ERCİYES ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMMING LABORATORY NOTLARI

5. HAFTA NOTU (C# ile Veritabanı Programlama)

Bu not dersin 5. haftasında yapacağınız uygulamaya hazırlanmanızı sağlayacak olup temel kavramları anlatan bilgileri içermektedir.

DERSİN ÖĞRETİM ÜYESİ

Prof. Dr. Bahriye AKAY

DENEYİ YAPTIRAN ÖĞRETİM ELEMANI

Arş. Gör. Osman Buğra KAHRAMAN

2026
KAYSERİ

1. DENEYDE KULLANILACAK KÜTÜPHANELER VE FONKSİYONLAR HAKKINDA KISA BİLGİ

1.1. System.Data

Veritabanından çekilen ham verilerin, uygulama içerisinde satır ve sütun mantığıyla organize edilmesini sağlayan ADO.NET mimarisinin merkezidir. Özellikle DataTable sınıfı, SQL sorgu sonuçlarını geçici bellek alanlarında tutarak arayüzde sergilenmesine olanak tanır.

1.2. System.Windows.Forms

Kullanıcı etkileşimli pencerelerin, butonların ve veri tablolarının oluşturulmasını sağlar. Olay-temelli (event-driven) yapısı sayesinde, kullanıcının bir butona tıklaması veya bir metin kutusuna yazı yazması gibi eylemleri yakalar ve ilgili kod bloklarını tetikler.

1.3. Microsoft.Data.SqlClient

SQL Server ile C# uygulaması arasındaki veri alışverişini sağlayan temel kütüphanedir. Bu kütüphane; veritabanı bağlantısını yöneten SqlConnection, SQL komutlarını yürüten SqlCommand ve verileri bellek üzerine aktaran SqlDataAdapter gibi kritik sınıfları barındırır.

1.4. System.Reflection

Programın çalışma zamanında (runtime) kendi yapısını incelemesine veya değiştirmesine imkân tanıyan bir kütüphanedir. Projede, özellikle 500 ve üzeri satırın bulunduğu listelerde ekranın titremesini engellemek ve kaydırma işlemini akıcı hale getirmek için DoubleBuffered özelliğini aktif etmek amacıyla kullanılmıştır.

1.5. MarketIslem(IslemTipi tip, params object[] veriler)

Yazılımın yönetim merkezidir. Modüler programlama prensibi gereği; ekleme, silme, güncelleme ve arama gibi tüm işlemler bu fonksiyon üzerinden geçer. params yapısı sayesinde değişken sayıda veri girişi kabul ederek kod tekrarını minimize eder.

1.6. TabloDoldur(string sql, params object[] p)

Belirtilen SQL sorgusunu veritabanına iletir ve dönen sonuçları bir DataTable nesnesine yerleştirir. Bu fonksiyon, veritabanındaki binlerce verinin arayüzdeki tablolara (Grid) aktarılmasından sorumludur.

1.7. SorguCalistir(string sql, params object[] p)

Veritabanı üzerinde fiziksel değişiklik yapan (INSERT, UPDATE, DELETE) komutları yürütür. İşlem sonucunda veritabanından dönen "etkilenen satır sayısı" bilgisini kullanarak işlemin başarısını kontrol eder ve kullanıcıya mantıksal bir geri bildirim sağlar.

2. DENEYDE KULLANILACAK PROGRAMLAMA YAPILARI HAKKINDA KISA BİLGİ

2.1. enum (Numaralandırma Yapısı)

IslemTipi adıyla kurgulanan bu yapı, program akışında kullanılan kritik komutları (Listele, Ekle, Sil vb.) isimsel olarak gruplandırır. Karmaşık sayısal değerler yerine anlamlı kelimelerle çalışmak, kodun okunabilirliğini artırır ve yazılımcı hatalarının önüne geçer.

2.2. params Anahtar Kelimesi ve Esnek Parametre Yönetimi

Bu yapı, bir fonksiyona gönderilecek argüman sayısının önceden sabitlenmediği durumlarda kullanılır. Örneğin, 5 kolonlu bir müşteri ekleme işlemi ile tek bir ID gerektiren silme işlemi aynı fonksiyon üzerinden bu yapı sayesinde yürütülebilir. Bu, fonksiyonel ayrıştırmanın en güçlü örneklerinden biridir.

2.3. using Blokları ve Otomatik Bellek Yönetimi

C# dilinde veritabanı bağlantıları gibi dış kaynaklar bellek yönetimi açısından risklidir. using bloğu, bir nesne ile iş tamamlandığında Dispose metodunu otomatik tetikleyerek bağlantının kapatılmasını ve belleğin boşaltılmasını garanti altına alır. Bu, sistem kaynaklarının verimli kullanılmasını sağlar.

2.4. Olay-Temelli (Event-Driven) Programlama Akışı

Geleneksel "yukarıdan aşağıya" akan programlar yerine, bu projede akış kullanıcı eylemlerine bağlıdır.

- **CellClick**: Kullanıcı tabloda bir satıra tıkladığı anda, bellekteki o satır bilgisi değişkenlere atanır ve formdaki kutucuklara yansıtılır.
- **TextChanged**: Her karakter girişinde veritabanına "anlık sorgu" gönderilerek, 500 satırlık veri seti içerisinde milisaniyeler içinde arama yapılmasına imkân tanır.

2.5. Veri Validasyonu (Doğrulama) ve Güvenlik

Kullanıcının girdiği veriler (örneğin telefon numarasının rakamlardan oluşması veya e-posta formatı) veritabanına gönderilmeden önce kontrol edilir. Bu denetim, "kirli veri"nin sisteme girmesini engelleyerek veritabanı bütünlüğünü (Integrity) ve yazılımın kararlılığını korur.

3. SQL OPERATÖRLERİ VE SORGULAMA DİLLERİ HAKKINDA BİLGİ

SQL dili, kullanım amaçlarına göre farklı alt kümelere ayrılır. Projemizde veritabanı şemasının oluşturulmasından, verilerin manipüle edilmesine kadar tüm süreçlerde bu operatörler aktif rol oynar.

3.1. Veri Tanımlama Dili (DDL - Data Definition Language)

Veritabanının yapısını, tabloları ve bu tablolar arasındaki ilişkileri oluşturmak veya değiştirmek için kullanılır.

- **CREATE**: Yeni bir veritabanı, tablo veya indeks oluşturur. Projemizde Musteriler, Urunler ve Siparisler tabloları bu komut ile tanımlanmıştır.
- **ALTER**: Mevcut bir tablonun yapısında değişiklik yapar. Örneğin, tabloya yeni bir sütun eklemek veya mevcut bir sütunun veri tipini değiştirmek için kullanılır.
- **DROP**: Veritabanındaki bir nesneyi (tablo, view vb.) tamamen siler. Nesne silindiğinde içerisindeki tüm veriler de geri döndürülemez şekilde kaybolur.

3.2. Veri Kullanma Dili (DML - Data Manipulation Language)

Tablo içerisindeki kayıtlar üzerinde işlem yapmak için kullanılan operatörlerdir. C# tarafındaki SorguCalistir metodumuz bu komutları yürütür.

- **INSERT**: Tabloya yeni bir veri satırı eklemek için kullanılır. Kullanıcı formdan "Müşteri Kaydet" butonuna bastığında bu operatör tetiklenir.
- **UPDATE**: Tablodaki mevcut verileri günceller. WHERE koşulu ile birlikte kullanılarak sadece belirli bir ID'ye sahip kaydın değiştirilmesi sağlanır.
- **DELETE**: Tablodan belirli kayıtları siler. Projemizde silme onayı alındıktan sonra ilgili kaydı veritabanından çıkarmak için kullanılır.

3.3. Veri Sorgulama Dili (DQL - Data Query Language)

Veritabanındaki verileri belirli kriterlere göre süzmek ve raporlamak için kullanılan en geniş kapsamlı bölümdür.

- **SELECT**: Hangi sütunların getirileceğini belirler. Projemizde SELECT * (tüm sütunlar) veya arama işlemlerinde belirli alanları seçmek için kullanılır.
- **FROM**: Verinin hangi tablodan veya tablolardan çekileceğini tanımlar.
- **WHERE**: Verileri belirli şartlara göre filtreler. Arama kutusuna yazılan metne göre LIKE operatörü ile birleştirilerek kullanılır.
- **GROUP BY**: Verileri belirli bir sütuna göre gruplandırır. Örneğin; her kategoride kaç ürün olduğunu saymak için kullanılır.
- **HAVING**: Gruplandırılmış veriler

üzerinde filtreleme yapar. WHERE satır bazlı çalışırken, HAVING grup bazlı çalışır (Örn: Toplam siparişi 10'dan fazla olan gruplar).

3.4. İlişkisel Operatörler (JOIN Türleri)

Birden fazla tabloyu ortak kolonlar üzerinden birleştirerek tek bir sonuç kümesi elde etmemizi sağlar.

- **INNER JOIN**: Her iki tabloda da karşılığı olan kayıtları getirir. (Örn: Sadece siparişi olan müşterileri ve sipariş detaylarını listeler).
- **LEFT JOIN**: Sol taraftaki tablonun tüm kayıtlarını getirir; sağ tarafta karşılığı yoksa NULL değer basar. (Örn: Siparişi olsun ya da olmasın tüm müşterileri listelemek için kullanılır).
- **RIGHT JOIN**: Sağ taraftaki tablonun tüm kayıtlarını öncelikli tutar.

3.5. Veritabanı Kısıtlamaları (Constraints)

Veritabanı bütünlüğünü (Data Integrity) sağlamak için kullanılan bu kurallar, hatalı veri girişini sistem seviyesinde engeller:

- **PRIMARY KEY (Birincil Anahtar):** Tablodaki her bir satırı benzersiz (unique) kılan sütundur. Hiçbir zaman NULL (boş) olamaz ve aynı değerden iki tane bulunamaz. Projemizde her müşterinin ve ürünün benzersiz bir **ID** numarası alması bu kısıtlama ile sağlanmıştır.
- **FOREIGN KEY (Yabancı Anahtar):** İki tabloyu birbirine bağlayan sütundur. Bir tablodaki verinin, başka bir tablodaki Primary Key ile eşleşmesini zorunlu kılar. Projemizde Siparisler tablosundaki MusteriID, Musteriler tablosuna bağlı bir yabancı anahtardır; böylece sistemde kayıtlı olmayan bir müşteriye sipariş açılması engellenir.
- **NOT NULL:** Bir sütunun boş geçilemeyeceğini garanti eder. Örneğin, bir müşterinin adı veya bir ürünün fiyatı olmadan kaydedilmesi mantıksal bir hatadır. Bu alanlar NOT NULL olarak işaretlenerek zorunlu hale getirilir.
- **IDENTITY(1,1):** SQL Server'a özel olan bu yapı, Primary Key alanının sistem tarafından otomatik olarak (1'den başlayıp 1'er artarak) atanmasını sağlar. Yazılımcının manuel ID vermesine gerek kalmaz.

3.6. Kullanılan Temel Veri Türleri

C# ve SQL tarafında verilerin bellekte kaplayacağı alan bu türler ile belirlenir:

- **INT (Integer):** Tam sayı değerleri saklamak için kullanılır. ID'ler, stok miktarları ve adet bilgileri bu türde tutulur.
- **NVARCHAR(n):** Değişken uzunluklu karakter dizilerini (string) saklar. "N" ön eki, Unicode desteği (Türkçe karakterler vb.) sağlar. Müşteri isimleri ve adresler için tercih

edilir.

- **DECIMAL(18,2):** Hassas ondalıklı sayılar için kullanılır. Para birimi (Fiyat, Tutar) işlemlerinde yuvarlama hatası yapmaması nedeniyle standarttır.
- **DATETIME:** Tarih ve saat bilgisini bir arada tutar. Kayıtların ne zaman oluşturulduğunu izlemek (Audit) amacıyla kullanılır.
- **BIT:** Mantıksal (Boolean) değerleri temsil eder. Sadece 0 (Yanlış) veya 1 (Doğru) değerini alır.

3.7. Bellek ve Yapısal Farkındalık

Yazılım geliştirme sürecinde bu türlerin seçimi sistem performansını etkiler:

- **Bellek Alanı:** Her veri türü bellekte farklı boyutta yer kaplar; örneğin bir char türü tek bir karakter saklarken , bir int sayısal hesaplamalar için optimize edilmiştir.
- **Diziler ve Tablolar:** Bellekte ardışık alanlarda saklanan diziler , veritabanı seviyesinde tablolara dönüştürülerek satır ve sütun mantığıyla organize edilir.

4. DERS KATKILARI

Bu uygulama kapsamında kullanılan yapılar, öğrencinin aşağıdaki teknik kazanımları edinmesini sağlar:

4.1. Algoritmik Düşünme ve Mantıksal Kurgu

Durum Geçişlerinin Modellenmesi: Bir kaydın eklenmesi, güncellenmesi veya silinmesi sırasında veritabanı ile arayüz arasındaki veri akışının ve durum değişikliklerinin nasıl yönetileceği kavranır.

Koşullu Kontrol Mekanizmaları: Kullanıcıdan alınan verilerin (E-posta formatı, telefon uzunluğu vb.) doğrulanması süreçlerinde karmaşık mantıksal denetimlerin yapılandırılması öğrenilir.

Olay-Temelli Akış: Programın sadece kullanıcı etkileşimlerine (buton tıklaması, hücre seçimi) yanıt verecek şekilde asenkron bir yapıda kurgulanması yetisi kazanılır.

4.2. Bellek Yönetimi ve Veri Yapıları Bilinci

İlişkisel Veri Yapıları: SQL tablolarındaki satır ve sütun mantığının, uygulama tarafındaki DataTable ve DataGridView gibi iki boyutlu yapılarla bellekteki temsili ve işlenmesi anlaşılır.

Kapsam Yönetimi: Proje genelinde kullanılan global değişkenler ile fonksiyonlara özel yerel değişkenlerin farkı ve veri güvenliği açısından önemi kavranır.

Veri Bütünlüğü: İşlem sırasında durum bilgisinin (ID takibi, seçili satır verisi) nasıl korunacağı ve tutarlı kalacağı deneyimlenir.

4.3. Modüler Programlama ve Kod Sürdürülebilirliği

Fonksiyonel Ayrıştırma: Veritabanı işlemlerinin arayüz kodlarından bağımsız bir sınıfta (Veritabani.cs) toplanmasıyla kodun modüler hale getirilmesi sağlanır.

Parametre Kullanımı: params ve enum yapıları aracılığıyla fonksiyonlara esneklik kazandırma ve tip güvenli kod yazma becerisi gelişir.

Yeniden Kullanılabilirlik: Yazılan merkezi fonksiyonların projenin farklı sekmelerinde (Müşteriler, Ürünler) tekrar tekrar kullanılarak kod kalabalığının önlenmesi hedeflenir.

4.4. Zaman, Performans ve Girdi-Çıktı Kontrolü

Gerçek Zamanlı Ölçüm: Sistem zamanı ile çalışarak sipariş tarihlerinin kaydedilmesi ve veritabanı üzerinden süre bazlı filtreleme algoritmaları geliştirilir.

Anlık Veri İşleme: Kullanıcı metin kutusuna yazı yazdığı anda (TextChanged) tetiklenen anlık klavye okuma ve arama teknikleri uygulanır.

Formatlı Çıktı Üretimi: Verilerin kullanıcıya düzenli tablolar ve mesaj kutuları aracılığıyla profesyonel bir formatta sunulması sağlanır.

4.5. Yazılım Mühendisliği Perspektifi

Temiz ve Okunabilir Kod: Anlamlı isimlendirmeler ve düzenli kod blokları ile ekip çalışmasına uygun, okunabilir yazılım geliştirme disiplini edinilir.

Sabitlerin Yönetimi: Bağlantı dizeleri ve işlem tiplerinin merkezi yapılarda tanımlanmasıyla, projenin bakımının ve güncellenmesinin kolaylaştırılması amaçlanır.

Mantıksal Soyutlama: Veritabanı karmaşıklığının arayüzden gizlenmesi (Abstraction) yoluyla katmanlı mimari mantığı kavranır.