ayağı sembolleriyle gösterilir. Model ayrıca zayıf varlıkları çift çizgili dikdörtgenler ve genelleştirme/özelleştirme hiyerarşilerini üçgen sembollerle ifade edebilir. ER modellemesi, kavramsal tasarımı mantıksal şemaya dönüştürme öncesinde kontrol etmek için kullanılır: varlıkların tam olup olmadığı, ilişkilerin doğru tanımlanıp tanımlanmadığı ve niteliklerin yerinde olup olmadığı görsel olarak doğrulanır. ER modelinden ilişkisel şemaya geçişte, her varlık bir tabloya dönüştürülür; birden çoğa ilişkilerde yabancı anahtarlar eklenir; çoka-çok ilişkiler için ise ara tablolar kullanılır.

Varlık Kümesi

Bir varlık kümesi, aynı varlık tipinin tüm örneklerini içeren topluluktur. Veri tabanı tasarımında her varlık kümesi genellikle bir tabloya karşılık gelir. Bu tablo birincil anahtar sayesinde her satırı benzersiz bir varlık örneği olarak tanımlar. Varlık kümesi kavramı, varlık türü (soyut tanım) ile varlık örneği (somut kayıt) arasındaki farkı açıkça ortaya koyar. Tasarım sırasında hangi kavramların

bağımsız olarak var olacağı belirlenir ve bu kavramlar ayrı varlık kümeleri hâlinde modellenir; bağımsız olmayan, bir üst varlığa bağlı olan kavramlar ise zayıf varlık olarak ele alınır. Varlık kümelerinin nitelikleri, tanımlayıcı (anahtar) ve açıklayıcı olmak üzere sınıflandırılır; aday anahtarlar arasından birincil anahtar seçilir. İleri seviye modellerde genelleştirme (üst varlık) ve özelleştirme (alt varlık) ilişkileri kullanılarak varlık kümeleri hiyerarşik olarak düzenlenebilir. Örneğin bir kütüphane sisteminde Kitap, Üye, Yazar ve Kategori ayrı varlık kümeleri hâlinde tutulur; her biri kendine özgü niteliklere sahiptir ve ilişkisel şemada bir tablo olarak karşılık bulur.

Zayıf Varlık Kümesi

Zayıf varlık kümeleri, var olabilmek için bir veya daha fazla güçlü varlık kümesine bağlı olan ve kendi başlarına benzersiz bir anahtara sahip olmayan varlıklardır. Örneğin bir sigorta poliçesine bağlı lehdar kayıtları veya bir kitabın ödünç işlemleri, üst varlık olmadan anlam taşımaz. Zayıf varlıkların anahtarı, güçlü varlığın anahtarı ile bir ayırıcı öznitelik (örneğin sıra numarası veya tarih) birleşiminden oluşan bir birleşik anahtar ile tanımlanır. ER diyagramlarında zayıf varlıklar çift çizgili dikdörtgenle; zayıf varlığı bağlayan tanımlayıcı ilişkiler ise çift çizgili elmasla gösterilir. Bir zayıf varlık için tanımlayıcı ilişkinin katılımı her zaman zorunludur. İlişkisel şemada zayıf varlıklar, birden fazla sütunu kapsayan birincil anahtar ve ilgili güçlü varlığın yabancı anahtarına sahip bir tablo olarak uygulanır; bu yabancı anahtarlara silme ve güncelleme kuralları eklenerek referans bütünlüğü korunur. Zayıf varlık modelinin kullanımı hem anlamsal doğruluğu hem de veri tabanı tasarımının tutarlılığını artırır.

ER Diyagram Çizim Kuralları

ER diyagramı, kavramsal modeli görsel olarak ifade etmenin standart yoludur ve notasyon kurallarına uygun olarak çizilmelidir. Varlıklar dikdörtgenlerle, zayıf varlıklar ise çift çizgili dikdörtgenlerle gösterilir. Nitelikler oval sembollerle ve anahtar niteliğin altı çizili olarak belirtilir. Çok değerli nitelikler çift ovalle, türetilmiş nitelikler kesik çizgili ovalle gösterilir. Varlıklar arasındaki ilişkiler elmas şeklinde sembollerle gösterilir. Bu sembollerin içine ilişki adı yazılır. Kardinalite ve katılım kuralları, ilişki çizgilerinin uçlarına yazılan “1”, “N” veya karga ayağı sembolleriyle ifade edilir ve katılımın zorunlu mu isteğe bağlı mı olduğu çizgi türüyle gösterilir. Hiyerarşik yapılar için üçgen semboller kullanılır; genelleştirme/özelleştirme ilişkileri bu sembol üzerinden alt varlıklara bağlanır. Diyagram çizerken okunabilirliği artırmak için çizgiler mümkün olduğunca kesiştirilmez, benzer varlıklar gruplanır ve kardinalite sembolleri varlığa yakın yerleştirilir. Diyagramın doğruluğunu sağlamak için model paydaşlarla birlikte gözden geçirilir ve hatalar erken aşamada düzeltilir.

Gerçek Hayat Örnekleri

Kavramsal ve ilişkisel tasarım prensiplerinin uygulamada nasıl çalıştığını anlamak için en iyi yol, gerçek bir senaryo üzerinden ilerlemektir. Kütüphane yönetim sistemi örneği, veri tabanı tasarımının tüm adımlarını içerir. Öncelikle gereksinim analizi yapılarak sistemde hangi nesnelerin yönetileceği (Kitap, Kopya, Üye, Yazar, Kategori, Ödünç) belirlenir ve bu varlıklara ait nitelikler tanımlanır. Ardından varlıklar arasındaki ilişkiler incelenir: Kitap–Kopya ve Üye–Ödünç ilişkileri birden çoğa; Kitap–Yazar ve Kitap–Kategori ilişkileri çoka-çok olduğundan KitapYazar ve KitapKategori adlı ara tablolar tasarlanır. Bu adımlar sonucunda ER diyagramı çizilir ve kardinalite sembolleri eklenerek görsel bir model elde edilir. Diyagramdan ilişkisel şemaya geçilirken her varlık için bir tablo oluşturulur, yabancı anahtarlar eklenir ve çoka-çok ilişkiler bağlantı tablolarıyla temsil edilir. Normalizasyon kuralları uygulanarak verinin tekrar etmemesi sağlanır ve zayıf varlıkların (ör. Ödünç) anahtarları güçlü varlıkların anahtarlarıyla birleşim hâlinde tanımlanır. Model gerçek verilerle test edilir; gecikme cezaları gibi türetilmiş alanlar doğrulanır ve eksik nitelikler belirlenir. Geliştirme sürecinde e-kitap, dergi veya abonelikler gibi yeni gereksinimler ortaya çıkarsa, varlık kümesi ve ilişkiler hiyerarşik yapılarla genişletilerek tasarım güncellenir. Bu örnek, öğrencilere veri tabanı tasarımının nasıl adım adım ilerlediğini ve tasarım kararlarının nasıl verildiğini pratik olarak gösterir.

DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI Normalizasyon
ÜNİTE NO 3
YAZAR Öğr.Gör.Dr. ERKAN BAYRAM

NORMALİZASYON

Bu ünitede veri tabanı tasarımının temel ilkelerinden biri olan normalizasyon kavramı ele alınmaktadır. Normalizasyon, verileri belirli kurallar çerçevesinde düzenleyerek tekrarları azaltmayı, veri bütünlüğünü sağlamayı ve tutarsızlıkları önlemeyi amaçlayan sistematik bir süreçtir. İlişkisel veri tabanlarında hatalı tasarımlar çoğunlukla veri tekrarından, yanlış bağımlılıklardan ve uygun olmayan tablo yapılarından kaynaklanır. Bu nedenle normalizasyon yalnızca teknik bir kavram değil, aynı zamanda doğru ve sürdürülebilir sistem tasarımının temelidir. Ünitede önce veri tekrarının zararları incelenmekte, ardından normalizasyonun amacı açıklanmakta ve temel ile ileri düzey normal formlar örneklerle ele alınmaktadır.

Veri Tekrarının Zararları

Veri tekrarı, aynı bilginin birden fazla satırda veya tabloda saklanması durumudur. İlk bakışta basit gibi görünen bu durum, zamanla ciddi tasarım sorunlarına yol açar. En temel zararlarından biri depolama alanının gereksiz yere artmasıdır. Özellikle büyük ölçekli sistemlerde aynı bilginin binlerce kez saklanması performans kaybına neden olabilir. Ancak veri tekrarının asıl tehlikesi tutarsızlık üretmesidir. Bir bilgi birden fazla yerde tutuluyorsa güncelleme işlemi sırasında tüm kopyaların eş zamanlı değiştirilmesi gerekir. Bu işlem eksik yapılırsa sistemde çelişkili bilgiler oluşur. Örneğin bir müşterinin adresi üç farklı kayıtta yer alıyorsa ve adres değişikliği yalnızca iki kayıtta yapılırsa veri tabanı artık güvenilir değildir. Benzer şekilde bir kaydın silinmesi, başka bir önemli bilginin de kaybolmasına yol açabilir. Bu tür durumlar ekleme, silme ve güncelleme anomalileri olarak adlandırılır. Veri tekrarının bu olumsuz etkileri, normalizasyon ihtiyacını ortaya çıkarır.

Normalizasyonun Amacı

Normalizasyonun temel amacı veri tekrarını azaltmak ve veri bütünlüğünü sağlamaktır. Bu süreçte veriler mantıksal olarak anlamlı tablolara ayrılır ve tablolar arasındaki ilişkiler birincil ve yabancı anahtarlar aracılığıyla tanımlanır. Böylece her bilgi yalnızca bir yerde saklanır. Bu yaklaşım hem tutarlılığı artırır hem de bakım işlemlerini kolaylaştırır. Normalizasyon ayrıca sistemin esnekliğini güçlendirir. Yeni bir özellik eklendiğinde veya mevcut yapıda değişiklik yapıldığında tüm sistemi yeniden düzenlemek yerine ilgili tablo üzerinde değişiklik yapmak yeterli olur. Bu durum yazılım geliştirme sürecinde önemli bir avantaj sağlar. Sonuç olarak normalizasyon, veri tabanını daha düzenli, daha anlaşılır ve daha yönetilebilir hale getirir.

1NF, 2NF, 3NF

Birinci, ikinci ve üçüncü normal formlar normalizasyon sürecinin temelini oluşturur. Birinci normal form (1NF), her sütunun atomik yani tek bir değer içermesini şart koşar. Aynı hücrede birden fazla bilgi tutulması veya tekrar eden grup yapıları 1NF'e aykırıdır. Bu kural tablo yapısını sadeleştirir ve veri tekrarını azaltmanın ilk adımını oluşturur. İkinci normal form (2NF), tablonun 1NF koşullarını sağlamasını ve anahtar olmayan tüm alanların birincil anahtarın tamamına bağlı olmasını gerektirir. Özellikle birleşik anahtarlarda görülen kısmi bağımlılıklar bu aşamada giderilir. Böylece tablo içindeki gereksiz tekrarlar azaltılır. Üçüncü normal form (3NF) ise geçişli (transitif) bağımlılıkları ortadan kaldırır. Anahtar olmayan bir alan başka bir anahtar olmayan alana bağlı olmamalıdır. Bu kural, tablo yapısını daha mantıklı hale getirir ve gereksiz veri bağımlılıklarını ortadan kaldırır. 1NF, 2 NF ve 3NF birlikte uygulandığında çoğu veri tabanı tasarım problemi çözülmüş olur.

Normalizasyon Örnekleri

Normalizasyon yalnızca kuramsal bir kavram değildir. Gerçek tablolar üzerinde uygulandığında anlam kazanır. Tek bir tabloda hem müşteri hem sipariş hem de ürün bilgilerinin tutulduğu bir yapı düşünelim. Bu tür bir tasarımda müşteri bilgileri her siparişte tekrar edilir. Normalizasyon sürecinde önce tekrar eden alanlar ayrıştırılır, ardından müşteri, sipariş ve ürün tabloları ayrı ayrı oluşturulur. Bu örnekler, tabloların adım adım 1NF'ten 3NF'e nasıl dönüştürüldüğünü gösterir. Öğrenciler bu süreçte birincil anahtarların nasıl belirlendiğini, yabancı anahtarların nasıl tanımlandığını ve tablolar arası ilişkilerin nasıl kurulduğunu öğrenirler. Bu uygulamalı yaklaşım, normalizasyon kurallarının somut şekilde anlaşılmasını sağlar.

Hatalı Tasarımların Düzeltilmesi

Hatalı veri tabanı tasarımları genellikle fazla sütun kullanımı, tekrar eden veri grupları ve yanlış anahtar seçiminden kaynaklanır. Örneğin bir tabloda "Telefon1", "Telefon2", "Telefon3" gibi alanların bulunması esnek olmayan bir tasarıma işaret eder. Benzer şekilde birden fazla varlık türünün tek

tabloda tutulması da ciddi tutarsızlıklara yol açar. Bu bölümde hataların nasıl analiz edileceği ve normalizasyon kuralları kullanılarak nasıl düzeltileceği açıklanır. Tasarım hatalarını erken aşamada fark etmek, ileride oluşabilecek büyük sistem sorunlarını önler. Doğru tablo yapısı, veri bütünlüğünün korunmasında temel rol oynar.

4. Normal Form (4NF) – Çok Değerli Bağımlılıklar

Dördüncü normal form, çok değerli bağımlılıkları ortadan kaldırmayı hedefler. Bir varlığın birbirinden bağımsız birden fazla özelliğe sahip olduğu durumlarda bu özelliklerin aynı tabloda tutulması gereksiz kombinasyonlara neden olabilir. Örneğin bir öğretmenin hem verdiği dersler hem de bildiği yabancı diller aynı tabloda tutulduğunda gereksiz satır tekrarları oluşur. 4NF, bu tür bağımlılıkları ayrı tablolara ayırarak veri tekrarını azaltır. Böylece tablo daha sade ve anlaşılır hale gelir. 4NF genellikle daha karmaşık veri tabanı tasarımlarında ortaya çıkan ince yapısal sorunları çözmek için kullanılır.

5. Normal Form (5NF) – Birleşim Bağımlılığı

Beşinci normal form, birleşim bağımlılıklarını ele alır. Bir tablo alt tablolara ayrıldığında tekrar birleştirildiğinde veri kaybı olmamalıdır. Eğer ayrıştırma işlemi gereksiz kombinasyonlar oluşturuyorsa tablo 5NF'e uygun değildir. 5NF, özellikle çoklu ilişkilere sahip karmaşık sistemlerde önem kazanır. Bu normal form, gereksiz birleşimlerden kaynaklanan veri tekrarını önlemeyi amaçlar. 5 NF'e ulaşmak her zaman gerekli değildir ancak karmaşık yapılar için güçlü bir tasarım prensibi sunar.

DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI Veri Tabanı Araçlarının Kurulumunu Yapmak
ÜNİTE NO 4
YAZAR Öğr.Gör.Dr. ERKAN BAYRAM

VERİ TABANI ARAÇLARININ KURULUMUNU YAPMAK

Bu ünite, SQL Server 2019'un yapısını, bileşenlerini, özelliklerini, kullanım alanlarını ve kurulum sürecini bütüncül bir yaklaşımla ele almıştır. Öğrenci, ünite sonunda SQL Server'ın ne olduğunu, nasıl çalıştığını, hangi alanlarda kullanıldığını ve nasıl kurulacağını anlayabilecek bilgi ve beceri düzeyine ulaşmaktadır. Bu kazanımlar, veri tabanı teknolojileri alanında sağlam bir temel oluşturmaktadır.

SQL Server 2019 Tanıtımı

Bu bölümde SQL Server 2019'un genel çerçevesi ele alınmıştır. SQL Server 2019, Microsoft tarafından geliştirilen ve hem küçük ölçekli uygulamalarda hem de kurumsal sistemlerde yaygın olarak kullanılan güçlü bir ilişkisel veri tabanı yönetim sistemidir. Bu sürüm, gelişmiş güvenlik özellikleri, performans iyileştirmeleri ve bulut entegrasyon kabiliyetleri ile dikkat çekmektedir. SQL Server 2019'un veri yönetim süreçlerindeki rolü açıklanmış ve veri tabanı teknolojileri içindeki konumu değerlendirilmiştir. Bu başlık, öğrencilerin çalışacakları sistemin kapsamını ve amacını kavramalarına yardımcı olmuştur.

SQL Server Nedir?

Bu başlık altında SQL Server'ın temel tanımı yapılmıştır. SQL Server, verileri tablolar halinde saklayan ve SQL dili aracılığıyla bu veriler üzerinde işlem yapılmasını sağlayan bir veri tabanı yönetim sistemidir. Sistem, istemci–sunucu mimarisi ile çalışır. Kullanıcılar SSMS gibi istemci araçlar üzerinden sunucuya bağlanır ve Database Engine üzerinde sorgularını çalıştırır. Veri saklama, sorgu işleme ve sonuç üretme süreçleri kavramsal olarak açıklanmıştır. Ayrıca ilişkisel veri modeli ve tablo yapısının temel özellikleri vurgulanmıştır. Bu bölüm, SQL Server'ın mantıksal yapısının anlaşılması açısından temel oluşturmuştur.

SQL Server 2019'un Temel Özellikleri

Bu bölümde SQL Server 2019'un öne çıkan teknik özellikleri incelenmiştir. Güvenlik, performans, ölçeklenebilirlik, yüksek erişilebilirlik ve yedekleme mekanizmaları detaylandırılmıştır. Rol tabanlı erişim kontrolü, veri şifreleme ve kimlik doğrulama seçenekleri sayesinde güvenli bir veri yönetimi sağlandığı belirtilmiştir. Ayrıca sorgu optimizasyonu ve indeksleme gibi performans artırıcı mekanizmalar açıklanmıştır. Yüksek erişilebilirlik çözümleri ve felaket kurtarma senaryoları ele alınarak sistemin kurumsal ortamlardaki önemi vurgulanmıştır. Bu özellikler sayesinde SQL Server 2019'un geniş ölçekli bilgi sistemlerinde güvenilir bir altyapı sunduğu ifade edilmiştir.

SQL Server Bileşenleri

Bu başlık altında SQL Server'ın modüler yapısı ele alınmıştır. Database Engine, servisler ve yardımcı araçlar sistemin temel bileşenleri olarak açıklanmıştır. Her bileşenin farklı görevler üstlendiği ve sistemin bütünleşik biçimde çalıştığı belirtilmiştir. Özellikle Database Engine'in veri depolama ve sorgu işleme süreçlerini yönettiği vurgulanmıştır. Ayrıca servislerin arka planda sistemin sürekliliğini sağladığı ifade edilmiştir. Bu bölüm, SQL Server'ın tek bir yazılım değil, birbirini tamamlayan bileşenlerden oluşan bir yapı olduğunu göstermiştir.

Database Engine

Database Engine, SQL Server'ın çekirdeğini oluşturan en önemli bileşendir. Bu bölümde sorgu işleme süreci, transaction yönetimi ve veri bütünlüğü kavramları açıklanmıştır. Bir SQL sorgusunun analiz edilmesi, optimize edilmesi ve çalıştırılması aşamaları ele alınmıştır. Eşzamanlı kullanıcı işlemleri sırasında veri tutarlılığının nasıl korunduğu anlatılmıştır. ACID özellikleri kapsamında işlemlerin güvenli ve tutarlı biçimde gerçekleştirildiği vurgulanmıştır. Database Engine'in performans ve güvenlik açısından sistemin merkezinde yer aldığı belirtilmiştir.

SQL Server Management Studio (SSMS) nedir?

Bu bölümde SQL Server Management Studio'nun bir yönetim aracı olduğu açıklanmıştır. SSMS, kullanıcıların SQL Server ile etkileşim kurmasını sağlayan grafik arayüzlü bir istemci uygulamasıdır. Veri tabanı oluşturma, tablo yönetimi, sorgu yazma ve kullanıcı yetkilendirme gibi işlemler SSMS üzerinden yapılmaktadır. SSMS'in kullanıcıya sağladığı kolaylıklar ve öğrenme sürecine katkısı vurgulanmıştır. Bu araç sayesinde teknik işlemlerin daha anlaşılır ve yönetilebilir hale geldiği belirtilmiştir.

SQL Server Management Studio (SSMS) Arayüzü

Bu başlık altında SSMS arayüzünün temel bölümleri incelenmiştir. Object Explorer, Query Editor, Results ve Messages alanları ayrıntılı biçimde açıklanmıştır. Object Explorer'ın veri tabanı

nesnelerini hiyerarşik olarak gösterdiği, Query Editor'ın SQL komutlarının yazıldığı alan olduğu belirtilmiştir. Results alanında sorgu çıktılarının, Messages alanında ise hata ve bilgilendirme mesajlarının görüntülendiği ifade edilmiştir. Arayüzün doğru kullanımı sayesinde veri tabanı yönetim süreçlerinin daha verimli yürütülebileceği vurgulanmıştır.

SQL Server Kullanım Alanları

Bu bölümde SQL Server'ın farklı sektörlerdeki kullanım alanları ele alınmıştır. Eğitim, sağlık, finans, kamu ve e-ticaret gibi alanlarda veri yönetiminin önemi açıklanmıştır. Öğrenci bilgi sistemleri, hastane kayıt sistemleri, banka işlem altyapıları ve belediye otomasyonları gibi örnekler verilmiştir. Veri güvenliği ve işlem tutarlılığının bu sektörlerde neden kritik olduğu vurgulanmıştır. SQL Server'ın güvenlik, performans ve ölçeklenebilirlik özellikleri sayesinde bu alanlarda tercih edildiği ifade edilmiştir.

SQL Server 2019 Kurulumu

Son bölümde SQL Server 2019'un kurulum süreci adım adım özetlenmiştir. Kurulum öncesi gereksinimler, özellik seçimi, instance yapılandırması ve kimlik doğrulama seçenekleri açıklanmıştır. Kurulum sürecinin mantıksal yapısı ele alınarak doğru yapılandırmanın sistem performansı ve güvenliği açısından önemi vurgulanmıştır. Bu bölüm, öğrencilerin sistemi kendi bilgisayarlarında kurarak uygulama yapabilmelerini hedeflemiştir.

DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI Tabloları Oluşturmak Ve Özelliklerini Belirlemek
ÜNİTE NO 5
YAZAR Doç.Dr. MURAT ÇOBAN

Giriş

Bilgi teknolojilerinin hızla gelişmesiyle birlikte verinin üretilme, saklanma ve işlenme biçimleri köklü bir dönüşüm geçirmiştir. Bu dönüşümün merkezinde yer alan veritabanı yönetim sistemleri, özellikle kurumsal ve büyük ölçekli uygulamalarda kritik bir rol üstlenmektedir. Microsoft SQL Server, sunduğu güçlü altyapı, güvenlik mekanizmaları ve gelişmiş sorgulama yetenekleriyle günümüzde en yaygın kullanılan ilişkisel veritabanı yönetim sistemlerinden biridir. SQL Server ortamında verinin etkin bir şekilde yönetilebilmesi ise büyük ölçüde doğru tasarlanmış tablo yapılarının varlığına bağlıdır. Bu bağlamda tablo oluşturma, sütun özelliklerinin belirlenmesi, veri türü seçimi, kısıtlamaların tanımlanması ve tablolar arası ilişkilerin kurulması gibi süreçler, veritabanı tasarımının temel yapı taşlarını oluşturmaktadır. Bu bölümde, T-SQL kullanılarak tabloların oluşturulması ve özelliklerinin belirlenmesine ilişkin kavramlar sistematik ve uygulama odaklı bir yaklaşımla ele alınmıştır.

T-SQL ve SQL Server Ortamına Genel Bakış

Transact-SQL (T-SQL), Microsoft tarafından SQL Server için geliştirilmiş, standart SQL dilinin gelişmiş bir sürümüdür. T-SQL; değişkenler, koşullu ifadeler, döngüler, hata yönetimi ve sistem fonksiyonları gibi ek yapılar sunarak karmaşık veri işlemlerinin gerçekleştirilebilmesine olanak tanır. SQL Server mimarisi; veritabanı, şema ve tablo kavramları etrafında yapılandırılmıştır. Veritabanı, tabloların ve diğer veritabanı nesnelerinin saklandığı en üst düzey kapsayıcı yapı iken, şemalar bu nesnelerin mantıksal olarak gruplandırılmasını sağlar. Bu yapı, özellikle büyük projelerde erişim kontrolü, yetkilendirme ve nesne yönetimi açısından önemli avantajlar sunar. Tablolar ise satır ve sütunlardan oluşan, verinin fiziksel olarak saklandığı temel veri yapılarıdır ve tüm ilişkisel veritabanı işlemlerinin merkezinde yer alır.

CREATE TABLE Komutu ve Tablo Oluşturma Süreci

SQL Server'da tablo oluşturma işlemi CREATE TABLE komutu kullanılarak gerçekleştirilir. Bu komut ile tablo adı, sütun isimleri, her sütuna ait veri türleri ve isteğe bağlı olarak çeşitli sütun özellikleri tanımlanır. Tablo oluşturma aşamasında yapılan tercihler, yalnızca verinin nasıl saklanacağını değil, aynı zamanda veriye erişim hızını, veri bütünlüğünü ve bakım süreçlerini de doğrudan etkiler. Bu nedenle tablo tasarımı, sistem gereksinimleri doğrultusunda dikkatle planlanmalıdır. Şema belirtilerek tablo oluşturulması, özellikle çok kullanıcılı ve modüler sistemlerde tablo adlarının çakışmasını önler ve yönetilebilirliği artırır. CREATE TABLE komutu, veritabanı tasarımının ilk ve en kritik adımlarından biri olarak değerlendirilmektedir.

Veri Türleri ve Doğru Seçimin Önemi

Veri türleri, bir sütunda saklanacak verinin biçimini, boyutunu ve davranışını belirleyen temel unsurlardır. Sayısal veri tipleri (INT, BIGINT, SMALLINT, DECIMAL) genellikle sayma, hesaplama ve karşılaştırma işlemlerinde kullanılırken, karakter veri tipleri (CHAR, VARCHAR, NVARCHAR) metinsel bilgilerin saklanmasında tercih edilmektedir. Özellikle NVARCHAR veri tipi, Unicode desteği sayesinde farklı dillerdeki karakterlerin güvenli biçimde saklanmasına olanak tanır. Tarih ve zaman veri tipleri (DATE, DATETIME, TIME) ise kayıt takibi, zaman bazlı analizler ve işlem geçmişlerinin tutulması açısından büyük önem taşır. Yanlış veri türü seçimi, gereksiz depolama alanı kullanımına, performans düşüşüne ve veri tutarsızlıklarına yol açabilirken; doğru veri türü seçimi sistemin genel verimliliğini önemli ölçüde artırır.

Sütun (Kolon) Özelliklerinin Tanımlanması

Sütun özellikleri, tablonun veri kabul etme biçimini ve davranışını belirleyen önemli parametrelerdir. NULL ve NOT NULL ifadeleri, bir sütunun boş değer kabul edip etmeyeceğini belirler. DEFAULT özelliği, ilgili sütuna veri girilmediğinde otomatik olarak atanacak değeri tanımlar ve veri giriş sürecini kolaylaştırır. IDENTITY özelliği, otomatik artan sayısal alanlar oluşturarak özellikle birincil anahtarların yönetimini pratik hale getirir. Hesaplanan sütunlar (computed columns), diğer sütunların değerlerine bağlı olarak otomatik hesaplama yapar ve veri tekrarını önleyerek daha tutarlı bir tablo yapısı sunar. Bu özellikler, hem uygulama katmanındaki yükü azaltır hem de veritabanı düzeyinde kontrol sağlar.

Kısıtlamalar (Constraints) ve Veri Bütünlüğü

Kısıtlamalar, veritabanında veri bütünlüğünü sağlamak amacıyla tanımlanan kurallardır. PRIMARY KEY kısıtlaması, her kaydın benzersiz biçimde tanımlanmasını sağlayarak tablonun temel kimliğini

oluşturur. FOREIGN KEY kısıtlaması, tablolar arasında ilişki kurarak referans bütünlüğünü garanti altına alır. UNIQUE kısıtlaması, belirli alanlarda tekrar eden verilerin önüne geçerken; CHECK kısıtlaması, yalnızca belirlenen koşullara uygun verilerin tabloya eklenmesine izin verir. DEFAULT constraint ise sütunlara varsayılan değer atanmasını sağlar. Bu kısıtlamalar, tablo veya sütun seviyesinde tanımlanabilir ve veritabanının tutarlı ve güvenilir çalışmasına katkı sağlar.

Anahtarlar ve Tablolar Arası İlişkiler

İlişkisel veritabanı sistemlerinin temelini anahtarlar ve tablolar arası ilişkiler oluşturur. Birincil anahtarlar, tabloların benzersizliğini sağlarken yabancı anahtarlar aracılığıyla tablolar arasında mantıksal bağlar kurulur. Referans bütünlüğü, ilişkili tablolarda geçersiz veya tutarsız veri oluşumunu engeller. CASCADE seçenekleri sayesinde ana tabloda yapılan silme veya güncelleme işlemleri ilişkili tablolara otomatik olarak yansıtılabilir. Bu özellikler doğru kullanıldığında veri tutarlılığını artırırken, kontrolsüz kullanıldığında istenmeyen veri kayıplarına yol açabileceğinden dikkatle planlanmalıdır.

ALTER TABLE ile Tablo Yapısının Güncellenmesi

Zaman içerisinde değişen ihtiyaçlar doğrultusunda mevcut tablolar üzerinde yapısal değişiklikler yapılması kaçınılmazdır. ALTER TABLE komutu; yeni sütun ekleme, mevcut sütunların veri tiplerini değiştirme, sütun silme ve kısıtlamaları güncelleme gibi işlemleri mümkün kılar. Özellikle aktif kullanılan sistemlerde ALTER TABLE işlemleri gerçekleştirilirken veri kaybı riski ve performans etkileri göz önünde bulundurulmalıdır. Bu nedenle bu tür işlemlerin mümkünse test ortamlarında denenmesi önerilmektedir.

DROP, TRUNCATE ve DELETE Komutlarının Kullanımı

Veri silme işlemleri farklı ihtiyaçlara göre farklı komutlarla gerçekleştirilir. DELETE komutu, koşula bağlı olarak satır silme imkânı sunar ve geri alınabilir olmasıyla esneklik sağlar. TRUNCATE TABLE komutu, tablodaki tüm verileri hızlı bir şekilde silerken kimlik değerlerini sıfırlar ve geri alınamaz. DROP TABLE komutu ise tabloyu tamamen ortadan kaldırarak hem veriyi hem de tablo yapısını siler. Bu komutların doğru senaryolarda kullanılması, veri güvenliği ve sistem bütünlüğü açısından büyük önem taşır.

Geçici Tablolarla Çalışma

Geçici tablolar, özellikle ara işlem sonuçlarının saklanması ve karmaşık sorguların sadeleştirilmesi amacıyla kullanılmaktadır. Yerel geçici tablolar yalnızca oluşturuldukları oturum boyunca geçerli iken, global geçici tablolar birden fazla oturum tarafından erişilebilir. SELECT INTO ifadesiyle hızlıca oluşturulabilen geçici tablolar, performans gerektiren raporlama ve analiz işlemlerinde önemli avantajlar sunar. Oturum sona erdiğinde otomatik olarak silinmeleri, sistem temizliği açısından da fayda sağlar.

Tablo Tasarımında En İyi Uygulamalar ve Sonuç

Tablo tasarımında normalizasyon ilkelerine uyulması, veri tekrarını azaltarak tutarlı ve bakımı kolay bir yapı oluşturur. Performans odaklı tasarım, yalnızca indeksleme ile değil; doğru veri türü seçimi, gereksiz sütunlardan kaçınılması ve sade tablo yapılarıyla mümkündür. Sonuç olarak, bu bölümde ele alınan tüm konular SQL Server üzerinde güvenilir, ölçeklenebilir ve yüksek performanslı veritabanı sistemleri oluşturmanın temelini oluşturmaktadır.

SQL Sorgu Mantığı ve Analitik Veri Yönetimi

Günümüz bilgi sistemlerinde verinin hacmi ve çeşitliliği sürekli artmakta, bu durum verinin yalnızca depolanmasını değil, aynı zamanda sistematik biçimde sorgulanmasını ve analiz edilmesini zorunlu kılmaktadır. İlişkisel veritabanı yönetim sistemleri, bu ihtiyaca yanıt verebilmek amacıyla yapılandırılmış veri modelleri ve güçlü sorgulama mekanizmaları sunmaktadır. Bu mekanizmaların merkezinde SQL (Structured Query Language) yer almaktadır. SQL, veritabanlarında saklanan verilerle kontrollü ve anlamlı biçimde etkileşim kurulmasını sağlayan standart bir sorgulama dili olarak kabul edilmektedir.

SQL sorguları; kullanıcıların veya uygulamaların veriye erişmesini, verileri filtrelemesini, sıralamasını ve analiz etmesini mümkün kılar. Bu bağlamda sorgu oluşturma süreci yalnızca teknik bir yazım işlemi değil; veriyle düşünme, veri üzerinden yorum yapma ve analitik problem çözme sürecinin temel bir bileşeni olarak değerlendirilmelidir. Doğru yapılandırılmış bir sorgu, büyük veri kümeleri içerisinden güvenilir ve anlamlı sonuçlar üretirken; hatalı veya optimize edilmemiş sorgular performans sorunlarına ve yanlış analizlere yol açabilir.

İlişkisel Yapı ve Sorgu Yazım İlkeleri

İlişkisel veritabanlarının temel özelliklerinden biri, verilerin belirli kurallar çerçevesinde yapılandırılmış tablolar içerisinde saklanmasıdır. Bu yapı, veri bütünlüğünü korurken aynı zamanda sorgulama sürecinde mantıksal bir düşünme yaklaşımı gerektirir. Bu nedenle sorgu yazımı sürecinde SELECT, WHERE ve ORDER BY, ALTER gibi temel ifadelerin işleyiş mantığının doğru kavranması önem taşır.

SQL sorgularının etkin kullanımı yalnızca doğru bilgi üretimini değil, aynı zamanda sistem kaynaklarının verimli kullanılmasını da hedefler. Gereksiz sütun seçimi, hatalı filtreleme koşulları veya bilinçsiz sıralama işlemleri sistem performansını olumsuz etkileyebilir. Bu nedenle sorgu yazımında performans, okunabilirlik ve sürdürülebilirlik birlikte değerlendirilmelidir. Özellikle büyük veri kümelerinde sorgu optimizasyonu, sistem başarısının temel belirleyicilerinden biridir. Bu bölüm, SQL sorgularının temel yapısını ele alarak sütun seçimi, satır filtreleme, sıralama işlemleri, sonuç kümesinin sınırlandırılması ve performans ilkeleri gibi konuları bütüncül bir çerçevede sunmaktadır. Amaç, yalnızca SQL sözdiziminin öğretilmesi değil; sorgu mantığının anlaşılması ve farklı veri senaryolarında doğru yapıların oluşturulabilmesidir.

SQL Sorgularının Temel Rolü

İlişkisel veritabanı yönetim sistemlerinde sorgular, verinin depolanmasının ötesinde veriye erişim ve veriyi anlamlandırma süreçlerinin merkezinde yer alır. SQL, bu süreçleri standart ve sistematik biçimde gerçekleştirmeyi sağlayan bir dildir. Günümüzde pek çok bilgi sistemi, veriye erişim ve analiz işlemlerini SQL sorguları aracılığıyla yürütmektedir. SQL sorguları, kullanıcı ile veritabanı arasındaki etkileşimi düzenler ve büyük veri kümeleri içerisinden anlamlı sonuçların elde edilmesini sağlar. Doğru yazılmış sorgular hem doğru bilgi üretir hem de sistem kaynaklarının etkin kullanılmasına katkı sunar.

Sorgu Kavramı ve Temel Sözdizimi

Sorgu (query), veritabanında saklanan veriler üzerinde belirli işlemler gerçekleştirmek amacıyla yazılan komutlar bütünüdür. SQL sorgularının temelini SELECT ifadesi oluşturur. SELECT, hangi alanların görüntüleneceğini belirlerken; FROM ifadesi verinin hangi tablodan alınacağını tanımlar. WHERE ve ORDER BY gibi ifadeler, temel sorgu yapısını genişleterek daha kontrollü ve hedefli sonuç kümeleri elde edilmesini sağlar. SQL'in modüler yapısı, farklı veri ihtiyaçlarına uygun sorguların sistematik biçimde yazılmasına imkân tanır.

Temel SELECT Sorguları ve Alan Seçimi

Temel SELECT sorgularında tüm alanların seçilmesi ile yalnızca gerekli alanların seçilmesi arasında önemli farklar bulunmaktadır. SELECT * ifadesi, tüm sütunları sonuç kümesine dahil eder ve küçük veri kümelerinde pratik olabilir. Ancak büyük ölçekli veritabanlarında gereksiz veri okuma, bellek kullanımı ve ağ trafiği gibi performans sorunlarına yol açabilir.

Etkili sorgu yazımında yalnızca ihtiyaç duyulan sütunların seçilmesi temel bir ilkedir. Bu yaklaşım, hem sorgu sonuçlarının daha anlaşılır olmasını sağlar hem de performans açısından avantaj sunar.

Filtreleme İşlemleri ve Koşul Yapıları

Filtreleme işlemleri, SQL sorgularının en önemli bileşenlerinden biridir. WHERE ifadesi, sonuç

kümesinde yer alacak kayıtların belirli koşullara göre sınırlandırılmasını sağlar. Karşılaştırma operatörleri kullanılarak sayısal, tarihsel ve metinsel alanlarda koşullar tanımlanabilir.
Mantıksal operatörler (AND, OR, NOT), birden fazla koşulun birlikte değerlendirilmesine imkân tanır. BETWEEN, IN ve LIKE ifadeleri ise aralık, küme ve desen temelli filtreleme işlemlerinde esneklik sağlar.

NULL Değerlerle Çalışma

NULL değerler, bilinmeyen veya girilmemiş verileri temsil eder. Bu değerler klasik karşılaştırma operatörleriyle kontrol edilemez; bu nedenle IS NULL ve IS NOT NULL ifadeleri kullanılır.
NULL değerlerin doğru şekilde ele alınmaması, sorgu sonuçlarının doğruluğunu olumsuz etkileyebilir. Özellikle analiz süreçlerinde eksik verilerin bilinçli biçimde değerlendirilmesi gerekmektedir.

Sıralama ve Sonuç Kümesi Kısıtlama

SQL sorgularında sonuçların düzenlenmesi ORDER BY ifadesi ile gerçekleştirilir. ORDER BY, belirli bir alana göre artan (ASC) veya azalan (DESC) sıralama yapılmasını sağlar. Bu yapı, özellikle raporlama ve analiz süreçlerinde sonuçların yorumlanabilirliğini artırır.
TOP, LIMIT veya FETCH gibi ifadeler ise sonuç kümesinin belirli sayıda kayıtla sınırlandırılmasını sağlar. Bu yöntem, büyük veri kümelerinde gereksiz veri aktarımını önleyerek performans avantajı sunar.

ALTER Komutu

ALTER komutu, SQL'de mevcut veritabanı nesnelerinin yapısını değiştirmek amacıyla kullanılan bir Veri Tanımlama Dili (DDL) komutudur. En yaygın kullanım biçimi ALTER TABLE ifadesidir ve bir tablonun şemasında değişiklik yapılmasını sağlar. Bu komut aracılığıyla tabloya yeni sütun eklenebilir (ADD), mevcut bir sütunun veri tipi, uzunluğu veya NULL/NOT NULL özelliği değiştirilebilir (ALTER COLUMN), sütun silinebilir (DROP COLUMN) ya da tabloya varsayılan değer (DEFAULT) ve çeşitli kısıtlar (CONSTRAINT) eklenebilir. ALTER komutu, tabloyu yeniden oluşturmaz; mevcut yapıyı güncelleyerek veri modelini esnek biçimde düzenleme imkânı sunar. Ancak yapılan değişiklikler doğrudan tablo yapısını etkilediğinden, özellikle büyük veri kümelerinde performans ve veri bütünlüğü açısından dikkatli uygulanmalıdır. Yanlış veri tipi değişiklikleri veya koşulsuz yapılan işlemler veri kaybına ya da sistem hatalarına yol açabileceğinden, ALTER komutunun test ortamında denenerek ve yedekleme alınarak kullanılması önerilir.

Sorgu Performansı ve İyi Uygulamalar

Sorgu performansı, veritabanı sistemlerinin genel verimliliğini doğrudan etkiler. Seçilen sütun sayısı, filtreleme koşullarının doğruluğu, sıralama işlemlerinin gerekliliği ve indeks kullanımı performans üzerinde belirleyici rol oynar.
Gereksiz kolon seçimi, bilinçsiz sıralama ve optimize edilmemiş filtreleme işlemleri sistem kaynaklarının verimsiz kullanılmasına neden olur. Etkili sorgu yazımı; doğruluk, okunabilirlik ve performans ilkeleri doğrultusunda geliştirilmelidir. Bu bağlamda SQL sorguları, modern bilgi sistemlerinde verinin anlamlı bilgiye dönüştürülmesini sağlayan temel araçlardan biri olarak değerlendirilmektedir.

VERİLERİ GRUPLAYARAK ANALİZ ETMEK

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. 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.

Sorguda gruplama operatörü kullanılmışsa sadece gruplanan alanlar ve gruplama fonksiyonlarından 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 sütun adlarının gruplamaya alınmış olması gerekir.

GRUP FONKSİYONLARI

Gruplama fonksiyonları bir sütundaki verileri işleme tabi tutarak tek bir değer döndürür. Bu fonksiyonlardan en çok kullanılan beş tanesi: "MAX", "MIN", "AVG", "SUM" ve "COUNT". Gruplama fonksiyonları, parametre olarak aldıkları sütundaki "null" olmayan değerler üzerinden işlem yapmaktadır.

Veriler gruplanmadığında yani "Group By" ifadesi kullanılmadığında tüm veriler tek bir grup içinde kabul edilir. Dolayısıyla gruplanmayan veriler üzerinde de grup fonksiyonları kullanılabilir.

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

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.

MIN() Fonksiyonu

"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.

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 de kullanılabilmektedir. "COUNT" fonksiyonunu diğer gruplama fonksiyonlarından ayıran özellik ise bu fonksiyonun, "*" parametresiyle çalışabilen tek gruplama fonksiyonu olmasıdır.

AVG() Fonksiyonu

"AVG" fonksiyonu, parametre olarak aldığı değerlerin aritmetiksel ortalamasını almak için kullanılır. Bu fonksiyon sadece sayısal türdeki veriler ile çalışmaktadır.

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.

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.

BİRDEN FAZLA SÜTUNA GÖRE GRUPLAMA

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

GRUP KOŞULLARININ KULLANIMI

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.

"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.

"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.

İlişkisel veri tabanı türünde, veri tekrarının önüne geçilebilmesi için bir varlığa ilişkin veriler farklı 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. Bunun için birleştirme (join) işlemi uygulanır. Birden fazla tabloyu birleştirirken de tabloların ortak olan sütunları bir koşul ifadesiyle eşleştirilir.
Birleştirme şartındaki eşleştirme sonrası iki tablodan da sadece eşleşen kayıtların görüntülenmesinin sağlandığı, eşiti olan birleştirme, klasik birleştirme veya iç birleştirme ile yapılabilmektedir.
Birleşme şartına uymayan kayıtların da listelendiği birleştirme türüne de eşiti olmayan birleştirme denilmektedir.

Takma Ad Verilmesi

Sorgulama esnasında veri tabanında yeralan bir tabloya veya kolona, takma ad verilmesi, sorgunun yazılmasını kolaylaştırır ve anlaşılır kılar. 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
Birleştirilen tablolarda aynı isimde sütunlar bulunması durumunda bu kolonlardaki verilere erişilirken tablo adı uzun hâliyle veya takma adı ile kullanılmalıdır.

ÇOKLU TABLOLARIN KULLANILMASI

Normalizasyon formları uygulanarak bir çok tabloya dağıtılmış olan verilere tek bir sorguda ihtiyaç duyulan durumlar olacaktır. Bu durumda birden çok tabloyu bir araya getirip tablolar üzerindeki ilgili alanlarla tabloları ilişkilendirip birleştirmemiz gerekecektir.
Birden fazla tabloyu birleştirmek için kullanılan birleştirme yöntemleri aşağıdaki gibidir:

- 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 yer alan kayıtları (diğer tabloda eşleşen bir kayıt olmasa dahi) 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. En nadir ihtiyaç duyulan bu birleştirme türünde ilişkili olsun veya olmasın birleştirilecek tabloların tüm satırları listelenmektedir. 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 birleştirmede “From” deyiminden sonra ilk tablonun adı ile ikinci tablonun adı arasına “Cross Join” ifadesi yazılmaktadır.
Örnek olarak haftalık ders programı oluşturmak için “Gunler” ve “Saatler” tablolarını “Cross Join” ile birleştirelim. Gunler tablosunda 5 kayıt, Saatler tablosunda ise 8 kayıt bulunmaktadır. Dolayısıyla çapraz birleştirme sonrası oluşan listede toplam 40 satır sonuç gelecektir.

EŞİTİ OLAN BİRLEŞTİRME

Bu birleştirmede bir tablo üzerinde bulunan değerler diğer tablonun ilgili alanı ile eşleştirilip sadece eşleşen kayıtlar birleştirilir.
Eşiti olan birleştirmeyi “iç birleştirme” (Inner Join) ve klasik birleştirme ile gerçekleştirebiliriz.

Klasik Birleştirme

Bu birleştirme türünde tabloların eşleştirilmesi için gerekli koşul, “Where” ifadesinden sonra kullanılır. İki veya daha fazla tabloda yer alan aynı türde verileri içeren alanlar, “Where” ifadesinden sonra kullanılan koşul ile eşitlenerek klasik birleştirme işlemi gerçekleştirilir.

İç Birleştirme (Inner Join)

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 birleştirme aşağıda görüldüğü gibi iki tablonun alanlarının ortak olduğu kayıtları getirmektedir. Taralı alandaki kayıtların hem tablo1 de hem de tablo2 de karşılığı vardır.

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 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.

EŞİTİ OLMAYAN BİRLEŞTİRME

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.

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.

Sol dış birleştirme yukarıdaki de görüldüğü gibi birinci tablonun tüm kayıtlarını ve ikinci tablonun ise koşula uyan kayıtlarını getirmektedir.

Sağ Dış Birleştirme (Right Outer Join)

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.

Sağ dış birleştirme yukarıdaki şekilde görüldüğü gibi ikinci (sağdaki) tablonun tüm kayıtlarını ve birinci (soldaki) tablonun ise koşula uyan kayıtlarını getirmektedir.

Tüm Dış Birleştirme (Full Outer Join)

Tüm dış birleştirme (“Full Outer Join”) 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.

Veri Tabanlarında Veri Eklemenin Önemi

Bu ünitede, ilişkisel veri tabanı yönetim sistemlerinde veri üretiminin ve veri bütünlüğünün temelini oluşturan tabloya satır ekleme işlemleri ayrıntılı biçimde ele alınmıştır. Günümüzde bilgi sistemleri yalnızca verileri depolamak için değil; analiz yapmak, rapor üretmek, karar süreçlerini desteklemek ve iş akışlarını yönetmek için kullanılmaktadır. Bu nedenle verinin doğru, tutarlı ve güvenli biçimde veri tabanına eklenmesi kritik bir öneme sahiptir. Veri tabanına yanlış girilen bilgiler, hatalı raporlamalara, yanlış kararların alınmasına ve sistem güvenliğinin zedelenmesine neden olabilir. Bu nedenle veri ekleme işlemleri, veri tabanı yönetiminin en temel ve en hassas adımlarından biridir. Ünite boyunca SQL Server 2019 ortamı temel alınmış ve veri ekleme süreçleri hem kuramsal hem de uygulamalı boyutlarıyla ele alınmıştır. Öğrencinin yalnızca komutları ezberlemesi değil, veri ekleme işleminin mantığını kavraması hedeflenmiştir. Böylece öğrenci, gerçek hayatta karşılaşacağı veri yönetimi problemlerini çözebilecek bir bakış açısı kazanacaktır.

INSERT Deyimi ve Veri Ekleme Süreci

Ünitenin ilk bölümünde, tabloya satır ekleme kavramı açıklanmış ve veri eklemenin iş süreçleriyle ilişkisi ortaya konmuştur. Yeni bir öğrencinin sisteme kaydedilmesi, bir hastanın kaydının oluşturulması, bir ürünün stoklara eklenmesi veya bir siparişin sisteme girilmesi gibi örnekler üzerinden veri eklemenin günlük hayatla bağlantısı kurulmuştur. Bu örnekler, veri tabanı işlemlerinin yalnızca teknik bir süreç olmadığını; aynı zamanda organizasyonların işleyişini doğrudan etkileyen stratejik bir süreç olduğunu göstermektedir.

İkinci bölümde, SQL Server 2019'da kullanılan INSERT deyiminin temel sözdizimi ayrıntılı biçimde açıklanmıştır. INSERT komutu ile tabloya yeni kayıt eklenebileceği, sütun listesi belirtilerek veri girmenin daha güvenli olduğu vurgulanmıştır. Sütun listesi yazılmadan yapılan veri ekleme işlemlerinin tablo yapısı değiştiğinde hatalara yol açabileceği belirtilmiştir. Ayrıca tek komutla birden fazla satır ekleme yöntemi tanıtılarak toplu veri girişinin nasıl daha verimli gerçekleştirilebileceği gösterilmiştir.

Bu bölümde ayrıca veri ekleme sırasında sık karşılaşılan hatalar da ele alınmıştır. Sütun sayısı ile değer sayısının uyuşmaması, veri türü uyumsuzluğu, NOT NULL sütunlara NULL girilmesi ve PRIMARY KEY ihlali gibi hataların veri bütünlüğünü nasıl bozabileceği açıklanmıştır. Öğrencinin bu hataları tanıyabilmesi ve önleyebilmesi hedeflenmiştir.

NULL Değer Yönetimi

Ünitenin üçüncü bölümünde NULL kavramı ayrıntılı olarak ele alınmıştır. NULL değerinin boş veya sıfır olmadığı; değerin bilinmediğini veya henüz girilmediğini ifade ettiği açıklanmıştır. Bu ayrımın özellikle veri analizi ve raporlama süreçlerinde önemli olduğu vurgulanmıştır. NULL değer eklenebilen sütunlar ile NOT NULL tanımlı sütunlar arasındaki farklar gösterilmiş ve veri ekleme işlemlerinde bu kısıtların nasıl dikkate alınması gerektiği açıklanmıştır.

Ayrıca NULL değer içeren kayıtların sorgulanmasında eşitlik operatörü yerine IS NULL ifadesinin kullanılması gerektiği örneklerle gösterilmiştir. Bu bölüm, öğrencinin NULL değer yönetimini doğru yaparak veri kalitesini koruyabilmesini amaçlamaktadır.

Başka Tablodan Veri Ekleme (INSERT … SELECT)

Dördüncü bölümde INSERT … SELECT yapısı ele alınmıştır. Gerçek uygulamalarda verilerin çoğu zaman farklı tablolardan alınarak başka bir tabloya aktarılması gerektiği belirtilmiştir. Özellikle arşivleme, veri taşıma, yedekleme ve raporlama işlemlerinde bu yöntemin sık kullanıldığı açıklanmıştır. Öğrencinin belirli koşulları sağlayan kayıtları seçerek başka bir tabloya ekleyebilmesi amaçlanmıştır.

Bu yapı sayesinde büyük veri setlerinin hızlı ve düzenli biçimde aktarılabileceği gösterilmiştir. Ayrıca kaynak ve hedef tablolar arasındaki sütun uyumunun önemine değinilmiştir.

SELECT … INTO … FROM Deyimi

Beşinci bölümde SELECT … INTO … FROM deyimi açıklanmıştır. Bu yapı, sorgu sonucuna göre yeni bir tablo oluşturmak için kullanılır. INSERT … SELECT ile SELECT … INTO arasındaki farklar karşılaştırmalı olarak verilmiş ve hangi durumlarda hangi yapının tercih edilmesi gerektiği açıklanmıştır.

SELECT … INTO yapısının özellikle analiz amaçlı veri setleri oluşturma, geçici tablolar hazırlama ve raporlama süreçlerinde önemli olduğu vurgulanmıştır. Bu sayede öğrenci, veri tabanı üzerinde daha

esnek ve etkili işlemler gerçekleştirebilecektir.

Veri Bütünlüğü ve Stratejik Veri Yönetimi

Ünite boyunca veri ekleme işlemlerinin veri bütünlüğü ile ilişkisi vurgulanmıştır. Doğru veri türü kullanımı, kısıtların doğru tanımlanması ve veri giriş hatalarının önlenmesi, veri tabanı yönetiminin temel ilkeleri arasında yer almaktadır. Veri ekleme işlemlerinin dikkatli yapılması, sistem güvenliğini ve veri doğruluğunu artırır.

Bu ünite, öğrencinin veri tabanını yalnızca kullanan değil, bilinçli biçimde yöneten bir yaklaşım geliştirmesini hedeflemektedir. Veri tabanı sistemlerinin başarısı, doğru veri yönetimine bağlıdır.

Gelecek Ünitelerle Bağlantı

Bu ünitede kazanılan bilgi ve beceriler, ilerleyen ünitelerde ele alınacak UPDATE, DELETE, view, stored procedure ve fonksiyon gibi ileri düzey veri tabanı konularının temelini oluşturacaktır. Veri ekleme süreçlerini doğru anlayan bir öğrenci, veri tabanı yönetim sistemlerini daha etkin ve güvenli biçimde kullanabilecektir.

Genel Değerlendirme

Sonuç olarak bu ünite, öğrencinin SQL Server 2019 ortamında veri ekleme işlemlerini doğru biçimde yapabilmesini, NULL değerleri yönetebilmesini, başka tablolardan veri aktarabilmesini ve gerektiğinde yeni tablolar oluşturabilmesini hedeflemiştir. Bu beceriler, veri tabanı yönetimi alanında sağlam bir temel oluşturur ve öğrencinin ilerleyen konuları daha kolay öğrenmesini sağlar.

DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI İlişkili Tablolar ile Sorgu Hazırlamak
ÜNİTE NO 10
YAZAR Dr.Öğr.Üyesi Önder YILDIRIM

İlişkili Tablolarda Veri Yönetimi ve Bütünlük

Bu ünitede, ilişkisel veri tabanı yönetim sistemlerinde en kritik operasyonlardan ikisi olan güncelleme (UPDATE) ve silme (DELETE) işlemleri, özellikle ilişkili tablolar bağlamında ele alınmıştır. Modern veri tabanı tasarımlarında veriler genellikle tek bir tabloda değil, birbirleriyle ilişkili birçok tabloda tutulur. Bu durum, veri bütünlüğünü sağlamak, tutarlı güncellemeler yapmak ve yanlış silmeleri önlemek açısından sorgu yazımının daha bilinçli ve sistematik olmasını zorunlu kılar. Bu nedenle ünite boyunca yalnızca temel UPDATE ve DELETE sözdizimleri değil, bu işlemlerin JOIN ve alt sorgu (subquery) gibi daha ileri tekniklerle nasıl güvenli biçimde gerçekleştirileceği ayrıntılı biçimde açıklanmıştır.

Ünitenin ilk bölümünde, Tablodaki Verileri Güncelleme (UPDATE) başlığı altında güncelleme işleminin mantığı ele alınmıştır. Güncelleme yaparken her zaman üç temel sorunun yanıtlanması gerektiği vurgulanmıştır: Hangi tablo güncellenecek? Hangi sütun(lar) değiştirilecek? Hangi kayıt(lar) etkilenecek? Bu soruların net biçimde belirlenmemesi durumunda veri kaybı, tutarsızlık veya istenmeyen toplu güncellemeler ortaya çıkabileceği belirtilmiştir. Basit UPDATE komutunun temel sözdizimi açıklanmış ve WHERE koşulunun önemi özellikle vurgulanmıştır. WHERE koşulu unutulduğunda tüm satırların güncellenebileceği ve bunun geri döndürülemez hatalara yol açabileceği somut örneklerle gösterilmiştir.

Ardından UPDATE Deyimi (JOIN ve Alt Sorgu ile Güncelleme) başlığı altında, ilişkili tablolar üzerinde güncelleme yapmanın yöntemleri sistematik biçimde ele alınmıştır. Gerçek dünya veri tabanlarında bir tablodaki verilerin çoğu zaman başka bir tablodaki değerlere göre güncellenmesi gerektiği belirtilmiş ve bu amaçla UPDATE … JOIN yapısının nasıl kullanılacağı gösterilmiştir. Örneğin, belirli bir bölümdeki öğrencilerin not ortalamasının artırılması gibi senaryolar üzerinden JOIN ile güncellemenin avantajları açıklanmıştır. JOIN kullanımı sayesinde güncellemenin yalnızca doğru kayıtları etkilemesi sağlanmıştır. Alternatif olarak alt sorgu (subquery) ile güncelleme yöntemi de tanıtılmış; önce alt sorgunun gerekli değeri bulduğu, ardından ana sorgunun bu değere göre güncelleme yaptığı açıklanmıştır. Bu iki yaklaşımın (JOIN ve subquery) birbirinin yerine kullanılabileceği, ancak performans ve okunabilirlik açısından bağlama göre tercih edilmesi gerektiği vurgulanmıştır.

Ünitenin ikinci ana bölümünde Tablolardan Veri Silme (DELETE) konusu ele alınmıştır. Silme işlemlerinin güncellemeden bile daha riskli olduğu; çünkü geri dönüşsüz veri kaybına yol açabileceği belirtilmiştir. Bu nedenle DELETE işlemlerinde WHERE koşulunun önemi bir kez daha altı çizilmiştir. Basit DELETE sözdizimi açıklanmış ve yalnızca belirli bir kaydı silmenin nasıl yapılacağı gösterilmiştir. Ayrıca toplu silmelerin (örneğin belirli bir kriteri karşılayan tüm kayıtların silinmesi) hangi koşullarda gerekli olabileceği tartışılmıştır.

Daha sonra DELETE Deyimi Yapısı (JOIN ve Alt Sorgu ile Silme) başlığı altında, ilişkili tablolar üzerinden silme işlemleri ele alınmıştır. Bir tablodaki kayıtların başka bir tablodaki değerlere göre silinmesi gerektiğinde DELETE … JOIN yapısının nasıl kullanılacağı gösterilmiştir. Örneğin, belirli bir bölümdeki tüm öğrencilerin silinmesi gibi senaryolar üzerinden JOIN ile silmenin mantığı açıklanmıştır. Alternatif olarak alt sorgu kullanılarak da aynı işlemin gerçekleştirilebileceği örneklerle gösterilmiştir. Bu iki yöntemin de geçerli olduğu, ancak büyük veri setlerinde performans ve okunabilirlik açısından dikkatli seçim yapılması gerektiği belirtilmiştir.

Ünite boyunca özellikle veri bütünlüğü kavramı sürekli olarak vurgulanmıştır. Yabancı anahtar (Foreign Key – FK) kısıtlarının silme işlemlerini engelleyebileceği; bu durumda ya önce bağımlı kayıtların silinmesi ya da ON DELETE CASCADE gibi mekanizmaların kullanılması gerektiği açıklanmıştır. Ayrıca yanlış JOIN kullanımı, eksik WHERE koşulu veya hatalı alt sorguların yanlış kayıtların güncellenmesine ya da silinmesine yol açabileceği örneklerle gösterilmiştir.

Genel olarak bu ünite, öğrencinin ilişkisel tablolar üzerinde yalnızca veri okuma değil, veriyi güvenli biçimde değiştirme ve silme becerisini geliştirmeyi hedeflemiştir. Öğrenci; basit UPDATE ve DELETE komutlarını, JOIN ile güncelleme ve silme tekniklerini, alt sorgu ile koşullu işlemleri ve yaygın hataları öğrenmiştir. Ünitenin sonunda öğrencinin, veri bütünlüğünü koruyarak güncelleme ve silme işlemlerini gerçekleştirebilmesi, uygun durumlarda JOIN veya alt sorgu kullanabilmesi ve riskli işlemleri önleyebilmesi beklenmektedir. Bu yetkinlik, ölçeklenebilir, güvenli ve sürdürülebilir veri tabanı yönetiminin temel taşlarından biri olarak değerlendirilmiştir.

DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI Görüntü (View), Saklı Yordam (Stored Procedure) ve Fonksiyonlar
ÜNİTE NO 11
YAZAR Dr.Öğr.Üyesi Önder YILDIRIM

SQL Server Gelişmiş Nesneleri: View, SP ve Fonksiyonlar

Bu ünitede, SQL Server 2019 ortamında veri tabanı uygulamalarını daha düzenli, güvenli ve sürdürülebilir hâle getiren üç temel yapı ayrıntılı olarak ele alınmıştır: Görüntüler (Views), Saklı Yordamlar (Stored Procedures) ve Kullanıcı Tanımlı Fonksiyonlar (User-Defined Functions – UDF). Modern bilgi sistemlerinde yalnızca tablolar üzerinde doğrudan sorgu yazmak çoğu zaman yeterli değildir. Kurumsal ölçekli veri tabanı uygulamalarında karmaşık sorguların yönetilmesi, iş kurallarının standartlaştırılması, veri güvenliğinin sağlanması ve performansın artırılması için veri tabanı katmanında ek programatik yapıların kullanılması gereklidir. Bu ünite, öğrencinin veri tabanı tasarımında daha profesyonel yaklaşımlar geliştirmesini hedeflemiştir.

Ünitenin ilk bölümünde View (Görüntü) kavramı incelenmiştir. View'lerin bir veya birden fazla tablodan türetilmiş sanal tablolar olduğu, fiziksel veri saklamadıkları ancak bir SELECT sorgusunun mantıksal katmanı olarak çalıştıkları açıklanmıştır. View'ler sayesinde karmaşık JOIN işlemleri tek bir görünüm altında toplanarak sorgular sadeleştirilebilir. Ayrıca yalnızca gerekli sütunların gösterilmesiyle veri güvenliği sağlanabilir ve kullanıcıların hassas verilere erişimi sınırlandırılabilir. CREATE VIEW, ALTER VIEW ve DROP VIEW deyimleri ile view oluşturma, güncelleme ve silme işlemleri örneklerle gösterilmiş; view'lerin raporlama sistemleri, veri soyutlama ve standart veri erişimi açısından önemli avantajlar sunduğu vurgulanmıştır.

İkinci bölümde Stored Procedure (Saklı Yordam) yapısı ele alınmıştır. Stored procedure'lar, veri tabanı içinde derlenmiş ve saklanmış SQL komutlarıdır. Tekrar eden işlemleri standartlaştırır, ağ trafiğini azaltır ve yürütme planlarının önbelleğe alınması sayesinde performansı artırır. Parametre alabilmeleri sayesinde dinamik işlemler yapılabilir ve uygulama katmanındaki iş kurallarının veri tabanında merkezileştirilmesine katkı sağlar. CREATE PROCEDURE deyimi ile saklı yordam oluşturma, EXEC komutu ile çalıştırma, input-output parametre kullanımı ve temel doğrulama (validation) işlemleri ayrıntılı biçimde açıklanmıştır. Ayrıca stored procedure'ların veri bütünlüğünü korumada, hata yönetiminde ve güvenlik politikalarının uygulanmasında nasıl kullanılabileceği örneklerle gösterilmiştir.

Ünitenin üçüncü bölümünde User-Defined Functions (Kullanıcı Tanımlı Fonksiyonlar) incelenmiştir. Fonksiyonların belirli bir hesaplama veya dönüşüm işlemini gerçekleştiren ve mutlaka bir değer döndüren programatik yapılar olduğu belirtilmiştir. Skaler fonksiyonlar tek değer döndürürken, inline table-valued fonksiyonlar bir SELECT sorgusu sonucunu tablo olarak döndürür. Çok ifadeli fonksiyonlar ise daha karmaşık işlemler gerçekleştirebilir. Fonksiyonların özellikle SELECT sorguları içinde hesaplama yapmak, veri dönüştürmek ve kod tekrarını azaltmak amacıyla kullanıldığı açıklanmıştır. Stored procedure ile fonksiyon arasındaki farklar karşılaştırmalı olarak verilmiş ve fonksiyonların veri değiştirme işlemleri için kullanılmaması gerektiği vurgulanmıştır.

Genel olarak bu ünite, öğrencinin SQL Server 2019 ortamında veri tabanı uygulamalarını daha modüler ve güvenli biçimde tasarlayabilmesini amaçlamıştır. View'ler veri soyutlama ve güvenlik için, stored procedure'lar iş süreçlerini standartlaştırmak ve performansı artırmak için, fonksiyonlar ise hesaplama ve dönüşüm işlemlerini yeniden kullanılabilir hâle getirmek için kullanılan güçlü araçlardır. Bu yapılar birlikte kullanıldığında veri tabanı sistemlerinde bakım kolaylığı, ölçeklenebilirlik ve sürdürülebilirlik sağlanır. Ünitenin sonunda öğrencinin view, stored procedure ve fonksiyon oluşturma ve kullanma becerisi kazanarak daha profesyonel veri tabanı uygulamaları geliştirebilmesi hedeflenmiştir.

Veritabanı Yönetimi Yapmak

Veritabanı yönetimi, modern bilgi sistemlerinin güvenli, sürdürülebilir ve yüksek performanslı biçimde çalışmasını sağlayan teknik ve yönetsel süreçlerin bütünüdür. Günümüzde kurumlar yalnızca veriyi depolamakla yetinmemekte; bu veriyi analiz etmekte, raporlamakta ve stratejik karar süreçlerine entegre etmektedir. Bu durum, veritabanı yönetimini basit bir depolama süreci olmaktan çıkararak performans, güvenlik, erişim kontrolü, hata toleransı ve süreklilik gerektiren çok katmanlı bir uzmanlık alanına dönüştürmüştür. SQL Server ortamında bu süreçler SQL Server Management Studio (SSMS) aracılığıyla yürütülmekte ve hem grafik arayüz hem de T-SQL komutları kullanılarak yönetilebilmektedir.

Yönetici Paneli

SQL Server Management Studio, SQL Server veritabanı motoru ile etkileşim kurmayı sağlayan tümleşik bir yönetim ortamıdır. SSMS, veritabanı yöneticilerine hem görsel hem de komut tabanlı yönetim imkânı sunar. Sunucu bağlantısı aşamasında Windows Authentication veya SQL Server Authentication yöntemleri kullanılmaktadır. Kimlik doğrulama mekanizması, sistem güvenliğinin ilk katmanını oluşturur ve erişim denetimi açısından kritik öneme sahiptir.

Object Explorer paneli, sunucu üzerindeki veri tabanlarını ve ilgili nesneleri hiyerarşik yapıda görüntüler. Tablolar, görünümler, indeksler, saklı yordamlar ve roller bu panel üzerinden yönetilir. Bu yapı, SQL Server'ın sistem katalogları ile ilişkilidir ve metadata üzerinden çalışır. Kullanıcılar bu panel aracılığıyla CREATE, ALTER ve DROP komutlarını dolaylı olarak çalıştırabilmektedir.

Query Editor ise T-SQL komutlarının yazıldığı ortamdır. IntelliSense özelliği, sistem nesnelerini algılayarak otomatik tamamlama desteği sunar. Yazılan sorgular, SQL Server'ın Query Optimizer bileşeni tarafından analiz edilir ve maliyet tabanlı bir yürütme planı oluşturulur. Bu süreçte istatistikler, indeks yapıları ve veri dağılımı dikkate alınır.

Veritabanı İşlemleri

Veritabanı işlemleri; oluşturma, güncelleme, özellik düzenleme ve silme aşamalarını kapsar. Bu işlemler, storage engine ve transaction log mekanizmasının koordineli çalışmasını gerektirir.

Veritabanı oluşturma

Veritabanı oluşturma işlemi grafik arayüz üzerinden yapılabildiği gibi CREATE DATABASE komutu ile de gerçekleştirilebilir. Bu aşamada veri dosyası (data file) ve işlem günlüğü dosyası (log file) yapılandırılır. Dosya boyutları, otomatik büyüme (Autogrowth) parametreleri ve dosya konumları performans üzerinde doğrudan etkilidir. Yanlış yapılandırılmış büyüme ayarları sık disk erişimine ve I/O darboğazına yol açabilir.

Veritabanının güncellenmesi

Veritabanı güncelleme işlemleri ALTER DATABASE komutu ile yapılır. Bu komut aracılığıyla veritabanı salt okunur (READ_ONLY) veya çok kullanıcılı (MULTI_USER) modlara alınabilir. Ayrıca dosya büyüme parametreleri ve uyumluluk düzeyi değiştirilebilir. Arayüzdeki "Refresh" işlemi ise yalnızca nesne listesini günceller; fiziksel veri üzerinde değişiklik yapmaz.

Veritabanı Özelliklerinin Düzenlenmesi

Veritabanı özellikleri ekranında Files, Options, Security ve Compatibility Level gibi kritik ayarlar bulunur. Recovery Model seçimi transaction log yönetimini belirler. Full Recovery modeli tüm işlemleri loglar ve point-in-time restore imkânı sunar. Simple Recovery modeli log dosyasını otomatik olarak temizler ve yönetimi kolaylaştırır. Bulk-Logged modeli ise toplu veri işlemlerinde minimal logging sağlar.

Compatibility Level ayarı, sorgu iyileştiricinin (optimizer) davranışını etkiler. Yeni sürümlerde gelişmiş cardinality estimation algoritmaları ve performans iyileştirmeleri bulunur. Yanlış uyumluluk düzeyi, beklenmeyen performans değişimlerine neden olabilir.

Veritabanı Silme

Veritabanı silme işlemi DROP DATABASE komutu ile gerçekleştirilir. Aktif bağlantılar varsa işlem başarısız olabilir. Bu durumda ALTER DATABASE komutu ile veritabanı SINGLE_USER moduna alınabilir. Silme işleminden önce BACKUP DATABASE komutu ile yedek alınması önerilir.

Sorgulama İşlemleri

Sorgulama işlemleri T-SQL komutları ile yürütülür. SELECT, INSERT, UPDATE ve DELETE komutları temel veri manipülasyon işlemleridir. JOIN işlemleri ise birden fazla tabloyu birleştirmek

için kullanılır. INNER JOIN, LEFT JOIN ve RIGHT JOIN gibi farklı türler mevcuttur.
SQL Server, sorgu yürütme sırasında Nested Loop, Merge Join veya Hash Match algoritmalarından birini seçer. Bu seçim, indeks yapısı ve veri büyüklüğüne bağlıdır. İndeks bulunmayan tablolarda genellikle Table Scan veya Index Scan işlemi görülür; bu durum yüksek I/O maliyetine neden olabilir. Sorgu performansının artırılması için gereksiz kolon seçiminin önlenmesi, uygun WHERE koşullarının yazılması ve indekslerin doğru tasarlanması gerekir. Ayrıca UPDATE STATISTICS komutu ile istatistiklerin güncel tutulması performans açısından önemlidir.

İstemci İstatistiği

İstemci istatistikleri, sorguların performansını istemci tarafında analiz etmeyi sağlar. Include Client Statistics seçeneği aktif edildiğinde Total Execution Time, Client Execution Time, Wait Time ve Bytes Received gibi metrikler görüntülenir.
Total Execution Time sorgunun toplam süresini, Wait Time ise sunucudan yanıt bekleme süresini ifade eder. Bytes Received değeri ağ trafiğini gösterir. Performans karşılaştırması yapılırken aynı sorgu birden fazla kez çalıştırılmalı ve ortalama değerler dikkate alınmalıdır.
Bu analiz, özellikle farklı indeks yapıları veya farklı JOIN yazım teknikleri karşılaştırılırken anlamlı sonuçlar verir.

Yürütme Planları

Execution Plan, sorgunun fiziksel işlem adımlarını gösterir. Estimated Execution Plan tahmini maliyetleri sunarken, Actual Execution Plan gerçek çalışma istatistiklerini içerir. Yürütme planı sağdan sola doğru okunur.
Plan üzerinde görülen Index Seek işlemi hedeflenen satırlara doğrudan erişimi ifade eder ve genellikle yüksek performanslıdır. Index Scan veya Table Scan işlemleri ise tüm veri kümesinin tarandığını gösterir ve daha maliyetlidir.
Maliyet yüzdeleri CPU ve I/O tahminlerine dayanır. Yüksek maliyetli operatörler performans iyileştirme sürecinde öncelikli olarak ele alınmalıdır.

Komut Satırından Veritabanı İşlemleri

SQLCMD modu, komut satırı üzerinden veritabanı yönetimine imkân tanır. SQLCMD Mode etkinleştirildiğinde özel komutlar kullanılabilir. GO komutu batch ayırıcıdır ve komut bloğunu tekrar çalıştırabilir. OUT komutu sorgu çıktısını dosyaya yönlendirirken, ERROR komutu hata çıktısını kaydeder. QUIT komutu ise oturumu sonlandırır.
Komut satırı yönetimi, özellikle otomasyon, zamanlanmış görevler ve raporlama süreçlerinde avantaj sağlar. SQL Agent ile birlikte kullanıldığında planlanmış bakım ve yedekleme işlemleri otomatikleştirilebilir.

Genel Değerlendirme

Veritabanı yönetimi; storage engine, query optimizer ve güvenlik katmanlarının koordineli çalışmasını gerektirir. Recovery Model seçimi, dosya büyüme ayarları ve uyumluluk düzeyi performansı doğrudan etkiler. İstemci istatistikleri ve yürütme planları birlikte kullanılarak sorgu darboğazları belirlenebilir. Grafik arayüz kullanıcı dostu bir deneyim sunarken, T-SQL ve SQLCMD komutları daha esnek ve otomasyona uygun çözümler sağlar. Etkili bir veritabanı yönetimi için hem görsel hem de komut tabanlı araçların bilinçli biçimde kullanılması gereklidir.
Sonuç olarak veritabanı yönetimi; teknik bilgi, analitik düşünme ve stratejik planlama gerektiren çok boyutlu bir uzmanlık alanıdır. Doğru yapılandırma, düzenli izleme ve performans optimizasyonu süreçleri, bilgi sistemlerinin güvenilirliğini ve sürekliliğini garanti altına alır.

DERS ADI Veri Tabanı Yönetim Sistemleri
ÜNİTE ADI Veri Tabanı Güvenliğini Sağlamak
ÜNİTE NO 13
YAZAR Doç.Dr. HAMZA POLAT

VERİTABANLARINDA GÜVENLİK VE ERİŞİM DENETİMİ

Veritabanları, modern bilgi sistemlerinin temel yapı taşlarından biridir ve günümüzde çok sayıda kullanıcı tarafından eş zamanlı olarak kullanılan kritik veri kümelerini barındırır. Üniversite bilgi sistemleri, finansal uygulamalar, sağlık kayıtları ve kamu hizmetleri gibi alanlarda tutulan veriler çoğu zaman gizli, hassas ve bütünlüğü korunması gereken niteliktedir. Bu nedenle veritabanı güvenliği, yalnızca teknik bir detay değil; sistemin güvenilirliği, sürekliliği ve yasal uyumluluğu açısından temel bir gereklilik olarak değerlendirilir. Veritabanı güvenliği genel olarak, veriye kimlerin erişebileceğini, bu veriler üzerinde hangi işlemlerin yapılabileceğini ve bu işlemlerin nasıl denetleneceğini belirleyen mekanizmalar bütünü olarak tanımlanabilir.

Bu bölümde ele alınan güvenlik yaklaşımları, yetkilendirme ve kimlik doğrulama mekanizmaları etrafında şekillenmektedir. Kimlik doğrulama, sisteme erişmeye çalışan kullanıcının gerçekten iddia ettiği kişi olup olmadığını belirlerken; yetkilendirme, kimliği doğrulanmış kullanıcının hangi verilere erişebileceğini ve bu veriler üzerinde hangi işlemleri yapabileceğini tanımlar. Bu iki mekanizmanın birlikte ve uyumlu şekilde çalışması, güvenli bir veritabanı sisteminin ön koşuludur. Bölüm boyunca, bu temel kavramlar ilişkisel veritabanı sistemleri bağlamında ele alınmış ve gerçekçi örneklerle desteklenmiştir.

YETKİLER, YETKİLENDİRME VE ROLLER

Yetkilendirme, bir kullanıcının veya rolün veritabanı nesneleri üzerinde hangi işlemleri yapabileceğini belirleyen temel güvenlik mekanizmasıdır. İlişkisel veritabanı sistemlerinde okuma, ekleme, güncelleme ve silme gibi işlemler ayrı ayrı yetkilendirilir. Bu yapı, kullanıcılara yalnızca görevlerini yerine getirebilmeleri için gerekli olan işlemleri yapma izni verilmesine olanak tanır. Bu yaklaşım, en az yetki ilkesi olarak adlandırılır ve veritabanı güvenliğinin temelini oluşturur. Kullanıcılara gereğinden fazla yetki verilmesi, veri gizliliği ve veri bütünlüğü açısından ciddi riskler doğurur.

Yetkilendirme işlemleri kullanıcı bazında yapılabildiği gibi, rol bazında da gerçekleştirilebilir. Roller, ortak yetkilere sahip kullanıcı gruplarını temsil eder ve yetkilendirme yönetimini büyük ölçüde kolaylaştırır. Özellikle kullanıcı sayısının fazla olduğu sistemlerde, yetkilerin tek tek kullanıcılara verilmesi yerine roller üzerinden tanımlanması daha sürdürülebilir bir yapı sunar. Roller, organizasyonel görev ve sorumlulukların veritabanına yansıtılmasını sağlar. Asistan, öğretim üyesi veya yönetici gibi roller, gerçek dünyadaki işlevlere karşılık gelecek şekilde tasarlanabilir.

KULLANICI YÖNETİMİ

Veritabanı güvenliği yalnızca yetkilerin tanımlanmasıyla sınırlı değildir. Sisteme kimlerin bağlanabileceğinin kontrol edilmesi de en az yetkilendirme kadar önemlidir. Kullanıcı yönetimi, kullanıcı hesaplarının oluşturulması, yetkilendirilmesi, izlenmesi ve gerektiğinde sistemden kaldırılması süreçlerini kapsar. Bu süreçler, kullanıcı yaşam döngüsü yönetimi olarak adlandırılır ve güvenliğin sürekliliği açısından kritik öneme sahiptir.

Kullanıcıların görevleri zamanla değişebilir veya kullanıcılar sistemden tamamen ayrılabilir. Bu durumlarda yetkilerin güncellenmemesi veya hesapların silinmemesi, görünmez güvenlik açıklarının oluşmasına yol açar. Bu nedenle kullanıcı hesapları ve rol atamaları düzenli olarak gözden geçirilmelidir. Ayrıca kullanıcıların güçlü parola politikalarına uyması, parolaların güvenli biçimde saklanması ve başarısız giriş denemelerinin izlenmesi de kullanıcı yönetiminin önemli bileşenleridir.

GÖRÜNÜMLER ÜZERİNDEN YETKİLENDİRME

Görünümler, veritabanı güvenliğinde ek bir koruma katmanı sağlar. Görünümler sayesinde kullanıcılar, temel tablolara doğrudan erişmeden yalnızca ihtiyaç duydukları satırları veya sütunları görebilir. Bu yaklaşım, özellikle hassas verilerin gizlenmesi gereken durumlarda büyük avantaj sağlar. Örneğin bir danışmanın yalnızca öğrenci notlarını görmesi, ancak adres veya kimlik bilgilerine erişememesi görünümler aracılığıyla sağlanabilir.

Güncellenebilir görünümler, kullanıcıların görünüm üzerinden veri eklemesine veya güncellemesine izin verebilir. Ancak bu tür durumlarda WITH CHECK OPTION kullanımı önemlidir. Bu seçenek, görünüm tanımında belirtilen koşulların ihlal edilmesini engeller ve veri tutarlılığını korur. Görünümler, yalnızca sorgu kolaylığı sağlayan yapılar değil; aynı zamanda güvenli erişim denetimi için etkili araçlardır.

KİMLİK DOĞRULAMA

Yetkilendirme mekanizmalarının etkili olabilmesi için, kullanıcı kimliğinin güvenilir biçimde

doğrulanması gerekir. Kimlik doğrulama, veritabanı güvenliğinin temel bileşenlerinden biridir. Parola temelli kimlik doğrulama yaygın olarak kullanılsa da tek başına yeterli değildir. Zayıf parolalar, parola sızıntıları ve ağ üzerinden dinleme gibi riskler, parola temelli sistemleri savunmasız hâle getirir.

Bu nedenle iki faktörlü kimlik doğrulama ve merkezi kimlik doğrulama yaklaşımları geliştirilmiştir. İki faktörlü kimlik doğrulama, kullanıcının bildiği bir bilgi ile sahip olduğu bir doğrulama aracını birlikte kullanır. Merkezi kimlik doğrulama ve tek oturum açma sistemleri ise kullanıcıların birden fazla sistem için ayrı ayrı parola yönetmesini engelleyerek hem güvenliği hem de kullanım kolaylığını artırır.

UYGULAMA DÜZEYİNDE YETKİLENDİRME

Bazı senaryolarda tablo veya sütun düzeyinde yetkilendirme yeterli olmaz. Özellikle her kullanıcının yalnızca kendisine ait kayıtları görmesi gereken durumlarda, satır düzeyinde erişim denetimi gereklidir. Geleneksel SQL yetkilendirme mekanizmaları bu ihtiyacı doğrudan karşılamaz. Bu nedenle uygulama düzeyinde yetkilendirme veya satır düzeyinde güvenlik yaklaşımları kullanılır.

Uygulama düzeyinde yetkilendirme, erişim denetiminin uygulama kodu içinde yapılmasını ifade eder. Ancak bu yaklaşım geliştirici hatalarına açıktır. Satır düzeyinde güvenlik ise bu denetimi doğrudan veritabanı katmanına taşıyarak tüm istemciler için zorlayıcı ve tutarlı bir güvenlik sağlar.

EK GÜVENLİK TEDBİRLERİ

Veritabanı güvenliği yalnızca yetkilendirme ve kimlik doğrulamadan ibaret değildir. Şifreleme, ağ güvenliği, fiziksel güvenlik ve insan faktörü gibi ek güvenlik tedbirleri de dikkate alınmalıdır.

Şifreleme, verinin yetkisiz kişiler tarafından okunmasını engeller. Açık anahtarlı şifreleme ve dijital sertifikalar, özellikle ağ üzerinden güvenli iletişim sağlamak için kullanılır.

Fiziksel güvenlik önlemleri, donanımın korunmasını hedeflerken; insan faktörü, kullanıcıların bilinçli davranmasını gerektirir. Güvenlik ihlallerinin önemli bir kısmı teknik zafiyetlerden değil, kullanıcı hatalarından kaynaklanır. Bu nedenle güvenlik, çok katmanlı ve bütüncül bir yaklaşım gerektirir.

SQL SERVER ORTAMINDA YEDEKLEME VE GERI YÜKLEME

Bu bölümde, SQL Server ortamında yedekleme ve geri yükleme süreçleri hem kuramsal hem de uygulamalı boyutlarıyla ele alınmıştır. Veri güvenliği, modern bilgi sistemlerinin sürdürülebilirliği açısından kritik bir unsur olup, yalnızca teknik bir gereklilik değil aynı zamanda kurumsal risk yönetiminin temel bileşenlerinden biridir. Kurumların dijital varlıkları; donanım arızaları, yazılım hataları, kullanıcı kaynaklı yanlış işlemler, kötü amaçlı yazılımlar ve doğal afetler gibi çok sayıda risk faktörüyle karşı karşıyadır. Bu nedenle verinin korunması, yalnızca yedek alınmasıyla sınırlı olmayan, planlı ve test edilmiş bir geri yükleme stratejisini de kapsayan bütüncül bir yaklaşımla ele alınmalıdır. Bölüm boyunca SQL Server'ın sunduğu Full, Differential ve Transaction Log yedekleme türleri; Recovery Model yapıları; T-SQL komutları; SSMS üzerinden grafiksel işlemler; otomasyon mekanizmaları ve ileri düzey kurtarma senaryoları ayrıntılı biçimde incelenmiştir. Ayrıca RPO ve RTO kavramları çerçevesinde veri kaybı toleransı ve sistem kesinti süreleri stratejik bir perspektifle değerlendirilmiştir.

Veri Güvenliği, Recovery Model ve Yedekleme Stratejilerinin Temelleri

Veri güvenliği kavramı, yalnızca verinin saklanması değil; bütünlüğünün, erişilebilirliğinin ve sürekliliğinin korunması anlamına gelmektedir. Bu bağlamda SQL Server ortamında yedekleme stratejisinin doğru belirlenmesi, sistem mimarisinin en önemli tasarım kararlarından biridir. Recovery Model seçimi, yedekleme ve geri yükleme süreçlerinin kapsamını doğrudan etkiler. Simple Recovery Model, işlem günlüğünü otomatik olarak temizlediği için log yedeğine izin vermez ve daha basit senaryolar için uygundur. Full Recovery Model ise tüm işlemleri log dosyasında tutarak zamana bağlı kurtarma (Point-in-Time Recovery) yapılmasına imkân tanır. Bulk-Logged Recovery Model ise toplu veri işlemlerinde performans avantajı sağlarken belirli sınırlamalar içerir. Bu modellerin seçimi, kurumun veri kaybına karşı toleransı ve iş sürekliliği hedefleriyle doğrudan ilişkilidir. RPO (Recovery Point Objective), kabul edilebilir maksimum veri kaybı süresini; RTO (Recovery Time Objective) ise sistemin yeniden çalışır hâle gelmesi için kabul edilebilir maksimum süreyi ifade eder. Bu iki kavram, teknik kararların ötesinde yönetsel stratejilerin de belirleyicisidir. Etkili bir yedekleme planı, genellikle Full yedeklerin belirli aralıklarla alınmasını, aradaki değişikliklerin Differential yedeklerle desteklenmesini ve kritik sistemlerde Transaction Log yedekleriyle detaylı kurtarma imkânı sağlanmasını içerir.

SQL Server Ortamında Yedekleme Türleri ve T-SQL ile Yedekleme İşlemleri

SQL Server'da yedekleme işlemleri farklı ihtiyaçlara göre çeşitlendirilmiştir. Full yedek, veritabanının belirli bir andaki tam görüntüsünü içerir ve diğer yedek türlerinin temelini oluşturur. Differential yedek, son Full yedekten sonra değişen veri sayfalarını içerir ve geri yükleme sürecini hızlandırır. Transaction Log yedekleri ise log zincirini koruyarak daha hassas kurtarma imkânı sunar. Copy-Only yedekler mevcut yedek zincirini bozmadan alınır ve genellikle test veya geçici ihtiyaçlar için tercih edilir. Büyük ölçekli sistemlerde File ve Filegroup yedekleme teknikleri kullanılarak belirli dosya gruplarının ayrı ayrı yedeklenmesi mümkündür. T-SQL ile yedekleme işlemleri BACKUP DATABASE ve BACKUP LOG komutları aracılığıyla gerçekleştirilir. Bu komutlara eklenen WITH seçenekleri, yedekleme davranışını özelleştirir. Örneğin INIT mevcut dosyanın üzerine yazılmasını sağlarken, FORMAT medya başlığını yeniden oluşturur, COMPRESSION yedek boyutunu azaltır ve STATS ilerleme yüzdesini gösterir. Yedeklerin disk, ağ paylaşımı veya URL (bulut) ortamına alınabilmesi, veri koruma stratejilerinde esneklik sağlar. Günümüzde hibrit yedekleme mimarileri, yerel depolama ile bulut ortamının birlikte kullanılmasını önerir. Otomatik yedekleme scriptleri ve SQL Server Agent ile zamanlanmış görevler, insan hatasını minimize ederek düzenli veri koruması sağlar.

T-SQL ile Geri Yükleme Süreçleri ve İleri Düzey Kurtarma Senaryoları

Yedekleme süreci kadar önemli olan geri yükleme aşaması, sistemin gerçek anlamda güvence altına alınmasını sağlar. T-SQL ile geri yükleme işlemleri RESTORE DATABASE ve RESTORE LOG komutları ile gerçekleştirilir. Geri yükleme sırasında doğru sıralama kritik öneme sahiptir: Önce Full yedek, ardından varsa Differential yedek ve son olarak sıralı biçimde Transaction Log yedekleri uygulanmalıdır. NORECOVERY seçeneği veritabanını restore modunda bırakır ve ek yedeklerin uygulanmasına imkân tanır. RECOVERY ise işlemi tamamlayarak veritabanını kullanıma açar. STANDBY modu, veritabanını salt okunur olarak açarak ara durum analizi yapılmasına olanak verir.

WITH MOVE seçeneği, dosyaların farklı fiziksel konuma taşınmasını sağlar ve özellikle farklı sunuculara geçiş senaryolarında önemlidir. STOPAT parametresi kullanılarak belirli bir tarih ve saate kadar geri dönüş yapılabilir; bu özellik, yanlış veri silme veya hatalı işlem sonrasında sistemin güvenli noktaya döndürülmesinde hayati rol oynar. Geri yükleme işlemlerinin yalnızca teorik olarak bilinmesi yeterli değildir; düzenli testlerin yapılması, felaket kurtarma planlarının uygulanabilirliğini doğrular. Başarılı bir kurtarma stratejisi, yalnızca veri kaybını minimize etmekle kalmaz, aynı zamanda iş sürekliliğini ve kurumsal itibarı korur.

SSMS ile Grafiksel Yedekleme ve Geri Yükleme İşlemleri

SSMS, yedekleme ve geri yükleme işlemlerinin grafiksel arayüz üzerinden gerçekleştirilmesine olanak tanır. Bu yöntem, özellikle öğrenme aşamasında ve küçük ölçekli sistemlerde pratik bir çözüm sunar. Tasks menüsü üzerinden Back Up ve Restore seçenekleri kullanılarak yedekleme türü, hedef konum ve kurtarma seçenekleri belirlenebilir. Maintenance Plan sihirbazı, düzenli yedekleme planlarının oluşturulmasını sağlar. SQL Server Agent, bu planların zamanlanmış görevler olarak çalıştırılmasından sorumludur. Ancak Express sürümde SQL Server Agent bulunmadığından otomatik görevler için alternatif yöntemler kullanılmalıdır. Grafiksel arayüz, teknik ayrıntıları sadeleştirirken kullanıcı hatasını azaltır; ancak ileri düzey senaryolarda T-SQL komutlarının sağladığı esneklik daha üstündür. Bu nedenle kurumsal sistemlerde her iki yaklaşımın birlikte kullanılması önerilir.

Stratejik Değerlendirme ve Sonuç

SQL Server ortamında yedekleme ve geri yükleme süreçleri, yalnızca teknik komutların öğrenilmesinden ibaret değildir. Bu süreçler; risk analizi, iş sürekliliği planlaması ve kurumsal politika geliştirme gibi stratejik kararlarla bütünleşmelidir. Recovery Model seçimi, yedekleme sıklığı, saklama politikaları ve depolama mimarisi; kurumun veri kaybına karşı toleransını belirler. Etkili bir strateji, Full + Differential + Log kombinasyonuna dayalı, düzenli test edilmiş ve otomasyonla desteklenmiş bir yapıyı gerektirir. Yedek alınmayan veri korunamaz; test edilmeyen yedek ise güvenilir değildir. Bu nedenle veri koruma yaklaşımı, planlama, uygulama ve doğrulama aşamalarını kapsayan döngüsel bir süreç olarak ele alınmalıdır. Doğru yapılandırılmış bir SQL Server yedekleme ve geri yükleme mimarisi, veri bütünlüğünü korurken sistem sürekliliğini güvence altına alır ve dijital çağın en değerli varlığı olan bilginin sürdürülebilirliğini sağlar.