T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YÜKSEK LİSANS TEZİ

Ersin Artan

VERİ VE VERİ ERİŞİM KATMANI ÜRETİCİSİ

İstanbul, 2009

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YÜKSEK LİSANS TEZİ

Ersin Artan
060820010

VERİ VE VERİ ERİŞİM KATMANI ÜRETİCİSİ

Tez Danışmanı
Prof. Dr. M. Yahya Karslıgil

İstanbul, 2009

ÖNSÖZ

Günümüzde telekomünikasyon sistemleri, sağlık bilgi sistemleri, otomotiv sanayisi, şirket yönetimi, bankacılık, lojistik ve eğitim gibi alanlarda faaliyet gösteren kurum ve kuruluşlar bünyelerindeki bilgileri depolamak için veritabanı kullanmaktadır. Söz konusu olan veritabanının mimari yapısı, kurulum, derleme ve güvenlik ön planda tutularak geleceğe yönelik tasarlanmalıdır. Veritabanına esnek, kolay, hızlı ve güvenli erişimi sağlamak var olan bilginin verimli ve etkin kullanılmasına yardımcı olacaktır. Hazırlanan projede bu önemli hususlar dikkate alınarak veritabanının oluşturulması ve erişim ağının kurulması hedeflenmiştir.

Bu tez çalışmasının; gerçekleştirilme aşamasında benden yardımlarını esirgemeyen ve her türlü konuda bana yol gösteren değerli hocam Prof. Dr. M. Yahya KARSLIGİL ile çalışmalarımda bana güç veren aileme teşekkür ederim.

2009, **Ersin ARTAN**

İÇİNDEKİLER

Sayfa

ÖZET

Günümüzde hazırlanan gelişmiş uygulamaların ihtiyaçlarını karşılamak için genellikle çok katmanlı mimari yapı kullanılır. Bu mimari yapıda Veri Erişim Katmanı, Veri Katmanı üzerinde bulunur ve uygulama ile veritabanı arasındaki iletişimi sağlar. Veri Erişim Katmanında Veritabanı erişim yöntemleri tanımlanır, gerekiyorsa giriş/çıkış arabirimleri ile haberleşme protokolleri de oluşturulur.
Tez kapsamında geliştirilen uygulamada, veritabanı kullanılan sistemlerde veri erişiminin yazılım tarafından oluşturulması ile uyguluma geliştirme işleminde oluşabilecek hataların engellenmesi, kod tekrarının ve zaman kaybının önüne geçilmesi hedeflenmiştir.

Bu tez çalışmasında Veri Katmanı ve Veri Erişim Katmanı üretecek uygulama Visual Studio.NET platformunda, VB.NET programlama dili kullanılarak geliştirilmiştir.
Veri Katmanı yapısı içerisinde veritabanının oluşturulması, uygulamada bulunan "SQL Oluştur" bölümünde SQL programlama diline uygun olarak saklı yordamlar üretilerek sağlanmıştır. Veri Erişim Katmanının oluşturulması "Sınıf Oluştur" bölümünde sınıf kodlarının üretilmesi ile gerçekleştirilmiştir.

Ersin ARTAN

ABSTRACT

While today's advanced applications are developed, N-Tier Architectural structures are usually used in order to supply requirements. In that architectural structure, Data Access Layer exists on Data Layer and provides the communication between the application and database. Database access methods are defined in Data Access Layer; if it's necessary, I/O interfaces and communication protocols are written in here too. In application developed as part of the thesis, it's aimed to prevent from the probable mistakes, code repeat and waste of time by the data access being formed by software.

In this thesis, the application that will produce Data Layer and Data Access Layer was developed on Visual Studio.NET platform by using VB.NET programming language. Database in Data Layer was provided by producing Stored Procedures as being appropriate for SQL programming language, in SQL Develop stage. Data Access Layer was developed in Class Develop stage by producing Class Codes.

Ersin ARTAN

KISALTMA LİSTESİ

CLR	Common Language Runtime (Ortak Dil Çalışma Platformu)
CORBA	Common Object Request Broker Architecture (Ortak Nesne İstem Aracısı Mimarisi)
DCOM	Distributed Component Object Model (Ortak Nesne İstem Aracısı Mimarisi)
DHTML	Dynamic Hyper Text Markup Language (Dinamik Hareketli-Metin İşaretleme Dili)
GUI	Graphical User Interface (Grafiksel Kullanıcı Arayüzü)
HTML	Hyper Text Markup Language (Hareketli Metin İşaretleme Dili)
JIT	Just In Time (Tam Zamanında)
MSIL	Microsoft Intermediate Language (Microsoft Ara Dili)
RDBMS	Relatianol Database Management System (İlişkisel Veritabanı Yönetim Sistemi)
RMI	Remote Method Invocation (Uzaktan Metot Çağırımı)
SOAP	Simple Object Access Protocol (Basit Nesne Erişim Protokolü)
SQL	Structured Query Language (Yapılandırılmış Sorgulama Dili)
VB	Visual Basic
WML	Wireless Markup Language (Kablosuz Biçimleme Dili)
XML	eXtensible Markup Language (Genişletilebilir Biçimleme Dili)

ŞEKİL LİSTESİ

TABLO LİSTESİ

1. GİRİŞ

Günümüz sistemleri için hazırlanan gelişmiş uygulamaların birçok bileşenden oluşması, yapının karmaşık hale gelmesine ve sistem bütünlüğünün zor algılanmasına neden olur. Karmaşık yapıda bulunan bu uygulamalar operasyonel açıdan bakım, güvenlik, esneklik ve tekrar kullanılabilirlik gibi temel gereksinimlerin karşılanmasında bazı sıkıntıları beraberinde getirir. Bu tarz sıkıntıları aşmak için geliştirilen uygulamalarda kullanılan bileşenlerin katmanlara ayrılması iyi bir çözüm olarak karşımıza çıkmaktadır. Katmanlı yapıya sahip bir uygulama geliştirmek için genellikle çok katmanlı mimari yapı tercih edilir. Çok katmanlı mimari yapı seçimi, temel işlemlerin farklı katmanlar üzerinden yapılması ile yazılım geliştirmede çok esnek ve tekrar kullanılabilir yapılar sunmaktadır. Bu mimari yapının temelini, kullanılacak veritabanının yapıldığı veri katmanı ve veritabanına erişimi sağlayacak veri erişim katmanı oluşturmaktadır. Bu çalışmada veri ve veri erişim katmanlarını üretecek genel bir uygulama geliştirilmiştir. Geliştirilen uygulama kullanılıp belirli bir sisteme dayandırılarak, bilinen ve tekrarlanılarak kullanılan kod yapılarının istenilen özellikler belirlenerek üretilmesi sağlanmıştır. Bu sayede zaman kaybının engellenmesine, farklı kişiler tarafından hazırlanan kodların rahat anlaşılmasına, gramer hatalarının en aza indirilmesine olanak sunulmaktadır. Hazırlanan uygulama, birden fazla sistem, aygıt ve kullanıcı gibi unsurların bir arada kullanılmasını sağlayan Visual Studio.NET platformunda, VB.NET programlama dili ile aynı kodların bir kütüphane içerisinde toplanmasını sağlayan Sınıf yapıları hazırlanarak geliştirilmiştir. Veritabanının oluşturulması için çoğu kurum ve kuruluşun istediği başarıma sahip ve bu başarıma ait ek maliyet gerektirmeyen bir veri yönetim çözümü olması sebebiyle tercih edilen SQL yapısı kullanılmıştır.

Bölüm 2'de; Tez çalışmasında hazırlanan uygulamanın amacı, tezin aşamaları ve yapılan analiz çalışmaları bulunmaktadır.

Bölüm 3’de; Hazırlanan uygulamanın sistem yapısı, modelleri ve sınıf yapıları bulunmaktadır.

Bölüm 4’de; Tez konusu ile ilgili genel bilgiler ve tez hazırlanırken kullanılan yazılımlar hakkında bilgi verilmektedir.

Bölüm 5’de; Hazırlanan uygulamanın tanıtımı yapılarak ve kullanımı ile ilgili bilgiler sunulmaktadır. Uygulama kullanılarak yapılan örnek bir proje tasarımının anlatımı yapılmıştır.

Bölüm 6’da; Çok katmanlı mimari yapı baz alınarak oluşturulacak sistemlerin geliştirilmesinde kullanılmak için hazırlanan uygulamadan elde edilen sonuçlar belirtilmekte ve uygulamanın değerlendirilmesi yapılmaktadır.

2. SİSTEM ANALİZİ

Bu bölümde tez çalışmasında hazırlanan uygulamanın amacı ve geliştirilmesi sırasında yapılan analiz çalışmaları aşağıdaki başlıklarda yer almaktadır.

2.1 Tezin Amacı

Birçok bileşenin kullanılması ile meydana gelen gelişmiş uygulamaların karmaşık yapıdan kurtarılıp bileşenlerin katmanlara ayrılarak ele alınması, yazılım geliştirmede ihtiyaç duyulan esneklik, tekrar kullanılabilirlik, güvenlik ve kolay bakım gibi temel gereksinimlerin karşılanmasını sağlar. Günümüzde bu unsurlar göz önünde bulundurularak uygulama geliştirilirken genellikle çok katmanlı mimari yapı tercih edilir. Bu çalışmada çok katmanlı mimarinin veri yapısını meydana getiren veri ve veri erişim katmanlarının oluşturulması için gereken kodların üretilmesini sağlayan genel bir yazılım geliştirilmesi amaçlanmıştır.

2.2 Tezin Sağladığı Yararlar

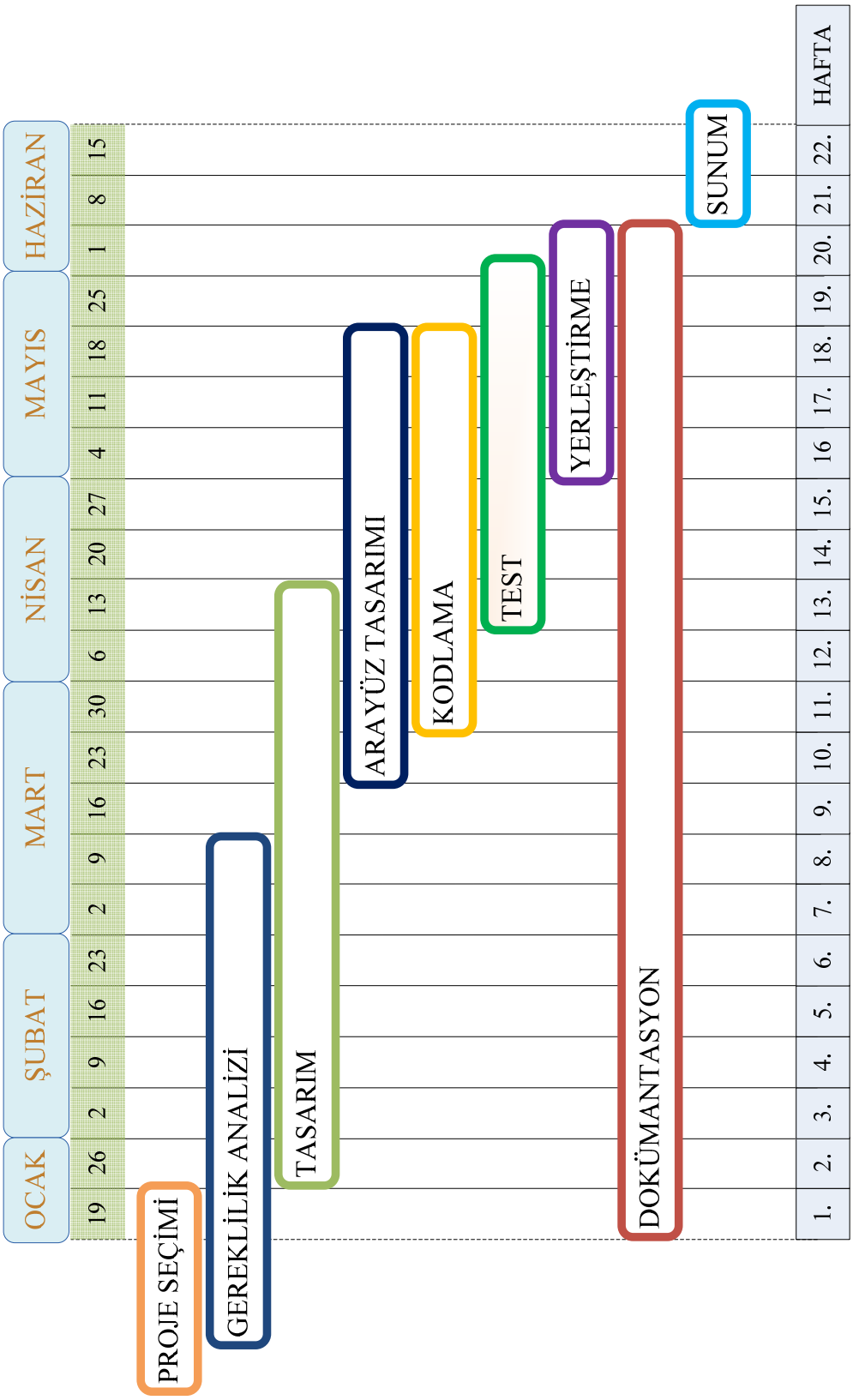
Tez çalışmasında geliştirilen uygulamanın sağlayacağı yararlar aşağıda madde madde belirtilmiştir:

- Veritabanı uygulamalarında yapılan bağlantıların ve veri erişiminin yazılım tarafından oluşturulması ile olası hataların engellenmesi
- Yazılım uygulamaları geliştirilirken yapılan çalışmaların, belirli bir sisteme dayandırılması
- Oluşturulacak programların yapısında bulunan alt programların, modüllerin ve objelerin belirli standartlarda üretilmesi
- Tekrarlanılarak kullanılan kod yapılarının istenilen özellikler doğrultusunda üretilmesi ile yazılım oluşturulurken meydana gelebilecek hataların önlenmesi ve zaman kaybının önüne geçilmesi

- Geliştirilen yazılımın kullanılması ile belli bir yapıda üretilen kodlar, yazılım projelerinde görev alan tüm kişiler tarafından rahatlıkla anlaşılabilir hale getirilmesi
- Olası eleman değişimlerinde, ayrılanın ürettikleri diğer çalışanlar tarafından rahat anlaşıldığından zaman kaybının önüne geçilmesi
- Üretimin herhangi bir aşamasında projeye katılan kişilerin projeye uyum sağlama sürecinin hızlanması
- Kolay geliştirilmeye imkan sağlanması
- Yapılan uygulamalarda değişiklik veya geliştirilme istenmesi durumunda, tüm yazılımın yeniden oluşturulması yerine, lüzum görülen bölümün düzenlenmesinin sağlanması
- Veri katmanının saklı yordamlar aracılığıyla oluşturulması sayesinde kullanıcı yetkilendirmelerine olanak sağlanması

2.3 İş – Zaman Akışı

Bu bölümde tez çalışmasının yapılışındaki aşamalar yazılım mühendisliği kurallarına uygun şekilde belirlenerek işlenmiştir. Bu konuda [1] ‘de belirtilen kaynaktan yararlanılmıştır. Aşağıda verilen Şekil 2.1’de İş – Zaman Akışı tüm aşamalar belirtilmiştir. Ayrıca bu aşamalarda yapılan işlemler sırasıyla açıklanmıştır.



Şekil 2.1 İş – Zaman Akışı

- **Proje Seçimi**

Bu aşamada tez çalışması belirlenirken, bilgisayar alanında bulunan herkesin kullanabileceği ve kullanıcılara fayda sağlayabilecek bir seçim yapılması düşünüldü. Bu nedenle bu konu ile ilgilenen bazı kişilerin fikirleri alınarak, ihtiyaçları tespit edildi. Bu doğrultuda geliştirilen yazılımlarda kullanılan veritabanının oluşturulması ve bu veritabanına erişimin sağlanması sırasında tekrarlanarak yazılan kodların büyük zaman ve emek kaybına neden olduğu, ayrıca geliştirilen uygulamalarda birçok bileşenin kullanılması ile yapının karmaşık hale geldiği gözlendi. Bu nedenle bu işlemlerin sistematik bir yapı ile oluşturulmasına olanak sunacak ve hazırlanması planlanan kodları üretecek bir uygulama geliştirilmesi kararına varıldı.

- **Gereklilik Analizi**

Sistemin ihtiyaç duyacağı yazılım ve donanım gereksinimleri, gereklilik analizi sonucunda ortaya konmuştur. Sistemin oluşturulmasında gerekli bulunan faktörler belirlenerek sistemden verimli sonuç alınması hedeflenmiştir.

- **Yazılım Gereksinimi**

Nesneye yönelik programlama ile geliştirilecek uygulamada, yazılım geliştirme aracı olarak, birden fazla sistem, aygıt ve kullanıcı gibi unsurların bir arada kullanılmasını sağlayan Visual Studio.NET platformunun kullanılması uygun görülmüştür. Programlama dili olarak VB.NET seçilmiştir. Veritabanının oluşturulması için çoğu kurum ve kuruluşun istediği başarıma sahip ve bu başarıma ait ek maliyet gerektirmeyen bir veri yönetim çözümü olması sebebiyle tercih ettiği SQL yapısı kullanılmıştır.

  - **İşletim Sistemi Gereksinimleri**

İşletim sistemi olarak Windows XP Professional, Windows Server 2003 veya Windows Vista tercih edilebilinir.

- **Donanım Gereksinimleri**

Geliştirilen uygulamayı koşabilecek donanım gereksinimleri birlikte çalıştığı sistemler olan .NET Platformu ve SQL Sunucusunun gereksinimleri göz önünde bulundurularak değerlendirilmelidir. Bu doğrultuda sistemin çalışması için gereken minimum sistem gereksinimleri Tablo 2.1' de verilmiştir. [2]

Tablo 2.1 Minimum Donanım Gereksinimi

İşlemci	2.0 GHz
Bellek	1 GB
Hafıza	8 GB

Tavsiye edilen sistem yukarıdaki tabloda belirtilen donanımdan daha güncel özelliklere sahip olmakla birlikte, sistemin çalışma başarımı bilgisayarın üzerinde koşan diğer programlara göre farklılık gösterebilir.

- **Tasarım**

Geliştirilmesi düşünülen yazılımın ihtiyaçları göz önünde bulundurularak, çalışmada oluşturulacak sistemin genel yapısı, sistemin modelleri, üretilecek olan sınıf yapıları ve kullanılacak metotların belirlenmesi bu aşamada yapılmıştır. Tasarım aşamasında oluşturulan sistemin bilgisi detaylı olarak üçüncü bölümde açıklanmıştır.

- **Arayüz Tasarımı**

Geliştirilerek kullanıma sunulacak yazılımın arayüz tasarımları bu aşamada belirlenmiştir. Arayüz tasarımı oluşturulurken kullanan kişiler tarafından basit anlaşılması, kolay kullanılması, ihtiyaçları karşılayacak nitelikte olması, hata ve uyarıların doğru şekilde verilmesi ve güvenilir olması göz önünde bulundurulmuştur. Günümüzde sıkça karşımıza çıktığı ismi ile kullanıcı dostu (user friendly) bir yazılım arayüzü tasarımı yapılmaya çalışılmıştır.

- **Kodlama**

Bu aşamada hazırlanan tasarım programlama diline dönüştürülerek yürütülebilir kod ve dinamik kütüphane şeklinde yazılım birimleri elde edilmiştir. Kullanılan programlama dili ve kodlama biçimi, yapılan çalışmalar sonucunda eksik noktalarında kapatılması ile geliştirilen uygulamanın niteliğini ve bakım özelliğini en üst düzeye taşıyacak şekilde hazırlanmıştır.

- **Test**

Yazılım geliştirme sürecinin çeşitli aşamalarında oluşan hataların belirlenmesi ve giderilmesi için gerekli testler yapılmıştır. Böylece doğru sonuçların elde edilerek sağlam bir ürün geliştirilmesi sağlanmıştır.
Arayüz tasarımı, kodlama ve test aşamaları birbirine paralel olarak sürdürülmüştür.

- **Yerleştirme**

Geliştirilen yazılımın kullanılacak sisteme entegre edilerek kurulması ve bu sisteme uyumlu olarak çalışmasının sağlandığı aşamadır.

- **Dokümantasyon**

Hazırlanan yazılım geliştirilirken kullanılan alt programların, modüllerin ve objelerin kullanım amaçları belirtilerek, kod bloklarına açıklama bölümleri eklenmiştir. Tez çalışmasının başından sonuna kadar takip edilen aşamaların, faydalanılan kaynakların ve bilgilerin düzenlenerek toplanması sağlanmıştır. Bu bilgiler ışığında tez kitapçığı düzenlenmiştir.

- **Sunum**

Bu aşamada tez çalışmasında geliştirilen yazılımın sunumunun hazırlanması ve yapılması planlanmıştır.

2.4 Risk Analizi

Bu çalışmada geliştirilen uygulama, yazılım geliştirilirken belirli bir sisteme dayalı projeler oluşturulmasına kolaylık sağlar. Bu bağlamda, uygulamayı kullanacak olan kişilerin yazılım alanında bilgili oldukları göz önünde bulundurularak risk analizi yapılmalıdır.

Yazılım ile oluşturulan projeler, XML data üzerinde kaydedilerek devam ettirilir. Kişi yaptığı çalışmayı XML data aracılığı ile diğer kişilerle paylaşabilir. Projenin devamı esnasında XML datanın bozulması söz konusu olabilir. Yine Yazılım arayüzü ile yapılan her değişiklik, sistemin çalışmamasına neden olabilir. Bu gibi durumlarda Proje bazında kayıpların yaşanmaması için XML datanın yedeklenmesi işlemi büyük önem teşkil etmektedir.

Yazılım kullanımı aynı anda tek kullanıcıya izin verdiği için, sistem üzerinde kullanıcı sayısından kaynaklanan bir yoğunluk yaşanmaz.

Uygulama ile programların düzenli bir yapıda geliştirilmesine olanak sunularak, çalışan kişilerin işten ayrılması ya da projeyi bırakması gibi durumlarda zaman kaybının en aza indirgenmesi sağlanmıştır.

SQL Server ve .NET Platformunun yapılarında değişiklik yapacak güncellemeler, yazılımda üretilen kodların entegrasyonunda bazı sorunlara neden olabilir. Bu nedenle güncelleme yapıldıktan sonra yazılım tarafında da güncelleme yapılıp projelerin değiştirilmesi gereklidir.

Yazılımın geliştirilmesi sürecinde, yazılım ile üretilen kodların SQL Server ve .NET Platformlarımı üzerinde çalışır olduğu proje bazlı yapılacak testler ile kontrol edilmelidir.

3. SİSTEMİN GENEL YAPISI, SİSTEM MODELLERİ VE SINIF YAPILARI

Bu bölümde üretilen sistemin genel yapısı, sistemi tanımlanmasını sağlayan bazı modeller ve kullanılan sınıf yapıları anlatılmıştır.

3.1 Sistem Yapısı

Sistem yapısındaki temel özellikler aşağıdaki gibidir.

3.1.1 Sistem Girdileri

Modelleyici kullanılarak oluşturulması beklenen veritabanı yapısını oluşturacak sisteme girilen temel bileşenler:

- Tablo İsimleri
- Tablo Tipleri
- Kolon İsimleri
- Kolon Nitelikleri

3.1.2 Sistemde Yapılan İşlemler

Kullanıcı isteğine bağlı olarak geliştirelecek veritabanı yapısının oluşturulması, veri katmanının şekillenmesi ve buna bağlı olarak veri erişim katmanının meydana getirilmesini sağlayan modelleme işlemleridir.

3.1.3 Sistem Çıktıları

Uygulama tarafından oluşturulan sistem çıktıları aşağıdaki gibidir.

- SQL saklı yordamları
- .NET Sınıf Kodları

Şekil 3.1' de sisteme yapılan girdiler, geliştirilen uygulamada yapılan işlemler ve bu işlemlerden sonra sistemden alınan çıktılar bulunmaktadır.

Şekil 3.1 Veri ve Veri Erişim Katmanı Üretim Sistemi

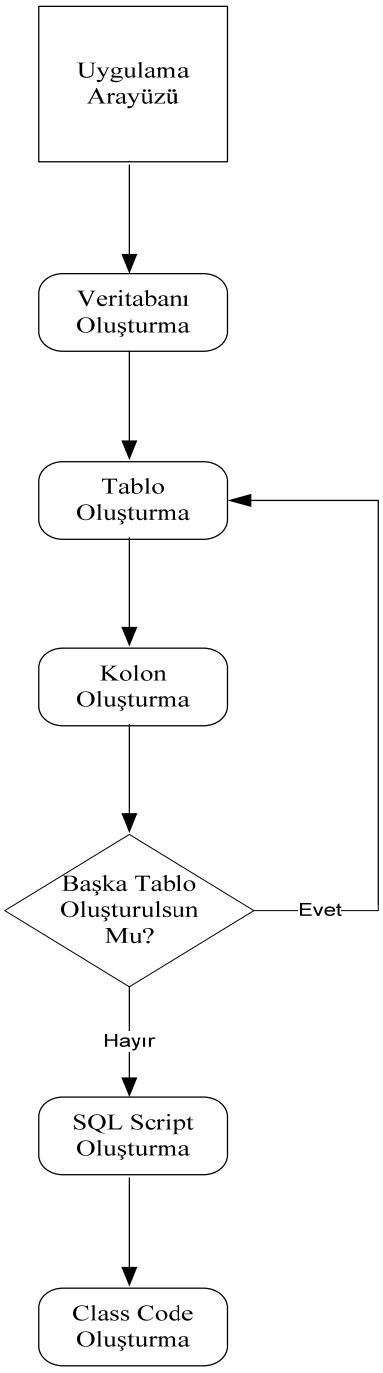
3.2 Sistem Modelleri

Sistemin tanımlanması, işleyişini ve durumlarını izah etmek ve göstermek amacıyla bazı şematik modellerden faydalanılmıştır. Bu bölümde kullanılan aşağıdaki modellere yer verilmiştir.

- Veri Akış Modeli
- Sınıf Diyagramı
- Sıralı Gidiş Diyagramı
- Varlık İlişki Diyagramı

3.2.1 Veri Akış (Data Flow) Modeli

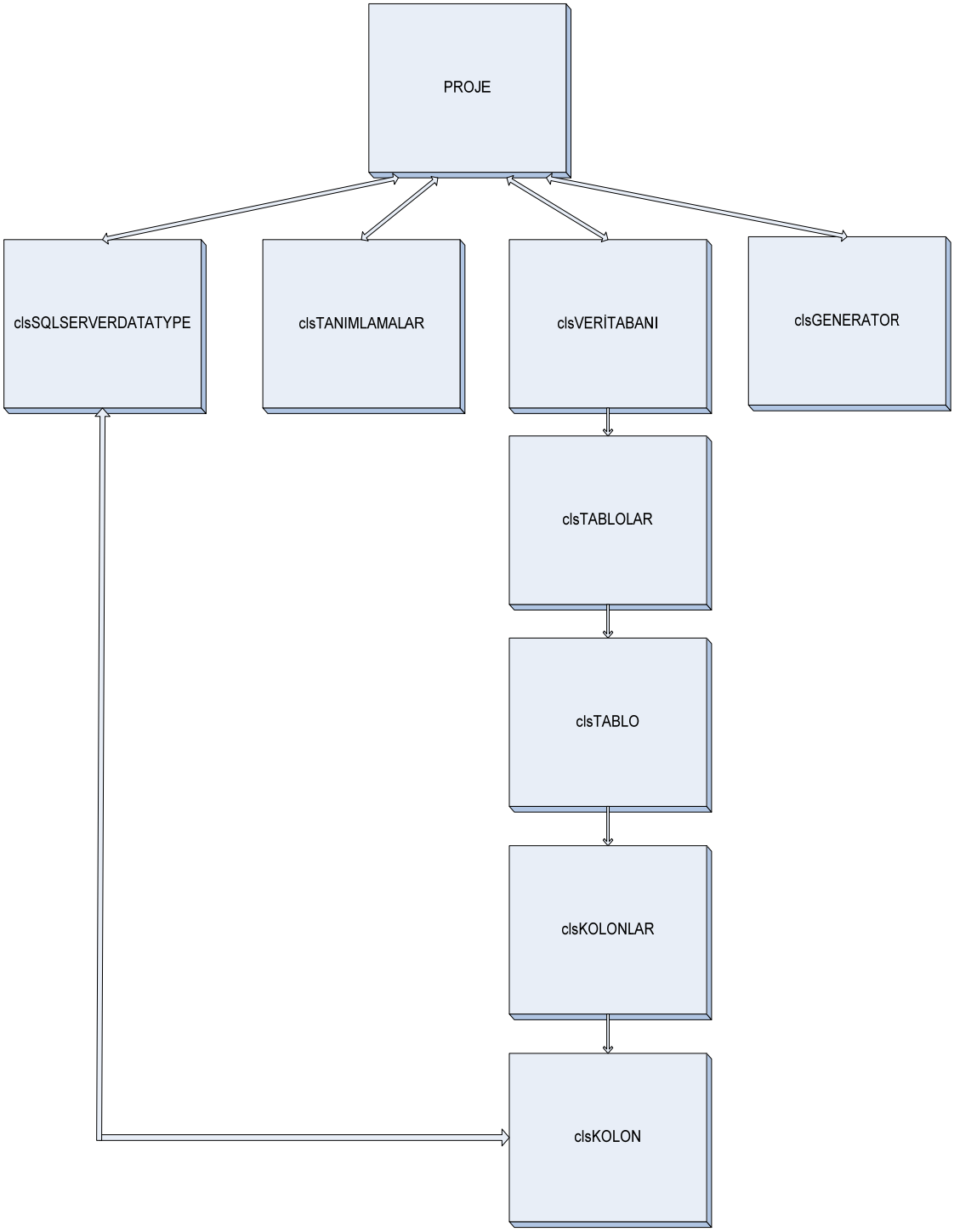
Sistemin veri akış modeline ait diyagram Şekil 3.2' de verilmiştir.



Şekil 3.2 Veri Akış Modeli

3.2.2 Sınıf Diyagramı

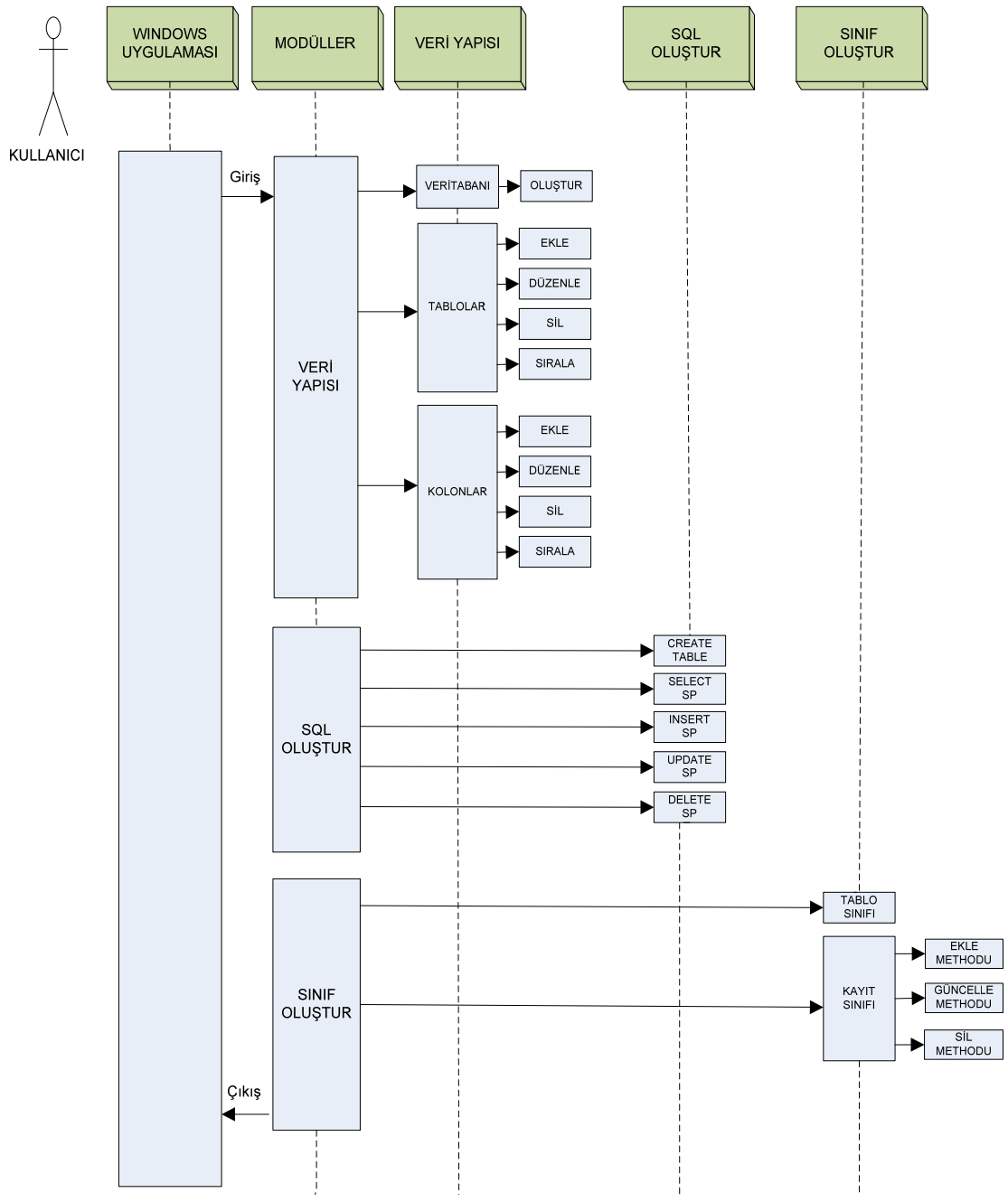
Bu uygulamada bir ana sınıf yapısı ve buna bağlı olan dört alt sınıf bulunmaktadır. Hiyerarşik yapı aşağıda bulunan Şekil 3.3' deki gibidir.



Şekil 3.3 Sınıf Diyagramı

3.2.3 Sıralı Gidiş (Sequence) Diyagramı

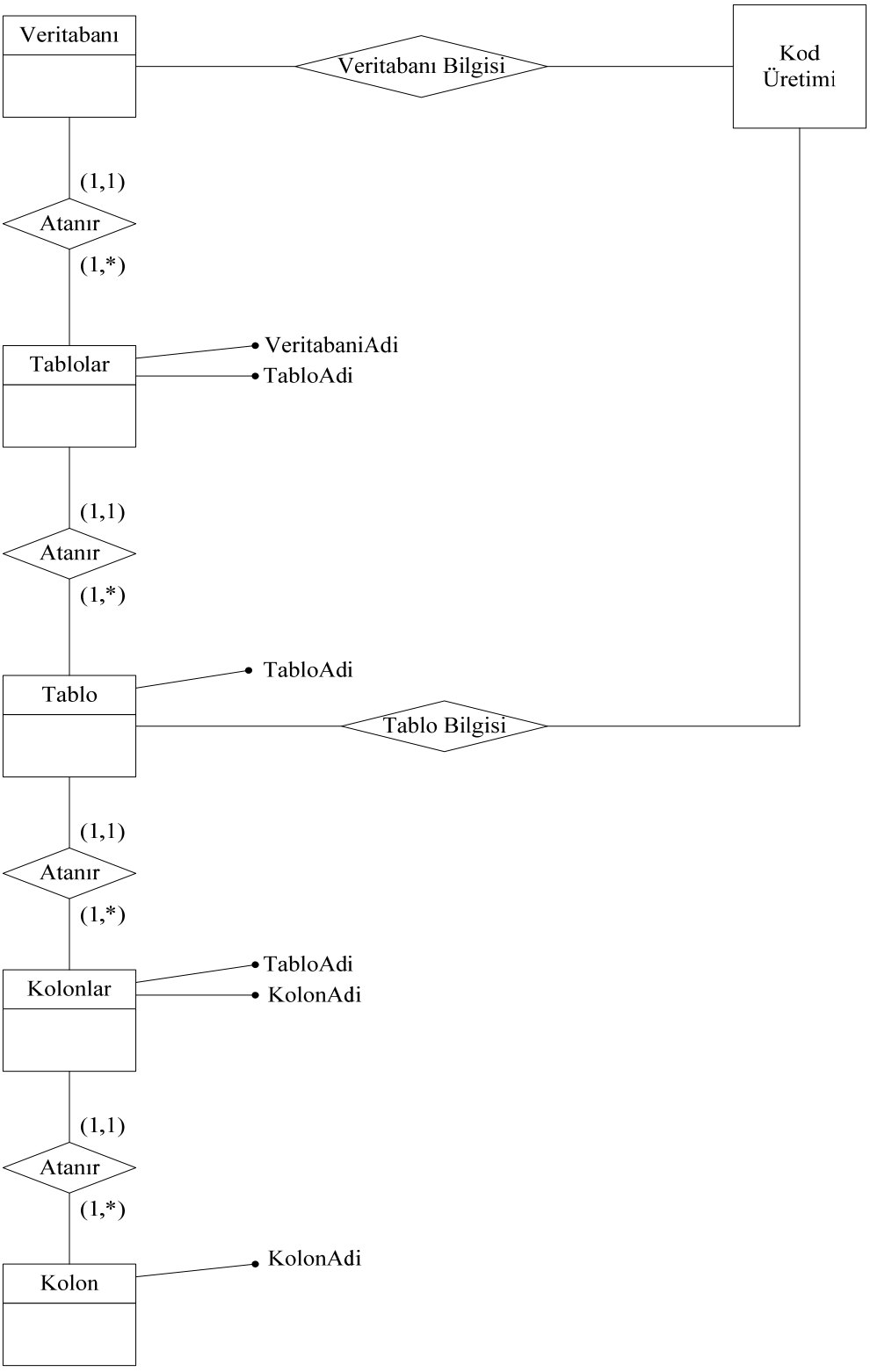
Sistem kullanıcısının işlemlerini sırasıyla belirten sıralı gidiş diyagramı Şekil 3.4' de verildiği gibidir.



Şekil 3.4 Kullanıcı Sıralı Gidiş Diyagramı

3.2.4 Varlık İlişki (Entity Relationship) Diyagramı

Sistemin varlık ilişki diyagramı Şekil 3.5' de verilmiştir.



Şekil 3.5 Varlık İlişki Diyagramı

3.3 Sınıf Yapıları

Sınıf veriler ve bu verilerin üzerinde yapılacak işlemleri tutan birimdir. Yani veri ve kodların saklandığı konteyner (container) olarak açıklanabilinir.
Sınıf yapısını oluşturan ana bölümler aşağıda belirtilmiştir. [3]

- **Yapıcı (Constructor) :** Bir sınıf *new* komutu ile yaratıldığında kod bloğunun işletildiği bölümdür.
- **Yıkıcı (Destructor) :** Bir sınıf yapısı bellekten atılırken kod bloğunun işletildiği bölümdür.
- **Alanlar (Fields) :** Sınıf yapısı içerisinde kullanılacak olan ana verileri tutan, gizli ve korunan değişkenlerdir. İstemci kodu tarafından özellikler, indeksler ve metotlar ile erişim sağlanır.
- **Özellikler (Properties):** Sınıf içindeki verilere erişmek için özellikler (properties) kullanılır. Bunlar da nesne dışındaki kodlar tarafından değiştirilebilen sınıf verileridir.
- **İndeks Oluşturucu (Indexer) :** Nesnelerin dizi indis numarasıyla kullanılabilmesini sağlayan yapıdır.
- **Metotlar (Methods) :** Bir nesne tarafından yerine getirilen herhangi bir faaliyete ise metot denir. Metotlar sınıf yapısı içerisinde tanımlaması yapılarak kullanılan alt programlar ve fonksiyonlar olarak açıklanabilir. [4]

Sınıf yapısı herhangi bir nesnenin program içerisindeki kalıbı olarak düşünülebilir. Bu kalıp nesnenin tüm özelliklerini tutup, program içerisinde değişik yerlerde kullanımını sağlar. Sınıf yapıları nesne yönelimli yazılım dillerinin kalbini oluşturmakla beraber, yazılımcılar için değişik avantajları vardır.

- Yazılan bir class aynı program içerisinde veya başka programlarda birçok kez kullanılabilir.
- Yazılım ve test süreçlerini kısaltır.
- Kod organizasyonunu sağlayıp hataların bulunmasını kolaylaştırır.
- Kod üzerindeki yapılacak olan değişikliklerin kolaylaşmasını sağlar.

3.3.1 Uygulamada Kullanılan Sınıflar ve Yapıları

Uygulama geliştirilirken hiyerarşik bir düzende birbirine bağlı olarak çalışan beş sınıf, ayrıca SQL veri tiplerinin tutulduğu bir sınıf oluşturulmuştur. Bunlar:

3.3.1.1 VeriTabanı Sınıfı

Yeni bir veritabanı oluşturmak için bu sınıf düzenlenmiştir. Sınıf yapısı;

- Verilerin tutulduğu "Alanlar",
- Verilere kod ile dışarıdan erişimi sağlayacak "Özellikler",
- Yeni oluşumu sağlayan kod bloğunun tutulduğu "Yapıcı",
- Kullanılacak fonksiyonların tanımlandığı "Metot" bölümlerinden oluşur.

Aşağıda "Veritabanı Sınıf" yapısının kod olarak görünümü verilmiştir.

```
"Fields"

    strVeriTabaniAdi As String
    strVeriTabaniEtiketi As String
    objTablolar As Tablolar
    strHata As String

"Properties"

    VeriTabaniAdi() As String
    VeriTabaniEtiketi() As String
    TabloKoleksiyonu() As Tablolar
    Tablo(tabloIndeksi As Integer) As Tablo
    Hata() As String

"Constructor"

    New(veriTabaniEtiketi As String)

"Methods"

    Temizle()
    XML()
    Nesne Adı Belirle()
    SQL Oluştur()
    Class Oluştur()
```

3.3.1.2 Tablolar Sınıfı

Veritabanı oluşturulduktan sonra hiyerarşik yapıda veritabanına tablo eklenmesi işlemi yapılır. Birden fazla tablo eklenmesi olası olduğu için bu durumda eklenen tablolar, "Tablolar Sınıfı" içerisinde tutulur. Yapısında "Alanlar", "Özellikler" ve "Metotlar" bulunur.

3.3.1.3 Tablo Sınıfı

Veritabanına eklenen her tablonun ayrı ayrı detaylarının tutulduğu sınıftır. Yapısında "Alanlar", "Özellikler" ve "Metotlar" bulunur.
Veritabanına eklenen iki çeşit tablo tipi vardır. Bunlar:

- **Veri Tablosu :** İçerisinde verilerin tutulduğu tablo tipidir.
- **İlişki Tablosu :** İçerisinde diğer tablolarda bulunan verilerin birbirleriyle aralarında bulunan ilişkilerin tutulduğu tablo tipidir.

Tablonun tipinin belirlenmesi için "Tablo Sınıfı" içerisinde bir değişken oluşturulmuştur. Bu değişken hem "Alanlar", hem de "Özellikler" bölümünde belirtilmiştir. Aşağıda bu değişkenin iki bölümde de kod olarak tanımlaması gösterilmiştir.

```
"Fields"
   boolIliskiTablosu As Boolean

"Properties"
   IliskiTablosu() As Boolean
```

3.3.1.4 Kolonlar Sınıfı

Tablo oluşturulduktan sonra, tablo içeriğini oluşturmak için kolon ekleme işlemi yapılır. Birden fazla kolon ekleme olası olduğu için bu durumda eklenen kolonlar, "Kolonlar Sınıfı" içerisinde tutulur. Yapısında "Alanlar", "Özellikler" ve "Metotlar" bulunur.

3.3.1.5 Kolon Sınıfı

Tabloya eklenen kolonun detaylarının tutulduğu sınıftır. Bu detaylar içerisinde öncelikle kolonda kullanılacak veri tipi belirlenir. Daha sonra bu verinin; salt okunur, kullanıcı tarafından değer girilebilir, birincil anahtar, gerekli, tekrarlanmayan özelliklerinin hangisine ya da hangilerine sahip olacağı değişkenler aracılığı ile belirtilir. Bu değişkenlerin tutulduğu "Alanlar" bölümünün kod görünümü aşağıda verilmiştir.

```
"Fields"

        strKolonAdi As String
        strKolonEtiketi As String
        objVeriTipi As SqlServerDataTypes
        boolSaltOkunur As Boolean
    boolKTDG As Boolean
        boolBirincilAnahtar As Boolean
        boolGerekli As Boolean
        boolTekrarlanmayan As Boolean
        boolOtomatikSayi As Boolean
        intOtomatikSayiBaslangici As Integer
        intOtomatikSayiArtisi As Integer
        strVarsayilanDeger As String
        strDegerKontrolu As String
        strDegerKontroluMesaji As String
        boolIliskili As Boolean
        boolOtomatikGuncellestir As Boolean
        intIOTI As Integer
        intIOKI As Integer
        strHata As String
```

3.3.1.6 SQL Veri Tipleri Sınıfı

Veritabanı oluşturulurken kullanılacak SQL veri tipleri bu sınıf içerisinde tanımlanmıştır. Aşağıda bu veri tiplerinin "Özellikler" bölümü altında tanımlanmasını sağlayan kod bloğu bulunmaktadır.

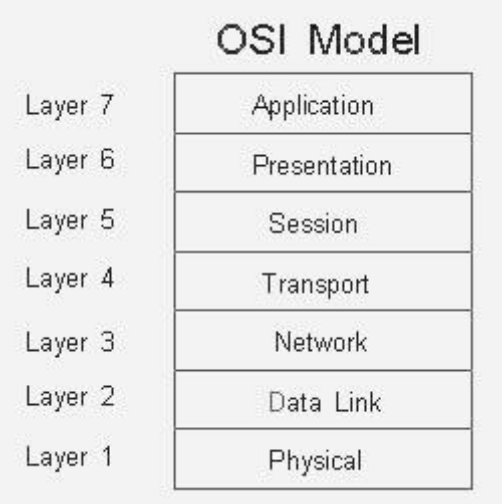
```
SqlServerDataType As Integer
    [BigInt]
    [Binary]
    [Bit]
    [Char]
    [DateTime]
    [Decimal]
    [Float]
    [Image]
    [Int]
    [Money]
    [NChar]
    [NText]
    [Numeric]
    [NVarChar]
    [Real]
    [SmallDateTime]
    [SmallInt]
    [SmallMoney]
    [Text]
    [TimeStamp]
    [TinyInt]
    [VarBinary]
    [VarChar]
```

4. GENEL VE KAYNAK BİLGİLERİN İNCELENMESİ

Bu bölümde tez konusu ile ilgili genel bilgiler verilmekte ve kullanılan yazılımlar kısaca anlatılmaktadır.

4.1 Çok Katmanlı Mimari (N-Tier Architecture)

Veritabanı uygulamalarının geliştirilmesinde genellikle çok katmanlı mimari kullanılır. Bu mimari türü, bazı temel işlemlerin farklı katmanlar üzerinden yapılmasını sağlar. Çok katmanlı mimari yapıda geliştirilen ugulamalar yazılım geliştiricilere çok esnek ve tekrar kullanılabilir yapılar sunmaktadırlar. Uygulamaları katmanlara bölerek geliştirmek, varolan teknolojinin değiştirilmesi ya da geliştirilmek istenmesi durumunda, sadece gereken katmanın düzenlenmesi ya da eklenmesi ile uygulamanın tekrar baştan geliştirilmesini engeller. Çok katmanlı anlamına gelen "N-tier" içerisinde kullanılan "N" ifadesinin bir standardı yoktur ve tasarıma göre şekillenir. Uygulama mimarileri OSI modelinin 7. katmanının parçalarından biridir. Ayrıca OSI modeli katmanlı yapıya verilebilecek en güzel örneklerden biridir. Aşağıda bulunan Şekil 4.1' de bu modelin katmanları görülmektedir.[5]



Şekil 4.1 OSI Modeli

1. Physical Layer (Fiziksel Katman)
2. Data Link Layer (Veri Bağlantı Katmanı)
3. Network Layer (Ağ Katmanı)
4. Transport Layer (Taşıma Katmanı)
5. Session Layer (Oturum Katmanı)

6. Presentation Layer (Sunu Katmanı)
7. Application Layer (Uygulama Katmanı)

Aşağıda verilen Tablo 4.1' de ise tipik bir çok katmanlı mimari gözükmektedir. [6]

Tablo 4.1 Çok katmanlı mimari yapı örneği

<table>
<tr><td>SUNUM
GUI</td><td colspan="3">Son Kullanıcı Sistemi
(HTML, Windows Forms, vb.)
Fiziksel olarak kullanıcı makinesinde</td></tr>
<tr><td>Sunum Katmanı</td><td>Web
Sunucu Tarafındaki IIS

(VBScript, Jscript, Web Forms, C#, VB.NET, vb.)

Üretilen: HTML, XML, DHTML, WML, vb.</td><td>Dağınık Mantık
İstek gönderip alabilmesi için sunucu tarafındaki Proxy Katmanına bağlanması gereklidir.

Proxy Katmanı
(SOAP, CORBA, RMI, DCOM, vb.)</td><td>Kullanıcı Arayüzü
(Windows tabanlı formlar, kullanıcının görüntüleyebildiği herhangi bir uygulama)</td></tr>
<tr><td>İş Katmanı</td><td colspan="3">İş Nesneleri ve Kuralları
Verinin Manipülasyonu ve Bilgiye Dönüşümü</td></tr>
<tr><td>Veri Erişim Katmanı</td><td colspan="3">Veri katmanı erişimi
Giriş/Çıkış arabirimleri ile haberleşme</td></tr>
<tr><td>Veri Katmanı</td><td colspan="3">Verinin en optimize bir şekilde saklanması, sorgulama yapılırken performans sağlanması</td></tr>
</table>

- **Veri Katmanı:** Verinin yer açısından en uygun şekilde saklanması, sorgulama yapılırken en hızlı sonuçların elde edilmesi gibi etkenler düşünülür. Bu katman bir veritabanı olarak düşünülürse MSSQL veya Oracle olabilir.

- **Veri Erişim Katmanı:** Veri katmanına erişimi sağlayan katmandır. Veritabanı erişim yöntemlerinin tanımlı olduğu, gerekiyorsa giriş/çıkış arabirimleri ile haberleşme protokollerinin yazılı olduğu katmandır. Veri erişim katmanı tablolar, saklı yordamlar ve diğer bileşenler ile sağlam bir ilişki kurarak veri mantığını sağlar. Veritabanına esnek, kolay, hızlı ve güvenli erişim sağlamanın yanı sıra diğer katmanlardan bağımsız olarak veritabanı yapılarını mantıksal bir biçimde soyutlar. Örneğin bir tablo satırında yapılan değişiklikler, uygulamanın diğer kısımlarını etkilemez. Diğer katmanlar isteklerini veri katmanına gönderirler. Bu istekleri alan veri katmanı, isteğe göre tablolara kayıt ekleme, silme, kayıt güncelleştirme, saklı yordamlar ile kayıt seçme ve dataset geri döndürme gibi işlemleri yapacaktır. Projelerde böyle bir veri erişim katmanın bulunması, veritabanının başarımını, hızını, güvenliğini ve kod okunabilirliğini arttırır.

- **İş Katmanı :** İşin gerçekleştirildiği katmandır. İçerisinde veri katmanına erişimi sağlayan kurallar veya veri katmanını modelleyen sınıflar bulunabilir.

- **Sunum Katmanı:** Uygulamanın gösterime girmesinden önceki tanımlama katmanıdır. İşin hem istemci(client) hem sunucu(server) tarafında nasıl gerçekleneceği ile ilgili tanımlamaların yapıldığı katmandır.

Bu mimari içerisindeki katmanlar birbirlerinden bağımsız olmakla beraber, istenirse birbirleriyle haberleşebilirler. Uygulama ihtiyaçlarına göre bu katmanların kendi alt katmanları da olabilir.

4.2 .NET Platformu

Birden fazla sistem, aygıt ve kullanıcı gibi unsurların bir arada kullanılmasını sağlayan sistemdir.
İki ana bileşenden oluşur. Bunlar:

- Ortak Dil Çalışma Platformu (Common Language Runtime)
- Sınıf Kütüphanesi (Class Library)

Uygulama derlendiği zaman Microsoft Ara Dili (Microsoft Intermediate Language (MSIL)) olarak bilinen diller (Visual Basic.NET, Visual C#.NET, Visual C++.NET, Visual J#.NET) Ortak Dil Çalışma Platformuna çevrilir. Yani dil yapısı ortak bir platform ortamına taşınır. Daha sonra Tam Zamanında (Just in Time (JIT)) olarak tanımlanan derleyici ile makine diline çevrilir. [4]

4.2.1 Bilinmesi Gereken Kavramlar

.NET Framework çalışmasının iyi anlaşılması için aşağıdaki kavramların bilinmesi gerekir.

4.2.1.1 Sınıf (Class)

Yaygın olarak programlarda kullanılan özellik ve metotların tanımlandığı program kütüphaneleridir. Birçok programda aynı kodlar kullanılacaksa class içerisinde tanımlanıp, kütüphane oluşturularak zamandan kazanç sağlanır.
Nesneye dayalı programlama modeli, birbirinin özellik ve metotlarını kullanabilen, gerektiğinde diğerine ait kodları alıp yeni kodlar eklenebilen sınıflardan oluşur. Her şey isimlendirilmiş sınıfların içerisinde kullanıldığı için, hangi metodun nerede tanımlandığı programcı tarafından kolayca takip edilebilir.
Nesneye Dayalı Programlamanın temelinde, kitapların raflardaki düzenli dizilişleri veya öğrencilerin hangi sınıfın içerisinde hangi sırada oturduğu önem arzetmektedir. Bu sayede, içerisindeki bilgisine başvurulan kitap zaman kaybetmeden bulanabilir, veya

ödevi yapacak öğrenci sınıfında rahatça yakalanabilir. Bu yöntem son dönemde yaygın olarak kullanılan tüm diller imkan verdiği ölçüde kullanılmaktadır.

Bu tür pogramlama tekniklerinde amaç, işi yapan metodu bir yerde tanımlamak, şayet diğer sınıflar da bu işi yaptıracaksa, içerisinde ikinci kez aynı kodları yazmak yerine önceki sınıfa başvurarak o metodu kullanmak istediğini belirtmek olmalıdır. Bu sayede aynı kodun ikinci kez yazılması sonucu oluşabilecek hatalar engellenebileceği gibi, yapılacak program çok daha az yer işgal edecektir. Nesneye Dayalı Programlama yöntemlerinin en çok kullandığı programlar "Dll" uzantılı uygulamalardır. Bu dosya programda referans gösterilerek istenilen metot çağrılabilinir.[7]

4.2.1.2 İsim Alanı (Namespace)

Birbiri ile ilişkili sınıf yapıları belirtilmek için isim alanı kullanılır. Ana sınıftan türetilen birçok sınıf ile bir isim alanı yapısı oluşturulabilinir. İsim alanı için kısaca sınıf ailesi denilebilir. [4]

4.2.1.3 Ortak Dil Çalışma Platformu

.NET Framework içerisinde, kodların koşulduğu ve geliştirme işlemlerinin daha kolay yapılmasına olanak sağlayan servislerin bulunduğu Ortak Dil Çalışma Platformu (CLR) geliştirilmiştir. [8]

Ortak Dil Çalışma Platformu, .NET Framework içerisinde .NET dilleri ile yazılmış kodların yönetimi ve çalıştırılması görevlerini üstlenmiş bileşenidir. Ortak Dil Çalışma Platformu nesneleri aktive eder, nesneler üzerinde güvenlik denetimleri gerçekleştirir, belleğe aktarır, çalıştırır ve çöp-toplama(garbage-collection) işlemini gerçekleştirir. [7]

4.3 SQL

İlişkisel veri tabanı yönetim sistemleri (RDBS) modeli ilk olarak 1970 yılında Dr.E.F. Codd tarafından tarif edilmiştir. SQL veya Structured English Query Language (SEQUEL), IBM firması tarafından Codd'un modeli kullanılmak için geliştirilmiştir. SEQUEL sonraları SQL olmuştur. 1979 yılında, Relational Software, SQL'in ilk ticari uygulamasını geliştirmiştir. Bugün SQL, ilişkisel veri tabanı yönetim sistemleri standardı olarak kabul edilmektedir.

SQL, ilişkisel veri tabanlarındaki bilgileri sorgulamak için kullanılan bir dildir. SQL, tüm kullanıcıların ve uygulamaların veri tabanına erişmek için kullandıkları komutlar bütünüdür. Bilgi alışverişi için kullanılan işaretlere veri denir. Veri sesli, görsel veya yazılı bilgi şeklinde olabilir. Belli bir tarzda düzenlenmiş ve birbirleriyle ilişkili olan bu veri topluluğuna veri tabanı denir. Veri tabanında çok sayıda veri güvenilir bir şekilde saklanır ve üzerinde çeşitli sorgulama işlemleri yapılabilir. Veri, veritabanı içinde tablo adı verilen dosyalarda muhafaza edilir. SQL komutları ile veri sorgulama, bir tabloya kayıt ekleme, değiştirme ve silme, veri tabanı nesneleri yaratma, veri tabanına ve nesnelerine erişimi kontrol etme ve veri tabanı bütünlüğünü ve tutarlılığını sağlama işlemleri yapılabilmektedir. [9]

SQL dilini kullanan ve en çok bilinen veri tabanları aşağıdaki gibidir.

- ORACLE
- SYBASE
- MICROSOFT SQL SERVER
- MICROSOFT ACCESS
- INGRES

4.3.1 En çok kullanılan SQL komutları

“SELECT”, “INSERT”, “UPDATE”, “CREATE”, “DELETE” ve “DROP” dur. Bu komutlar ile neredeyse bir veritabanının ihtiyaç duyabileceği herşey yapılabilmektedir.

4.3.2 SQL Komutundaki Koşullar

- = Eşittir
- > Büyüktür
- < Küçüktür
- >= Büyük ve eşittir
- <= Küçük ve eşittir
- <> Eşit değildir

[10]

4.3.3 Kullanılan Veri Tipleri

Veri tipleri tablolarda tutulacak olan verilerin yapısını belirlemeye yarar. Tabloları oluştururken doğru veri tiplerini kullanmak (hazırlanan veri tabanını kullanırken) kaynak ve başarım açısından büyük kazançlar sağlar. Aksi takdirde hazırlanan veri tabanları gereğinden büyük boyutta veriler tutulmasına sebep olacağı için hazırlanan veri tabanını kullanan programların başarımını da olumsuz yönde etkileyecektir.

SQL Server’ daki veri tipleri yedi temel kategori altında toplanabilir. Bunlar;

- Kesin Sayısal (Exact Numeric) Veri Tipleri
- Yaklaşık Sayısal (Approximate Numeric) Veri Tipleri
- Parasal (Monetary) Veri Tipleri
- Tarih ve Zaman (Date and time) Veri Tipleri
- Karakter (Character) Veri Tipleri
- İkili (Binary) Veri Tipleri
- Özel Amaçlı (Special Purpose) Veri Tipleri

4.3.3.1 Kesin Sayısal (Exact Numeric) Veri Tipleri

Ondalıklı veya ondalıksız kesin sayıları içerir. Bu tip veriler herhangi bir kısıtlama veya istisnai durum olmaksızın istenilen matematiksel işlemde kullanılabilinir. Kapasiteleri işlemciden bağımsız olarak sabittir. Örneğin, int veri tipinin Intel veya AMD işlemcilerde kapladığı alan 4 byte' tır. Dolayısıyla, veri tabanı bilgisayarlar arasında rahatlıkla taşınabilinir.

- **Bigint (8 byte):** Pozitif ve negatif tam sayıları tutmaya yarar. -2^{63} ile $2^{63}-1$ arasındaki tam sayıları tutabilir.
- **Int (4 byte):** Pozitif ve negatif tam sayıları tutmaya yarar. -2^{31} ile $2^{31}-1$ arasındaki tam sayıları tutabilir.
- **Smallint (2 byte):** Pozitif ve negatif tam sayıları tutmaya yarar. -2^{15} (-32768) ile $2^{15}-1$ (32767) arasındaki tam sayıları tutabilir.
- **Tinyint (1 byte):** Sadece 0 ile 255 arasındaki tamsayıları tutabilir.
- **Decimal (x,y):** Boyutu 5 ile 17 byte arasında değişir. Int ve türevi olan bigint, smallint ve tinyint veri tiplerinden farklı olarak ondalıklı sayıları tutmaya yarar. İki parametre alır. İlk parametre (x) sayının kaç karakter uzunluğunda olacağını, ikinci parametre (y) ise bu karakterlerin kaçının ondalık kısım olduğunu ifade eder. Örneğin decimal(4,2) şeklinde tanımlanan bir değişken 12,34 sayısı gibi 4 karakterden oluşan ve ilk iki karakterin tam sayı değerini, son iki karakterin ise ondalık kısmı ifade ettiği sayıları temsil eder.
- **Numeric (x,y):** Boyutu 5 ile 17 byte arasında değişir. Kullanımı decimal ile aynıdır.

Yukarıdaki veri tiplerinden en çok kullanılan int ve decimal' dir. Kullanılacak sayılar 32767' yi geçmeyecekse int yerine smallint kullanılması doğrudur veya 255' ten daha az olacaksa tinyint kullanılmalıdır. Bu durum harddiskte daha az alan kullanılmasını ve daha da önemlisi hazırlanan veri tabanını kullanan programların başarımının daha yüksek olmasını sağlar. Çünkü yapılan sorgularda veri tabanından küçük veriler çekilecektir.

4.3.3.2 Yaklaşık Sayısal (Approximate Numeric) Veri Tipleri

Bu veri tipleri ondalıklı sayıları tutabilir. Decimal veri tipinin tutamadığı sayıları saklamak için kullanılırlar. Ama yaklaşık sayısal veri tipini kullanmanın bazı dezavantajları vardır. Kaydedilen sayının ondalık kısmının kayıpsız bir şekilde saklanacağı garantisi yoktur. Örneğin 1.2345 sayısı float(4) şeklinde tanımlandığında, sistem sayının son 2 karakterini atıp 1.23 olarak tutacaktır. Yani veri kaybı olacaktır. Bu nedenle bu sayılara yaklaşık (approximate) sayısal denir. Ayrıca yaklaşık sayısal veri tipindeki veriler AMD ve Intel işlemcili bilgisayarlar arasında taşındığında çeşitli hatalara sebep olacaktır.

İki tane yaklaşık sayısal veri tipi vardır:

- **Float(x):** Boyutu 4 ile 8 byte arasındadır. x değeri maksimum 53 olabilir.
- **Real:** Boyutu 4 byte tır. SQL-92 standartlarını sağlamak için yerini float almıştır. Ancak halen real tipi de geçerli bir veri tipidir.

4.3.3.3 Parasal (Monetary) Veri Tipleri

Parasal verileri tutmaya yarar. 2 tane parasal veri tipi vardır:

- **Money(8 byte):** -922 337 203 685 477,5808 ile 922 337 203 685 477,5807 arası parasal veri tutabilir.
- **Smallmoney(4 byte):** -214 748,3648 ile 214 748,3647 arası parasal veri tutabilir.

4.3.3.4 Tarih ve Zaman (Date and time) Veri Tipleri

Tarih ve zamanı tutarlar. Sadece tarih veya sadece zaman tutan bir veri tipi henüz mevcut değildir. İki tane tarih ve zaman veri tipi vardır:

- **Datetime (8 byte):** 1 Ocak 1753 ile 31 Aralık 9999 tarihleri arasındaki herhangi bir tarihi saatiyle birlikte tutar. 3.33 milisaniyelik bir hassasiyete sahiptir.
- **Smalldatetime (4 byte):** 1 Ocak 1990 ile 6 Haziran 2079 tarihleri arasındaki herhangi bir tarihi saatiyle birlikte tutar. 10 dakikalık bir hassasiyete sahiptir.

4.3.3.5 Karakter (Character) Veri Tipleri

ANSI standartlarına uyan herhangi bir karakter 1 byte yer kaplar. Ancak Unicode karakterleri hafızada tutabilmek için 1 byte yeterli değildir ve ekstradan 1 byte daha kullanılır ve toplam 2 byte veri tutarlar. Peki, bu Unicode karakterler nedir? Unicode karakterler farklı dillere özgü karakterlerdir. Örneğin Türkçe' de kullanılan "ç,ğ,ş,ö" gibi harfler Türkçe' ye özgüdür ve ANSI standartlarında yer almamaktadır. Bu karakterlerin veri kaybı olmadan saklanabilmesi için Unicode olarak tanımlanması gerekir. 8 tane karakter veri tipi vardır ve Unicode olanlar başında "n" harfi olanlardır. Bunlar:

- **Char(n):** Boyutu1 ile 8000 arasında değişir. Maksimum 8000 karakter tutar.
- **Nchar(n):** Boyutu 2 ile 8000 arasında değişir. Maksimum 4000 karakter tutar.
- **Varchar(n):** Boyutu1 ile 8000 arasında değişir. Maksimum 8000 karakter tutar.
- **Varchar(MAX):** Maksimum 2 gigabyte (1.073.741.824 karakter) veri saklar.
- **Nvarchar(n):** Boyutu 2 ile 8000 arasında değişir. Maksimum 4000 karakter tutar.
- **Nvarchar(MAX):** Maksimum 2 gigabyte (536.870.912 karakter) veri saklar.
- **Text:** Maksimum 2 gigabyte (1.073.741.824 karakter) veri saklar.
- **Ntext:** Maksimum 2 gigabyte (536.870.912 karakter) veri saklar.

Karaker veri tiplerini kısaca özetlemek gerekirse temelde Char ve text olmak üzere 2 tane veri tipi vardır. Bu iki tip n (unicode) ve var (variable) öneklerini alırlar ve diğer veri tipleri oluşur. "n" öneki daha önce de belirtildiği gibi özel karakterler olan Unicode karakterleri temsil ederler. Bu karakterlerin her biri 2 byte yer kaplar. "Var" öneki ise boyutun değişken olduğu anlamına gelir. Mesela "örnekveri" kelimesi nchar(20) şeklinde tanımlandığında 40 byte yer tutarken nvarchar(20) şeklinde tanımlanırsa 9*2=18 byte yer tutar. "n" parametresi ise maksimum karakter uzunluğunu belirtir.

Varchar(MAX)' taki MAX parametresi 1.073.741.824 sayısını, nvarchar(MAX)' taki MAX parametresi ise 536.870.912 sayısını temsil eder. Bu iki veri tipi text ve ntext' e alternatif olarak SQL Server 2005' te ortaya çıkmıştır. Bunların text ve ntext' ten daha iyi olan yönü boyutlarının tuttukları veriye göre değişmesidir. Oysaki text ve ntext' te içerik ne olursa olsun boyut 2 gigabyte' tır. Ayrıca text ve ntext' te iki verinin eşitliğini sorgulama veya birden fazla veriyi birleştirme gibi özellikler yoktur.

4.3.3.6 İkili (Binary) Veri Tipleri

SQL Server' da dosyaları saklamak için ikili veri tipleri kullanılır. 4 çeşit ikili veri tipi vardır. Bunlar:

- **Binary(n):** Boyutu 1 ile 8000 byte arasında değişir. Sabit boyutta ikili veri tutmak için kullanılır.
- **Varbinary(n):** Boyutu 1 ile 8000 byte arasında değişir. Değişken boyutta ikili veri tutmak için kullanılır.
- **Varbinary(MAX):** Boyutu maksimum 2 gigabyte' tır. Değişken boyutta ikili veri tutmak için kullanılır.
- **Image:** Boyutu maksimum 2 gigabyte' tır.

Küçük boyuttaki dosyalar için binary ve varbinary veri tipleri kullanılır.
Varbinary(MAX) SQL Server 2005' te ortaya çıkmıştır.
Image veri tipinin özelliklerine ilave olarak varbinary tipinin özelliklerini de taşır.
Image veri tipi sadece resim dosyaları için değildir, bütün dosyaları kaydetmede kullanılabilir. Tek şart dosya boyutunun 2 gigabyte' tan daha az olmasıdır. [11]

4.3.3.7 Özel Amaçlı (Special Purpose) Veri Tipleri

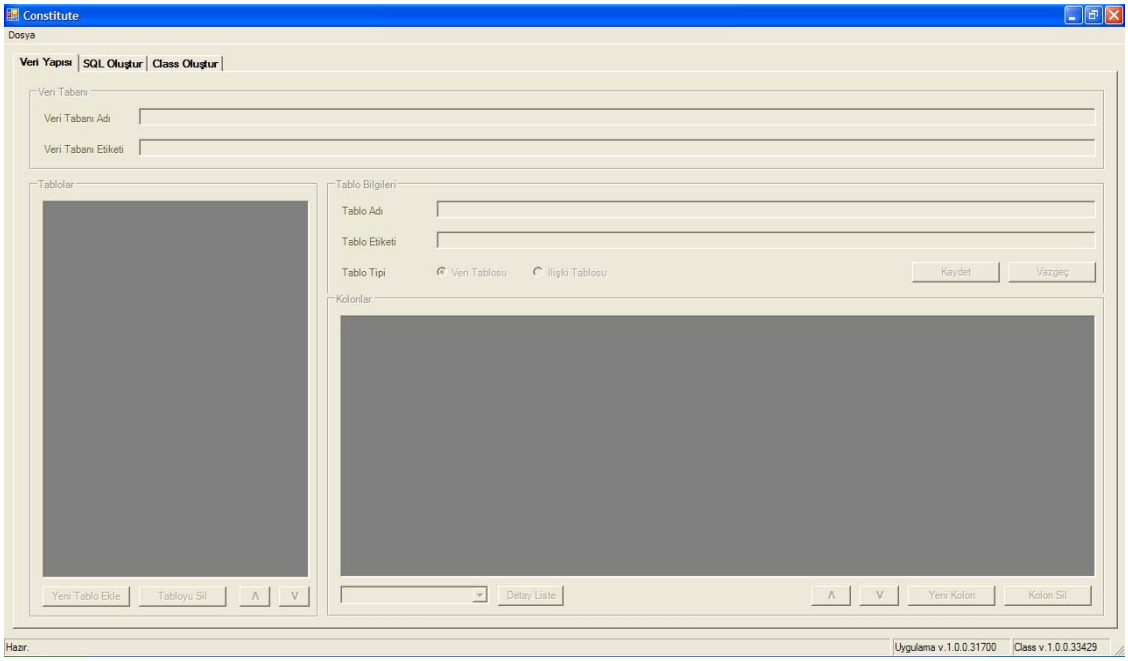
SQL Server 2005' te standart veri tiplerine ek olarak yedi tane de özel veri tipi vardır. Bunlar:

- **Bit:** 0,1 veya null verilerini tutar. Temel bayrak değerleri için kullanılır. False (yanlış) için 0, true (doğru) için ise 1 değeri tutulur.
- **Timestamp:** Sistem tarafından otomatik üretilen bir değerdir. Her veritabanı içinde sistem saatinden farklı bir zaman sayacı vardır. Bir tabloda sadece bir tane timestmap kolon olabilir, bu değer de satır eklenmesi, güncellenmesi gibi işlemlerde veritabanının içindeki sayacın ürettiği timestamp değerini tutar.
- **Uniqueidentifier:** Bir satırı tek (aynı kopyası bulunmayan) yapmak için tanımlanan 16 bitlik kimliktir (GUID-Global Unique Identifier).
- **Sql_variant:** Tuttuğu verinin tipi verinin içeriğine göre değişir. Maksimum 8000 byte veri tutar.
- **Cursor:** Kursör tanımlanan uygulamalar tarafından kullanılır. İşlemler için kursöre bir referans içerir. Bu veri tipi tabloda kullanılamaz.
- **Table:** Sonraki işlemde kullanılmak üzere bir "result set" tutmak için kullanılır. Bu veri tipi sütun için kullanılamaz. Sadece tetikleyici (trigger), saklı yordam (stored procedure) ve fonksiyonlarla tablo üretirken kullanılır.
- **Xml:** Boyutu 2 gigabyte a kadar olan xml dokümanlarını tutmaya yarar. Seçenekler yardımıyla sadece belirtilen yapıdaki bir xml dokümanının saklaması da sağlanabilinir.[12]

5. UYGULAMA

Bu bölümde geliştirilen uygulama örnek bir proje tasarımı ile desteklenerek anlatılmıştır.

Geliştirilen uygulama Windows uygulaması olarak tasarlanmıştır. Uygulamayı çalıştırmak için uygulama dosyaları içerisinde bulunan "Consitute.exe" dosyası kullanılır. Açılış dosyası daha hızlı erişim için masaüstüne kısayol oluşturalarak da kullanılabilinir. Uygulamanın ilk açılışında karşılaşılan ekran görünümü Şekil 5.1' de verilmiştir.

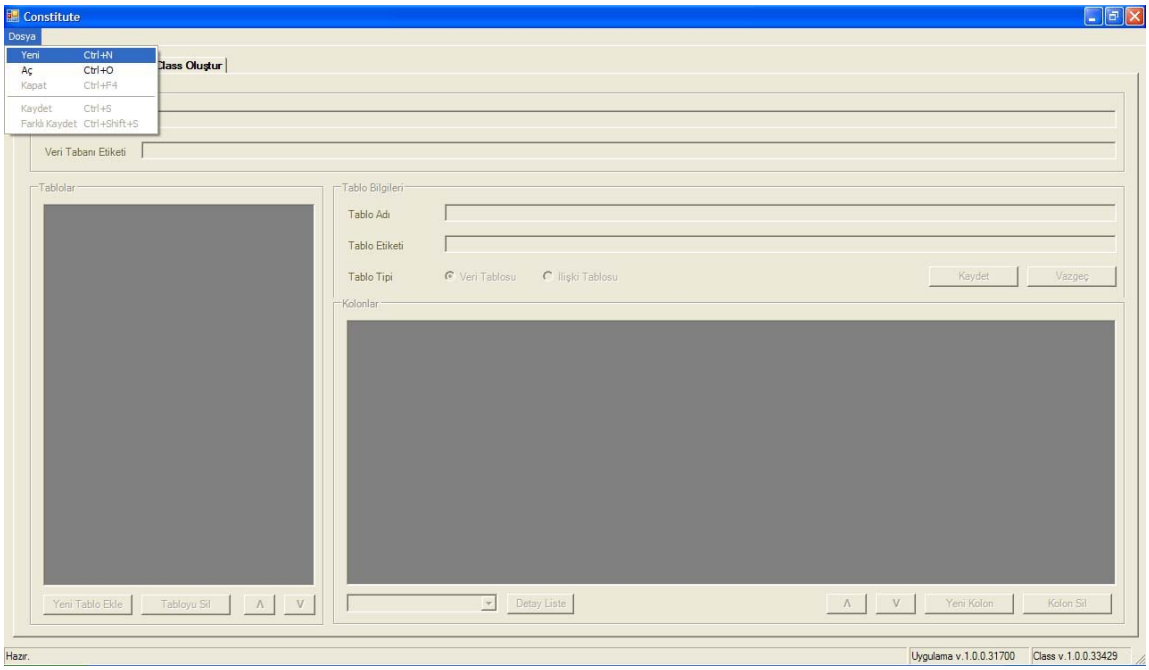


Şekil 5.1 Açılış Ekranı

5.1 Veri Yapısı

Uygulama çalıştırıldığında ana sekme olan "Veri Yapısı" sekmesi altında çalışır. Yeni bir proje ya da daha önceden oluşturulmuş bir proje açılarak çalışmaya başlanır. Aksi takdirde tüm butonlar pasif durumda bulunur.

Yeni bir iş açmak için üst kısımda bulunan “Dosya” menüsü altındaki “Yeni” seçeneği tıklanır. Bu işlem “Ctrl + N” klavye kombinasyonu kullanılarak da yapılabilmektedir.
Daha önceden çalışılıp kaydedilmiş bir projeyi açmak için ise yine “Dosya” menüsü altında bulunan “Aç” seçeneği tıklanır. Klavye kısayolu ise “Ctrl + O” olarak düzenlenmiştir. Aşağıda görülen Şekil 5.2’ de “Dosya” menüsünün görünümü verilmiştir.

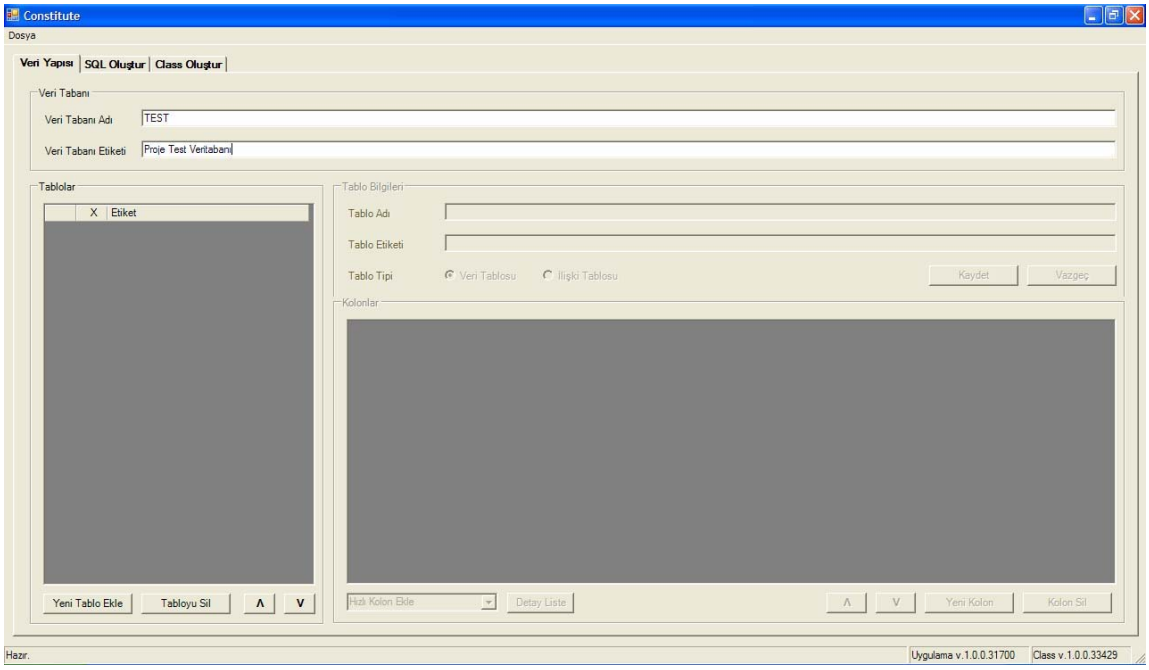


Şekil 5.2 Dosya Menüsünün Görünümü

Yeni bir proje açıldıktan sonra “Veri Yapısı” sekmesi altındaki butonların aktif hale geldiği gözlenir.
Yeni bir projede ilk olarak kullanılacak veri tabanı belirtilir. Veritabanı adı ve veritabanı etiket bilgileri girilerek belirtme işlemi yapılır.
Şekil 5.3’ de kullanılacak veritabanı belirtilirken alınan ekran görüntüsü bulunmaktadır.

Örnek:

Veri Tabanı Adı : TEST
Veri Tabanı Etiketi : Proje Test Veritabanı



Şekil 5.3 Veritabanı Belirtme

Veri tabanı oluşturulduktan sonra veri tabanında kullanılacak tabloların oluşturulmasına geçilir. Veri tabanına tablo eklemek için "Veri Yapısı" sekmesinde sol alt kısımda bulunan "Yeni Tablo Ekle" butonuna tıklanır. Bunun sonuncunda uygulamada "Tablo Bilgileri" bölümü aktif hale gelir.

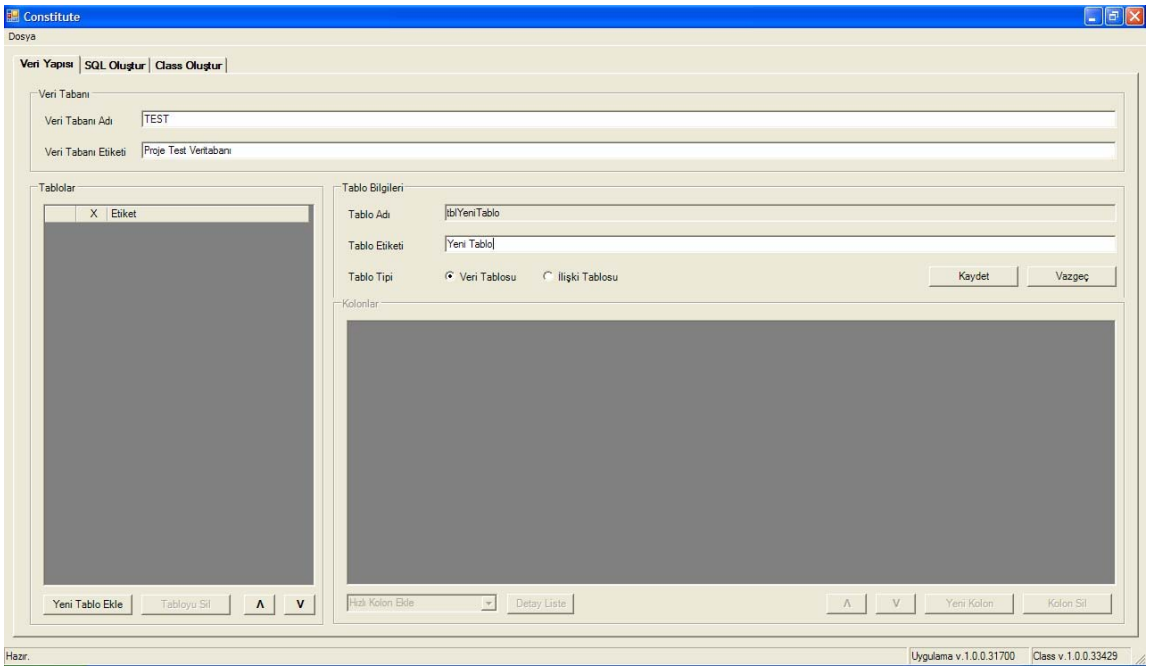
5.1.1 Tablo Bilgileri

"Tablo Bilgileri" bölümü altında belirtilmesi gereken özellikler şunlardır:

- **Tablo Adı:** Veri tabanında kullanılacak tablo adıdır. Bu alana doğrudan giriş izni verilmemiştir. Bunun nedeni kullanılacak boşlukların ve Türkçe karakterlerin tablo adına girilmesinin engellenmesidir. Tablo Adı bilgisi Tablo Etiketi bilgisine girilen değerden boşluklar ve Türkçe karakterler temizlenerek otomatik olarak oluşturulur.

- **Tablo Etiketi:** Tabloya verilen etiket bilgisidir. Bu bilgiden boşluklar ve Türkçe karakterler program içerisindeki fonksiyonlar yardımı ile temizlenerek Tablo Adı oluşturulur.

Şekil 5.4' te tablo bilgilerinin girildiği alanın ekran görünümü verilmiştir.



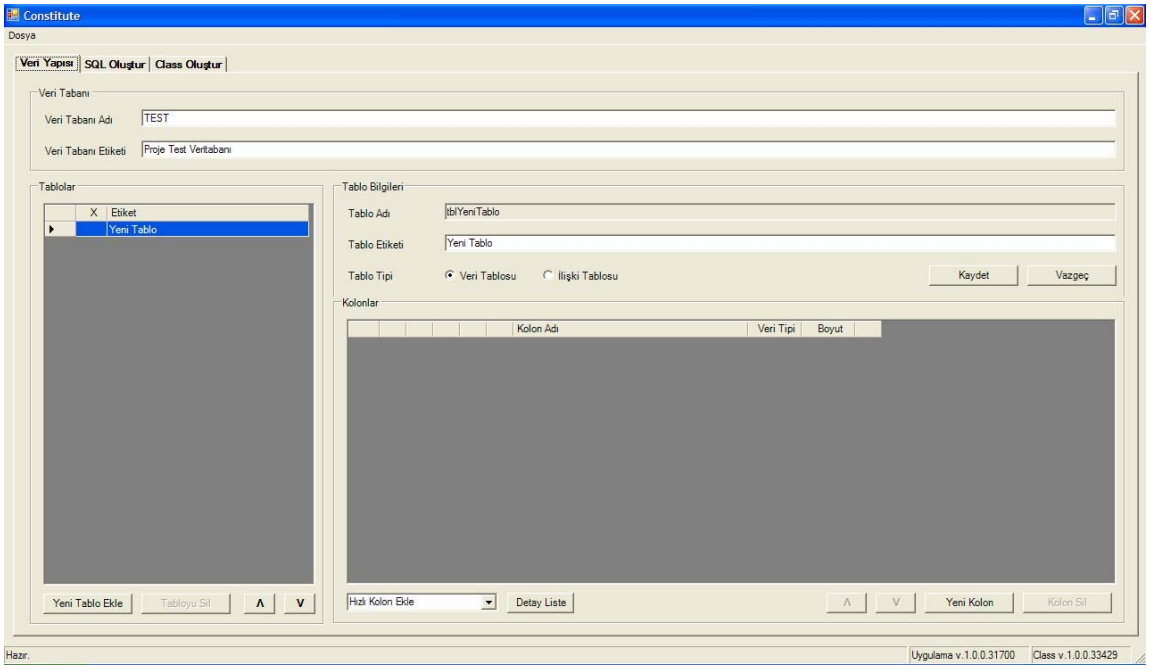
Şekil 5.4 Tablo Bilgileri

- **Tablo Tipi:** Oluşturulan tablonun içeriğine bağlı olarak tablonun tipinin seçilmesi için kullanılır. Burada kullanıcı için iki seçenek bulunmaktadır. Bunlar:

  - **Veri Tablosu :** İçerisinde verilerin tutulduğu tablo tipidir.
  - **İlişki Tablosu :** İçerisinde diğer tablolarda bulunan verilerin birbirleriyle aralarında bulunan ilişkilerin tutulduğu tablo tipidir.

Tablo Bilgileri girildikten sonra sağ kısımda bulunan "Kaydet" butonu ile tablo ekleme işlemi tamamlanır. Eğer girilen bilgilerle tablo eklemek istenmiyorsa "Vazgeç" butonu ile işlem iptal edilir.

"Kaydet" butonu ile devam edildiğini varsayarsak solda bulunan "Tablolar" bölümüne verilen tablo adı ile tablo listeye eklenir. Şekil 5.5' de Tablolar bölümünün ekran görüntüsü verilmiştir.



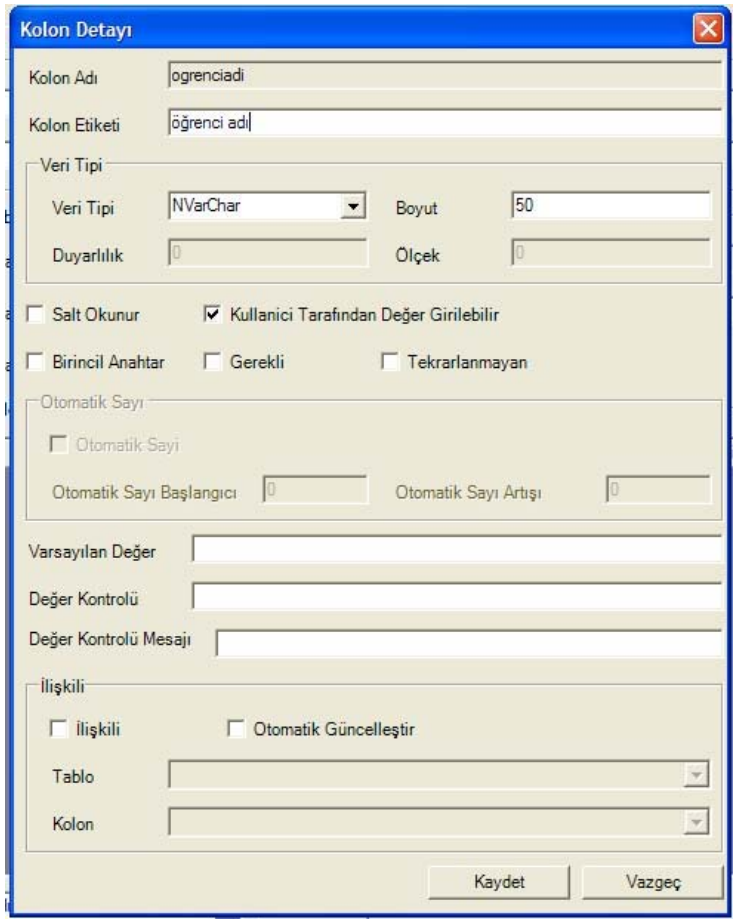
Şekil 5.5 Tablolar

Tablo ekleme işlemi tamamlandıktan sonra tablo içerisine kolonların eklenmesi işlemine geçilir. Yeni bir kolon eklemek için sağ altta bulunan "Yeni Kolon" butonu tıklanır. Kolonun özelliklerinin seçilmesi için yeni bir pencere açılır.

5.1.2 Kolonlar

Bu bölümde tabloda bulanacak kolonların tanımı yapılır.

"Kolon Detayı" altında açılan pencerede kullanılacak kolonun hangi özellikleri taşıyacağı seçilerek belirtilir. Şekil 5.6' da "Kolon Detayı" penceresinin ekran görünümü verilmiştir.



Şekil 5.6 Kolon Detayı

Şekil 5.6' da görülen pencerede girilecek özellikler şunlardır:

- **Kolon Etiketi:** Kolona verilen etiket bilgisidir. Bu bilgiden boşluklar ve Türkçe karakterler program içerisindeki fonksiyonlar yardımı ile temizlenerek Kolon Adı oluşturulur.

- **Veri Tipi:** Kolonda kullanılacak verinin tipinin seçildiği alandır. SQL veritabanında kullanılan tüm veri tipleri bu alanda tanımlanmıştır.

- **Boyut:** Kullanılan verinin veritabanında kaplayacağı alanın belirtiği bölümdür. Bu alan seçilen veri tipine göre otomatik olarak verilir ve yine veri tipine göre değiştirilebilir ya da değiştirilemez olarak ayarlanmıştır.

- **Duyarlılık:** Kullanılan verinin duyarlılığının girildiği bölümdür. Bu alan seçilen veri tipine göre otomatik olarak verilir ve yine veri tipine göre değiştirilebilir ya da değiştirilemez olarak ayarlanmıştır.

- **Ölçek:** Kullanılan verinin ölçek bilgisinin girildiği bölümdür. Seçilen veri tipine göre otomatik olarak verilir ve yine veri tipine göre değiştirilebilir ya da değiştirilemez olarak ayarlanmıştır.

- **Salt Okunur (SO):** Seçilerek kolona girilecek verinin SO olarak ayarlanması sağlanır. Salt okunur yazma yetkisinin verilmemesi demektir. Tabloda kolon bilgisinin sol tarafında bulunan bölüme SO harfleri eklenerek bu seçeneğin aktif olduğu belirtilir.

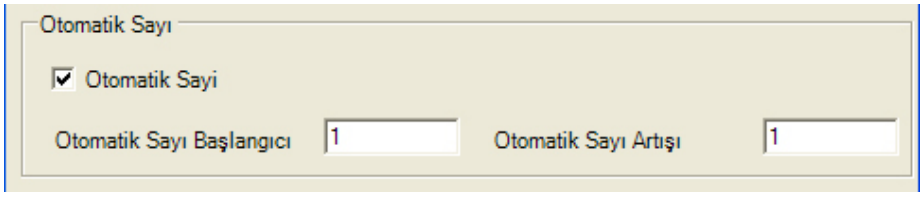
- **Kullanıcı Tarafından Değer Girilebilir (DG):** Seçilerek kullanıcının kolonda kullanılacak veriye değer girmesi sağlanır. Tablo da kolon bilgisinin sol tarafında bulunan bölüme DG harfleri eklenerek bu seçeneğin aktif olduğu belirtilir.

- **Birincil Anahtar (PK):** Kolonda kullanılan veriye PK özelliği verilmek için seçilir. Birincil anahtar seçilen alani eşşiz yapar, bir tabloda sadece bir tane birincil anahtar tanımlanır. PK olarak seçilen alanlar otomatik olarak boş bırakılamaz (not null) olarak seçilir. Tablo da kolon bilgisinin sol tarafında bulunan bölüme PK harfleri eklenerek bu seçeneğin aktif olduğu belirtilir.

- **Gerekli (NN):** Kolonda kullanılan verinin gerekli olup olmayacağı bu alan seçilerek belirtilir. Seçilirse kullanıcı tarafından girilmesi zorunlu olur. Tabloda kolon bilgisinin sağ tarafında bulunan bölüme NN eklenerek bu seçeneğin aktif olduğu belirtilir. Eğer seçilmemişse aynı bölümde NUL bilgisi verilir.

- **Tekrarlanmayan (UQ):** Kolonda kullanılan verinin eşsiz olup olmayacağı bu alan seçilerek belirtilir. Yani eşsiz yapılan alana girilen değer tekrarlanılarak kullanılamaz. Tabloda kolon bilgisinin sol tarafında bulunan bölüme UQ harfleri eklenerek bu seçeneğin aktif olduğu belirtilir.

- **Otomatik Sayı:** Bu bölüm veri tipi olarak sayı tipinde bir değişken seçilirse aktif hale gelir. Bu sayı değerinin sistem tarafından arttırılması isteniyorsa, "Otomatik Sayı" yanındaki kutucuk seçilip, sayının kaçtan başlayacağı ve artış değeri girilir. Aşağıda bulunan Şekil 5.7' de görülen ekran görüntüsünde Otomatik Sayı bilgisinin girilişi gösterilmiştir.



Şekil 5.7 Otomatik Sayı

- **Varsayılan Değer:** Kolon içerisinde kullanılan verinin ilk olarak aldığı değerdir. Kullanıcı yeni bir değer girmezse o şekilde veritabanında tutulur. Bu alanda SQL veritabanında bulunan fonksiyonlar kullanılabilinir.

Örnek: Herhangi bir kolonda veritabanına girilen değerin kayıt tarihi tutulmak isteniyorsa kayıt tarihi adında bir kolon açılarak bu kolonun varsayılan değeri *GETDATE()* fonksiyonu şeklinde ayarlanır. Her veri girişinde bu değer otomatik olarak sistem saatine bakılarak uygulama tarafından kaydedilir.

- **Değer Kontrolü:** Kolon içerisine girilen veri değerinin kontrolü bu alana SQL fonksiyonlarının girilmesi ile yapılır.

- **Değer Kontrolü Mesajı:** Kullanılan fonksiyon sonunda eğer bir kontrol yapılmışsa veya kullanıcıya verilmek istenen mesaj varsa bu alana girilir.

Örnek: Girilen değerin boyutu SQL fonksiyonu *LEN(<X>)* kullanılarak sınırlandırılabilir ve eğer bu sınır aşılırsa kullanıcıya seçilen mesaj verilir. "X" değeri kolona girilen değeri çeker.

- **İlişkili:** Oluşturulan kolonun başka bir kolonla ilişkisi varsa bu alanda belirtilir. İlişkili seçeneği aktif hale getirilirse kolonun hangi tablodaki kolon ile ilişkilendirileceği belirtilmelidir.

- **Otomatik Güncelleştir:** Kolonun ilişki kurulan kolondaki değere bakılarak güncellenmesi isteniyorsa bu alan seçilir. Böylece ilişki kurulan kolondaki bilgi değiştiği anda oluşturulan kolon da bu değeri alır.

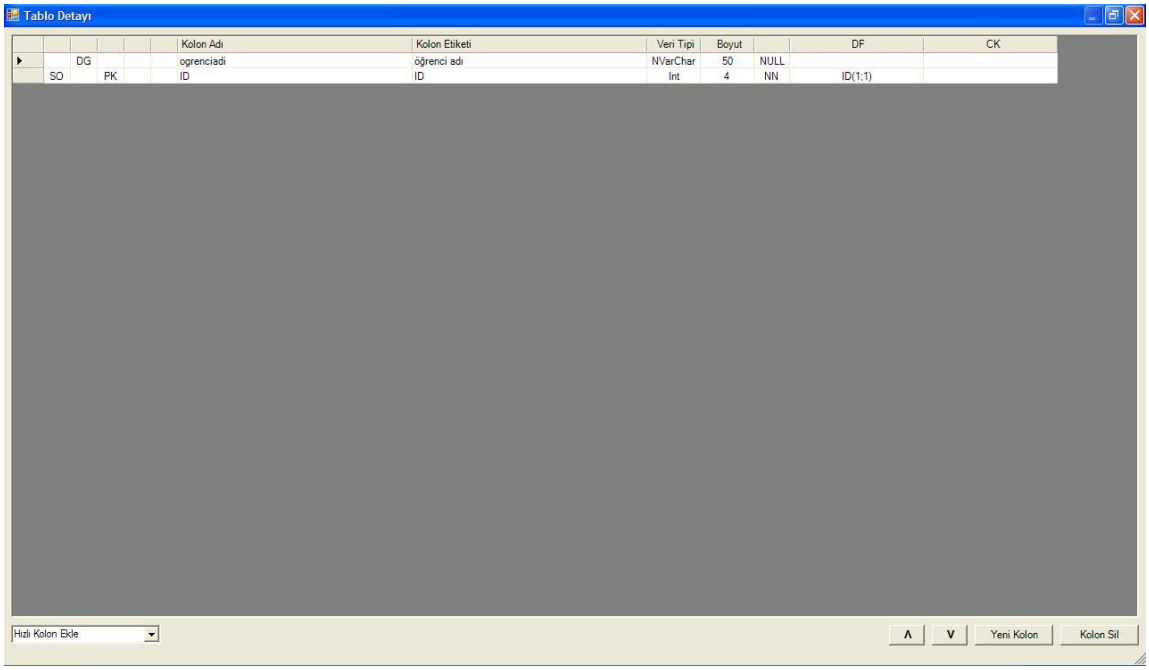
Bu bilgiler daha sonra istenildiği takdirde kolonun üzerine tıklanarak açılan "Kolon Detayı" bölümünde değiştirilebilmektedir.

Tabloda oluşturulan kolonların sıraları değiştirilmek isteniyorsa aşağıda "Yeni Kolon" butonunun yanında bulunan aşağı ve yukarı ok butonları kullanılır.

Tabloda oluşturulan kolonların silinmesi isteniyorsa kolon seçilerek sağ altta bulunan "Kolon Sil" butonu tıklanır.

Tablo oluşturulurken sık kullanılan kolonlar "Hızlı Kolon Ekle" butonu altında toplanmıştır. Bu kolonlar seçilerek daha hızlı kolon oluşturma işlemi yapılır. Bu bölüm sadece hızlı çalışma için oluşturulduğu için belirli sayıda kolon ile sınırlandırılmıştır. Kullanıcı tercih ederse önceden oluşturulmuş bu kolonları tablosuna ekler.

“Hızlı Kolon” ekleme seçeneğinin sağındaki “Detay Liste” butonu tıklanarak Tablo içeriğinin detaylarının bulunduğu pencereye erişim sağlanır. Şekil 5.8’ de verilen Detay liste penceresinde tabloda bulunacak kolonların detayları ile ilgili tüm işlemler yapılabilmektedir.

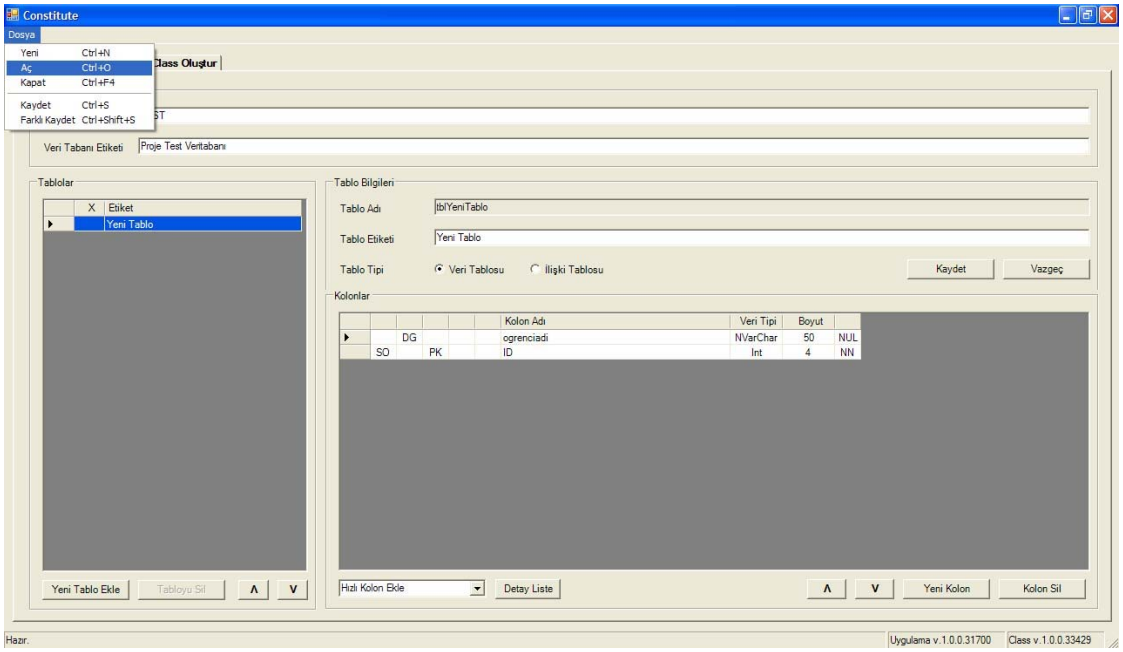


Şekil 5.8 Detay Liste

Tabloda bulunan kolonların detayları ile ilgili işlemler bitirildikten sonra, proje çalışmasının ana ekranına dönmek için pencerenin kapatılması yeterlidir.

5.1.3 Dosya Menüsü

Bir proje üzerinde çalışılırken "Dosya" menüsüne giriş yapılırsa Şekil 5.9' daki ekran görüntüsü elde edilir.

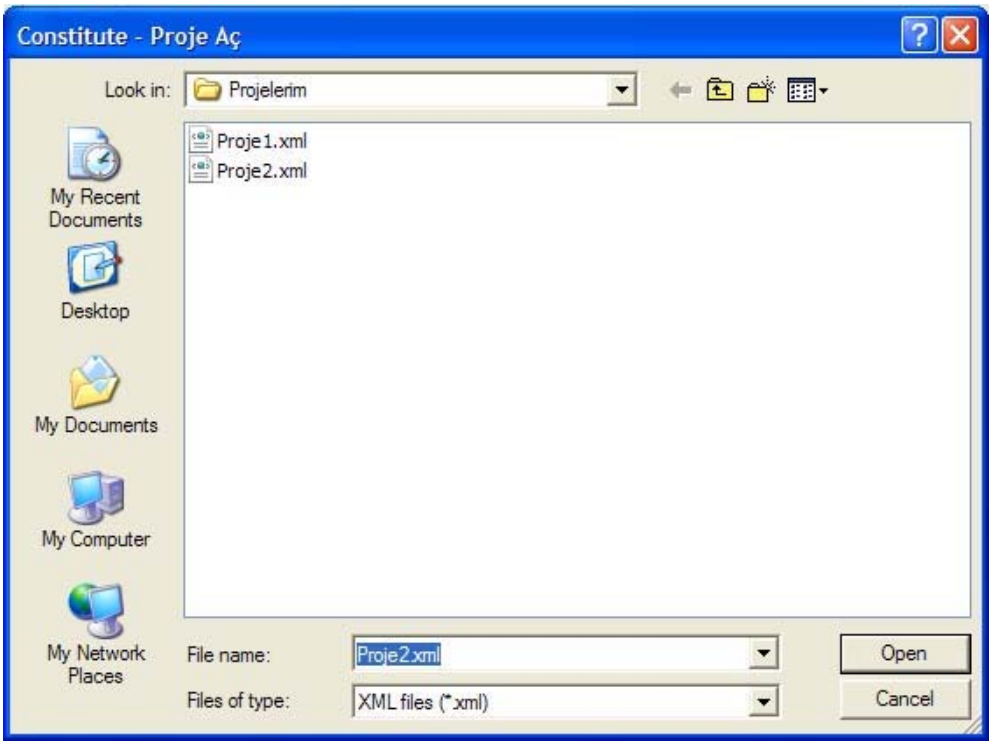


Şekil 5.9 Çalışılan Projenin Dosya Menüsü Görünümü

Bu menüde aşağıda belirtilen işlemler yapılır.

- **Yeni:** Yeni Proje açılma işlemi yapılır.
- **Aç:** Daha önceden çalışılıp kaydedilmiş Projenin açılması işlemi yapılır.

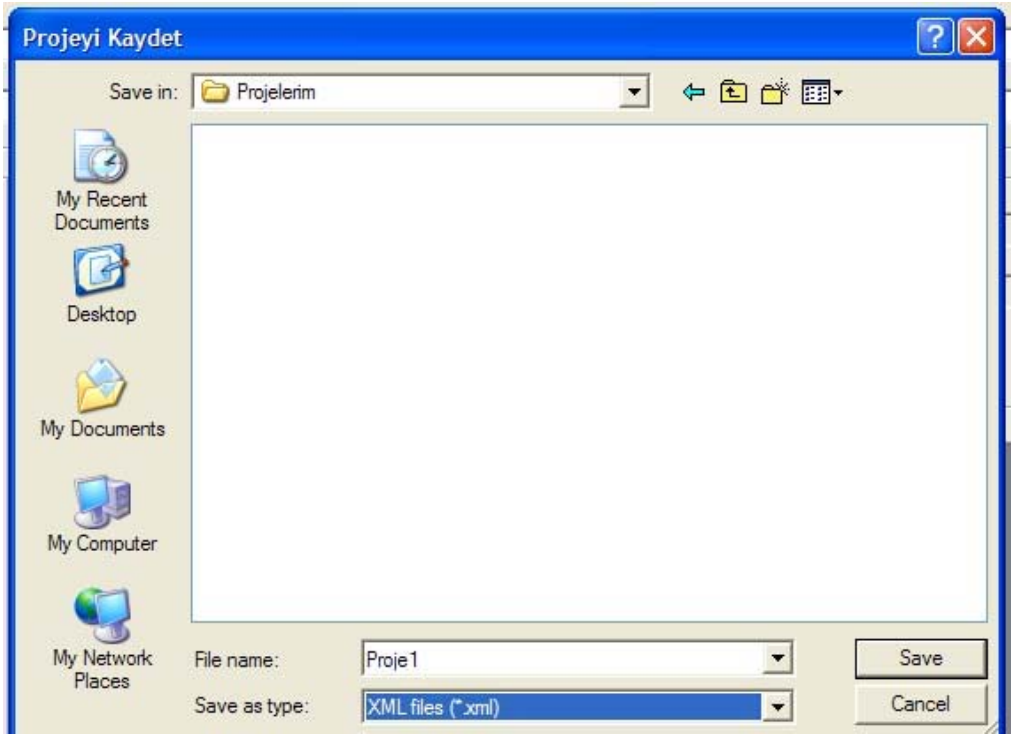
Dosya menüsündeki "Aç" butonu tıklanırsa Şekil 5.10' da ekran görünümü verilen "Proje Aç" penceresine erişilir.



Şekil 5.10 Proje Aç Ekranı

- **Kapat :** Çalışılan projenin kapatılması işlemi yapılır.
- **Kaydet:** Çalışılan projenin kaydedilmesi işlemi yapılır.

Kaydet butonuna tıklandığında Şekil 5.11' de ekran görünümü verilen "Projeyi Kaydet" penceresi açılır. Bu pencerede projenin bilgisayarda hangi lokasyona, hangi isimle kaydedileceği belirtilerek kaydetme işlemi yapılır. Kaydedilen projeler XML türünde saklanır.

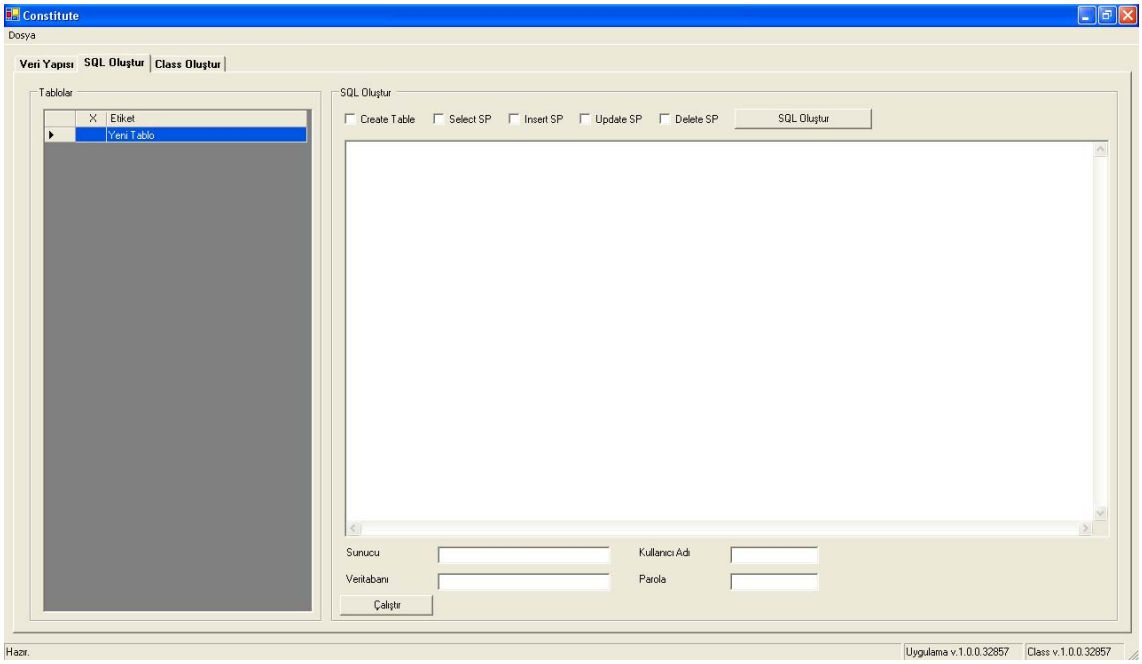


Şekil 5.11 Projeyi Kaydetme Ekranı

- **Farklı Kaydet :** Daha önce kaydedilmiş bir projenin, öncekinden farklı bir isim ile farklı bir lokasyona kaydedilmesi işlemine imkan sunar. Kaydet bölümünde bulunanan aşamalar bu bölümde de geçerlidir.

5.2 SQL Oluştur

Tablonun oluşturulması ve SQL sorguları için kullanılacak saklı yordam (stored procedure) kodlarının üretilmesi işlemi "SQL Oluştur" sekmesi altında yapılır. Bu menü ilk açıldığında bu alandaki bölümler pasif durumdadır. Oluşturulması istenen tablo solda bulunan listeden seçildikten sonra sağda bulunan "SQL Oluştur" bölümü aktif hale gelir. Şekil 5.12' de tablo seçimi sonrası aktif hale gelen SQL Oluştur bölümünün ekran görünümü verilmiştir.



Şekil 5.12 SQL Oluştur Ana Ekran

Bu bölümde oluşturulması istenilen yordamlar seçilir ve sağda bulunan "SQL Oluştur" butonu tıklanır. Geliştirilen uygulama ile üretilebilecek saklı yordamlar şunlardır:

- **Tablo Oluşturma Saklı Yordamı (Create Table SP) :** Seçilen tablonun oluşturulması için kullanılacak kodların program tarafından üretilmesi için bu saklı yordam kullanılır.

- **Seçim Saklı Yordamı (Select SP) :** SQL veritabanında Seçim (Select) sorgusunun yapılması için kullanılacak saklı yordamın üretilmesi işlemini yapar.

- **Ekleme Saklı Yordamı (Insert SP) :** SQL veritabanında Ekleme (Insert) işleminin yapılması için kullanılacak saklı yordamın üretilmesi işlemini yapar.

- **Güncelleme Saklı Yordamı (Update SP) :** SQL veritabanında Güncelleme (Update) işleminin yapılması için kullanılacak saklı yordamın üretilmesi işlemini yapar.

- **Silme Saklı Yordamı (Delete SP) :** SQL veritabanında Silme (Delete) işleminin yapılması için kullanılacak saklı yordamın üretilmesi işlemini yapar.

5.2.1 SQL Oluştur Bölümü ile SQL Sunucusu Bağlantısı

SQL Oluştur bölümünde oluşturulan saklı yordamlar, doğrudan SQL Sunucusu ile bağlantı kurulup veritabanına işlenebilir. Bu işlemin yapılması için SQL Sunucusu bağlantı bilgilerinin yazılımda bildirilmesi gereklidir. SQL Sunucusu bağlantısı için gerekli olan bilgiler aşağıdaki gibidir.

- **Sunucu :** SQL Sunucusunun bilgisi
- **Veritabanı :** Tablonun eklenmesi istenen veritabanının adı
- **Kullanıcı Adı :** SQL Sunucusu üzerinde bağlantı kurulacak veritabanına erişim yetkisi olan kullanıcının adı
- **Parola :** SQL Sunucusu üzerinde bağlantı kurulacak veritabanına erişim yetkisi olan kullanıcının paralosı

Aşağıda bulunan Şekil 5.13’ de SQL Sunucusu bağlantısının yapılması için gereken tanımlamalar, kullanılan sunucunun bilgileri girilerek verilmiştir.

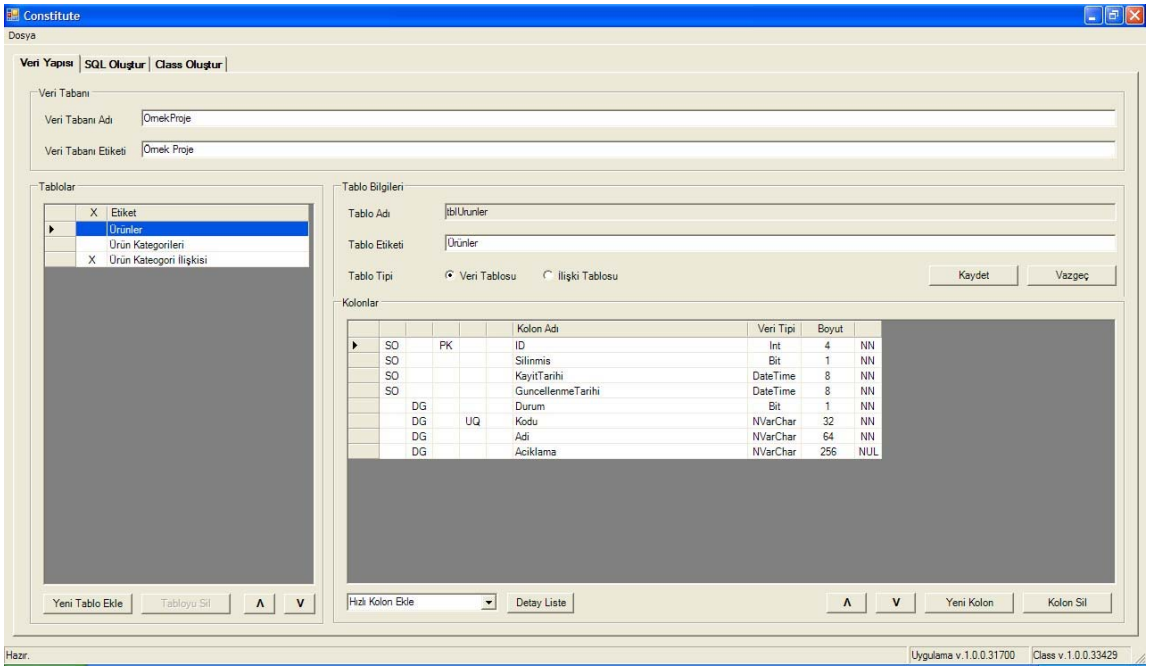
Şekil 5.13 SQL Sunucusu bağlantı ayarları

5.2.2 SQL Oluştur Örnek Proje Tasarımı

Bu bölümde geliştirilen uygulamanın kullanımının daha rahat anlaşılabilmesi için örnek bir proje geliştirilmiştir. Bu projede günlük yaşamda sıkça karşılaşılan bir senaryo düşünülmüştür. Proje için yeni bir veritabanı oluşturulup, bu veritabanın içeresine iki veri tablosu ve bu tablolar arasındaki ilişkiyi gösteren bir ilişki tablosu eklenmiştir. İlk olarak projede kullanılmak üzere “OrnekProje” adıyla yeni bir veritabanı oluşturulur. Daha sonra bu veritabanı içerisine, belirtildiği gibi iki veri tablosu ve bir ilişki tablosu eklenir. Bu tabloları aşağıdaki gibi ayrı ayrı inceleyebiliriz.

5.2.2.1 Ürünler Tablosu

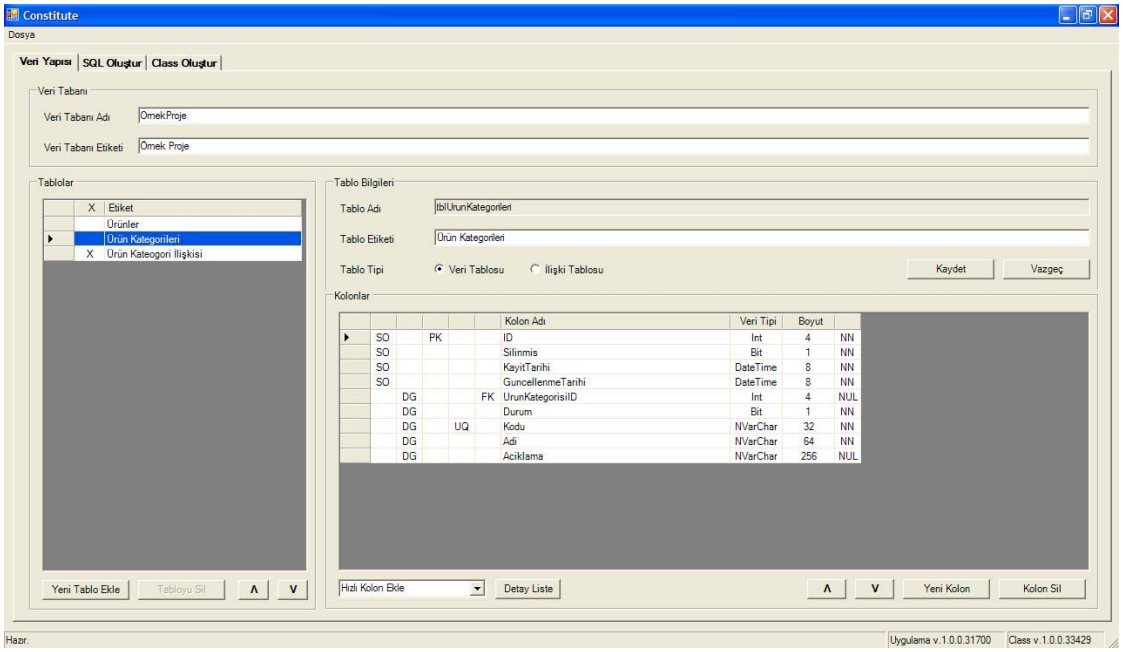
Ürünler tablosu içerisinde ürün bilgilerinin bulunacağı verileri içerecektir. Bu nedenle Veri tablosu olarak oluşturulmuştur. Projede oluşturulmuş tablonun içeriği Şekil 5.14’ deki ekran görüntüsünde verilmiştir.



Şekil 5.14 Ürünler Tablosu Ekran Görünümü

5.2.2.2 Ürün Kategorileri Tablosu

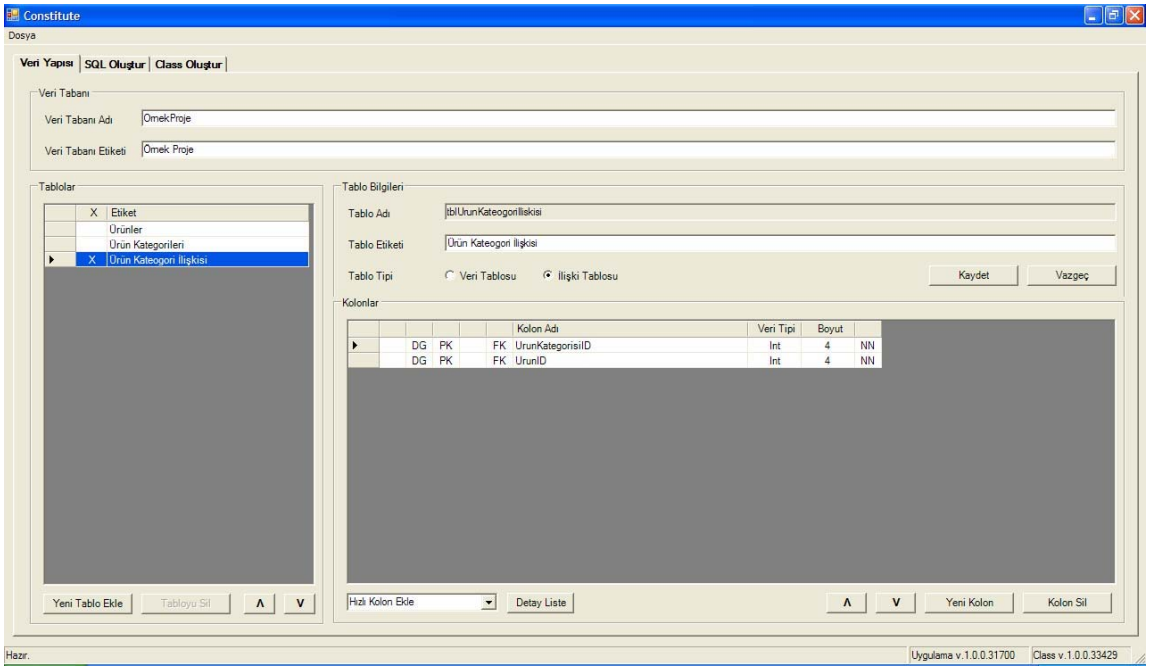
Bu tabloya ürün kategorilerinin detay bilgilerinin verileri girilecektir. Bu nedenle veri tablosu olarak oluşturulmuştur. Projede oluşturulmuş tablonun içeriği Şekil 5.15' deki ekran görüntüsünde verilmiştir.



Şekil 5.15 Ürün Kategorileri Tablosu Ekran Görünümü

5.2.2.3 Ürün Kategori İlişkisi Tablosu

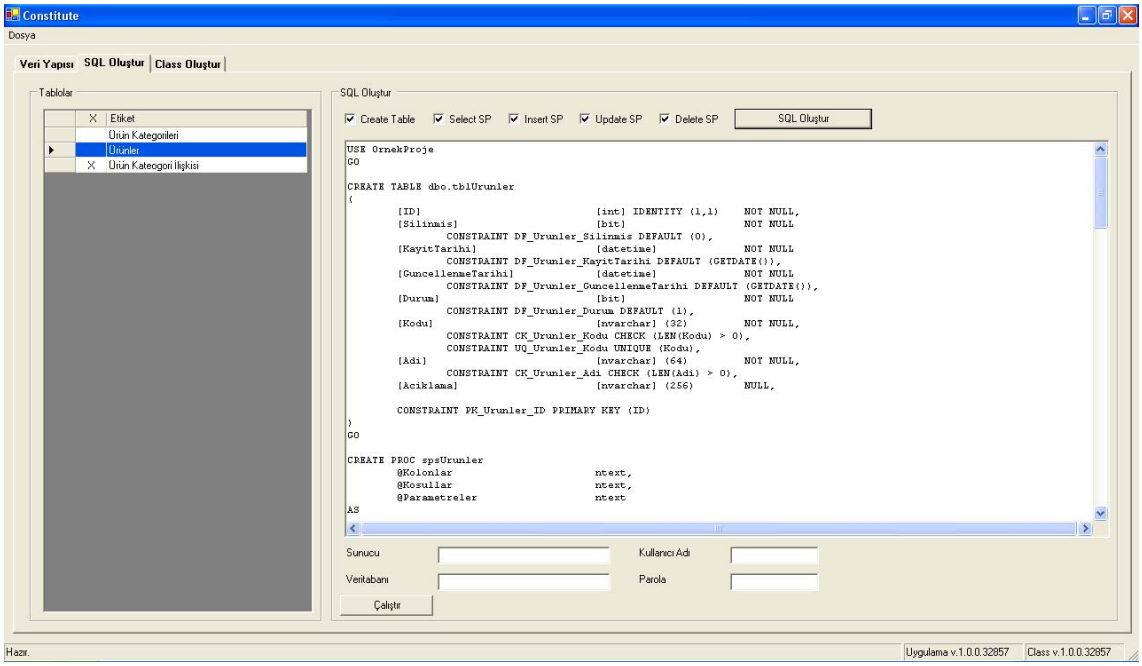
Bu tabloda ürünler ve ürün kategorileri arasındaki ilişki bilgilerinin tutulması planlanmıştır. Bu sebeple İlişki tablosu olarak oluşturulmuştur. Ürünler ve Ürün Kategorileri tablolarında bulunan ID kolonları ile ilişkili iki kolon oluşturulmuştur. Projede oluşturulmuş tablonun içeriği Şekil 5.16' deki ekran görüntüsünde verilmiştir.



Şekil 5.16 Ürün Kategori İlişkisi Tablosu Ekran Görünümü

5.2.2.4 SQL Oluştur Aşamasında Üretilen Saklı Yordam Kodları

Bu bölümde projede oluşturulan tablolar için üretilecek saklı yordam kodları ve bu kodların içerikleri ile ilgili bilgiler bulunmaktadır. Şekil 5.17' de Ürünler Tablosu için geliştirilen uygulama ile üretilebilinecek tüm saklı yordam türlerini içeren SQL Oluştur bölümünün ekran görüntüsü verilmiştir.



Şekil 5.17 Ürünler Tablosu SQL Oluştur Ekran Görüntüsü

Projede oluşturulan her tablo için ayrı ayrı seçim yapıp saklı yordam kodları üretilmelidir.

Geliştirilen yazılım aracılığıyla üretilen saklı yordam kodlarının anlaşılması amacıyla veri tablosu olarak planlanan Ürünler Tablosu' na ait saklı yordam kodları aşağıda verilmiştir.

- **Tablo Oluşturma (Create Table SP) Saklı Yordamı Kodları**

Saklı yordam kodunda tablo oluşturulurken dikkat edilmesi gereken noktalar; *USE* SQL komutu ile hangi veritabanının kullanılacağının belirtilmesi, *CREATE TABLE* SQL komutu ile tablo isminin belirtilmesi ve bu tabloda kullanılacak kolonların tanımlanması işlemleridir. Aşağıda Ürünler tablosunu oluşturacak saklı yordam kodları verilmiştir.

```
USE OrnekProje
GO

CREATE TABLE dbo.tblUrunler
(
	[ID]				[int] IDENTITY (1,1)NOT NULL,
	[Silinmis]			[bit]			NOT NULL
		CONSTRAINT DF_Urunler_Silinmis DEFAULT (0),
	[KayitTarihi]			[datetime]		NOT NULL
		CONSTRAINT DF_Urunler_KayitTarihi DEFAULT (GETDATE()),
	[GuncellenmeTarihi]		[datetime]		NOT NULL
		CONSTRAINT DF_Urunler_GuncellenmeTarihi DEFAULT
(GETDATE()),
	[Durum]					[bit]			NOT NULL
		CONSTRAINT DF_Urunler_Durum DEFAULT (1),
	[Kodu]				[nvarchar] (32)		NOT NULL,
		CONSTRAINT CK_Urunler_Kodu CHECK (LEN(Kodu) > 0),
		CONSTRAINT UQ_Urunler_Kodu UNIQUE (Kodu),
	[Adi]				[nvarchar] (64)		NOT NULL,
		CONSTRAINT CK_Urunler_Adi CHECK (LEN(Adi) > 0),
	[Aciklama]			[nvarchar] (256)	NULL,

	CONSTRAINT PK_Urunler_ID PRIMARY KEY (ID)
)
GO
```

- **Seçim Saklı Yordamı (Select SP)**

Bu yordamın oluşumunu anlamak için *SELECT* SQL komutunun kullanımının bilinmesi gereklidir. Bu komutun içeriğini üç ana bölümde inceleyebiliriz.

- Tabloda bulunan ve sorgu sonrasında gösterilmesi istenen kolonların belirtilmesi
- Tabloda aradığımız bilgiye ulaşmak için verilen koşulların belirtilmesi
- Sorgu sonucunun gösterim şeklinin girilen parametrik değer ile belirtilmesi
  Örnek: Group By parametresi

Belirtilen üç bölüm, veritabanını kullanacak kişi tarafından belirtilecektir. Bu nedenle saklı yordam kodu içerisinde bu bölümler farklı parametreler tanımlanarak, dışarıdan gelecek bilgi girişine hazır hale getirilir. SQL sorgusunun doğru çalışabilmesi için bu

parametrelerin doğru olarak girilmesi önemlidir. Tanımlanan parametrelerin adları aşağıda verilmiştir.

- Kolonlar
- Koşullar
- Parametreler

Hatalı girişlerin işlenmesini engellemek adına saklı yordam komutları altında girilen değerlerin kontrolleri yapılmıştır. Parametre içeriğine uygun olmayan bir değer girildiğinde, yazılım kullanıcıya girilen değerin geçersiz olduğunu belirten bir hata mesajı verir. Belirtilen parametrelerin tanımlanması, kontrollerinin yapılması ve sorgu komutu içerisine dahil edilmesi işlemlerini yapan kod bloğu aşağıda verilmiştir.

```
CREATE PROC spsUrunler
	@Kolonlar			ntext,
	@Kosullar			ntext,
	@Parametreler				ntext
AS
	IF	((@Kolonlar LIKE '') OR
		 (@Kolonlar IS NULL) OR
		 ((CHARINDEX(';', @Kolonlar)) <> 0))
		RAISERROR('Kolonlar için girilen değer geçersiz!', 16, 1)

	ELSE IF	((@Kosullar LIKE '') OR
		 ((CHARINDEX(';', @Kosullar)) <> 0))
		RAISERROR('Koşullar için girilen değer geçersiz!', 16, 1)

	ELSE IF	((@Parametreler LIKE '') OR
		 ((CHARINDEX(';', @Parametreler)) <> 0))
		RAISERROR('Parametreler için girilen değer geçersiz!', 16, 1)

	ELSE
	BEGIN
		IF	(@Kosullar IS NULL) AND (@Parametreler IS NULL)
			EXEC ('SELECT ' + @Kolonlar + ' FROM tblUrunler')

		ELSE IF	(@Kosullar IS NOT NULL) AND (@Parametreler IS
NULL)
			EXEC ('SELECT ' + @Kolonlar + ' FROM tblUrunler WHERE ' +
@Kosullar)

		ELSE IF	(@Kosullar IS NULL) AND (@Parametreler IS NOT
NULL)
			EXEC ('SELECT ' + @Kolonlar + ' FROM tblUrunler ' +
@Parametreler)

		ELSE
			EXEC ('SELECT ' + @Kolonlar + ' FROM tblUrunler WHERE ' +
@Kosullar + ' ' + @Parametreler)
	END
GO
```

- **Ekleme Saklı Yordamı (Insert SP)**

Bu yordam istenilen tabloya kayıt eklenmesi için kullanılır. Tablo oluşturulurken tabloda bulunan her bir kolon için daha önce belirtildiği gibi veri girişi belirli koşullara dayandırılabilir. Bu yordamda öncelikli olarak girilmek istenen verinin, eğer herhangi bir koşul tanımlandıysa bu koşula uyup uymadığı kontrol edilir. Eğer verilen koşullara uygun bir veri değil ise, kolon detayında kullanıcıya verilmesi istenen hata mesajı ekrana yansıtılır. Kontrol işleminin yapıldığı kod bloğu aşağıdaki gibidir.

```
        IF (@Kosul IS NOT NULL)
        BEGIN
                EXEC sp_executesql
                        @Kosul,
                        N'@Durum [bit],@Kodu [nvarchar] (32),@Adi [nvarchar]
(64),@Aciklama [nvarchar] (256)',
                        @Durum = @Durum,@Kodu = @Kodu,@Adi =
@Adi,@Aciklama = @Aciklama
        END
```

Veri girişi için koşul tanımlanmadıysa ya da belirtilen koşullara uygun bir veri girilmek isteniyorsa yazılım herhangi bir hata üretmez. Hata olup olmadığının kontrolü de yapıldıktan sonra, tabloda bulunan her kolon için tanımlanan parametre üzerinden girilmek istenen verinin değerlerinin gönderimi yapılır. Girilen veriden sonra veritabanında kalınan yerin geri dönüşünün yapılması için yordam içerisinde yeni ID değerinin ataması yapılır. Tabloya veri eklenmesi için yordam içerisinde tanımlanan kod bloğu aşağıdaki gibidir.

```
IF (@@ERROR = 0)
        BEGIN
                INSERT INTO tblUrunler (
                        Durum,
                        Kodu,
                        Adi,
                        Aciklama
                )
                VALUES (
                        @Durum,
                        @Kodu,
                        @Adi,
                        @Aciklama
                )

                SET @ID = @@IDENTITY
         END
```

- **Güncelleme Saklı Yordamı (Update SP)**

Bu yordamda da Ekleme yordamında olduğu gibi veri girişi için koşul kontrolü yapılır. Eğer koşul tamınlanmadıysa veya koşullara uygun bir veri girilmek isteniyorsa, hata geri bildirimi yapılmaz. Hangi veride güncelleme yapılacağı ID bilgisi yardımıyla kontrol edilir. Güncellenmek istenen verinin “silinmis” durumu kontrolü yapılarak güncellemeye uygun olup olmadığı da belirlendikten sonra işlem parametreler üzererinden değer bildirimi ile tamamlanır. Yordam içerisinde güncelleme işleminin yapıldığı kod bloğu aşağıda verilmiştir.

```
IF (@@ERROR = 0)
BEGIN
        UPDATE tblUrunler
        SET
                GuncellenmeTarihi = GETDATE(),
                Durum = @Durum,
                Kodu = @Kodu,
                Adi = @Adi,
                Aciklama = @Aciklama
        WHERE ((ID = @ID) AND (Silinmis = 0))
END
```

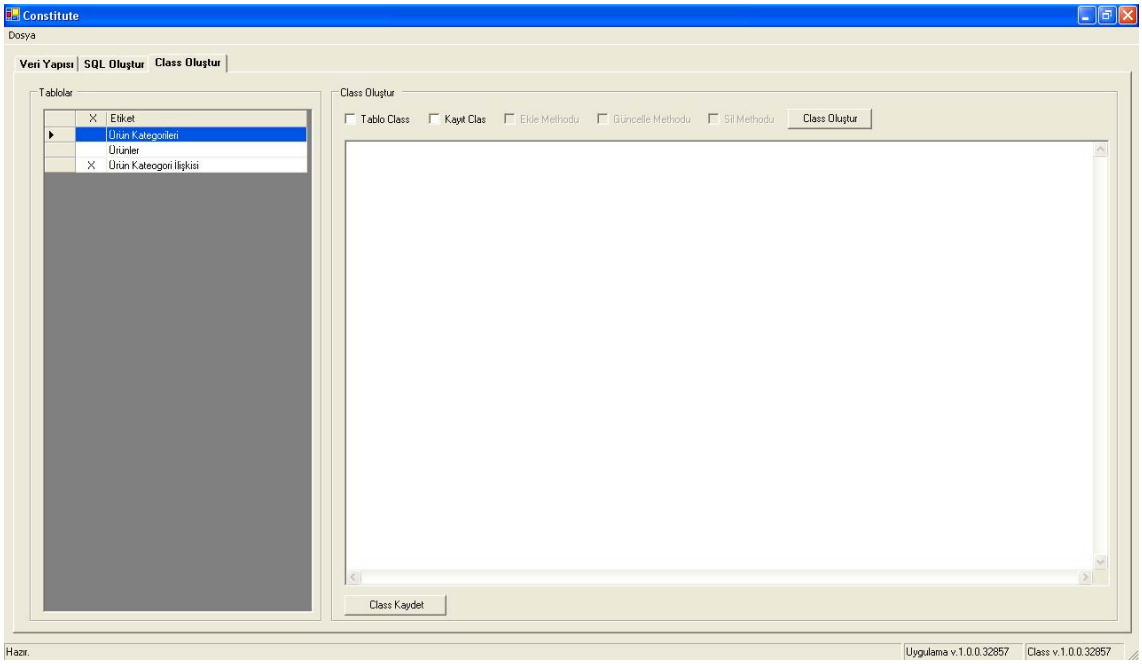
- **Silme Saklı Yordamı (Delete SP)**

Bu saklı yordamda silme işlemi için belirtilen tanımlamalar bulunmaktadır. Veri tabanından veri silme işlemi hiçbir şekilde tavsiye edilmediği için, üretilen her projede ek bir kontrol kolonu oluşturularak buna "Silinmis" adı verilir. Bu kolon bir kontrol biti olarak görev üstlenir. Bu kolonun değeri veri ilk girildiğinde "0" olarak ayarlanır ve veri silinmemiş kabul edilir. Bu değerin silinmesi için kullanıcı bir işlem yaparsa bu yordam aracılığıyla Silinmis kolonunun değeri "1" olarak ayarlanır ve veri silinmiş kabul edilir. Verinin geri dönüşünün yapılması istenirse bu değerin tekrar "0" olarak güncellenmesi yeterlidir. Aşağıda yordam içerisinde bu işlemleri gerçekleştiren kod bloğu verilmiştir.

```
IF (@@ERROR = 0)
BEGIN
        UPDATE tblUrunler
        SET
                Silinmis = 1,
                GuncellenmeTarihi = GETDATE(),
                Kodu = 'Silinmiş Kayıt [' + LEFT(Kodu, 14 - LEN(@ID)) + ']['
+ CAST(@ID AS varchar) + ']'
        WHERE ((ID = @ID) AND (Silinmis = 0))
END
```

5.3 Sınıf Oluştur

Veri erişim katmanının oluşturulması üretilecek olan sınıf kodları ile sağlanır. Veri katmanı ile bağlantıyı kurmak için kullanılan sınıf kodlarının üretilmesi işlemi “Class Oluştur” sekmesi altında yapılır. Bu sekme ilk açıldığında tüm bölümler pasif durumdadır. Projede tanımlanan her tablo için ayrı ayrı sınıf kodlarının üretilmesi gereklidir. Solda bulunan listeden sınıf kodlarının üretilmesi istenen tablo seçilerek kod üretim alanı olan sağ taraftaki bölüm aktif hale getirilir. Şekil 4.18’ de verilen ekran görüntüsünde sol tarafta oluşturulan tablolar liste halinde bulunmakta, sağ tarafta ise üretilecek sınıf türleri ve kod üretim alanı görülmektedir.



Şekil 5.18 Class Oluştur Ekran Görünümü

Sınıf Oluştur bölümünde tablo seçimi yapıldıktan sonra iki ana seçenek aktif olarak gözükmektedir. Bunlar:

- **Tablo Class:** Veri yapısı oluşturulmuş tablonun temel sınıf yapısının kodlarının üretilmesi işlemini gerçekleştirir.

- **Kayıt Class**: Oluşturulan tablonun kayıt işlemlerini gerçekleştirecek sınıf kodlarının üretilmesi işlemini gerçekleştirir. Kayıt Class seçildikten sonra kayıt işlemlerinin yapıldığı üç metot aktif hale gelir. Bu metotlar:

  - **Ekle Metodu:** Tabloya ekleme yapılması için kullanılacak sınıf kodlarının üretilmesi işlemini gerçekleştirir.

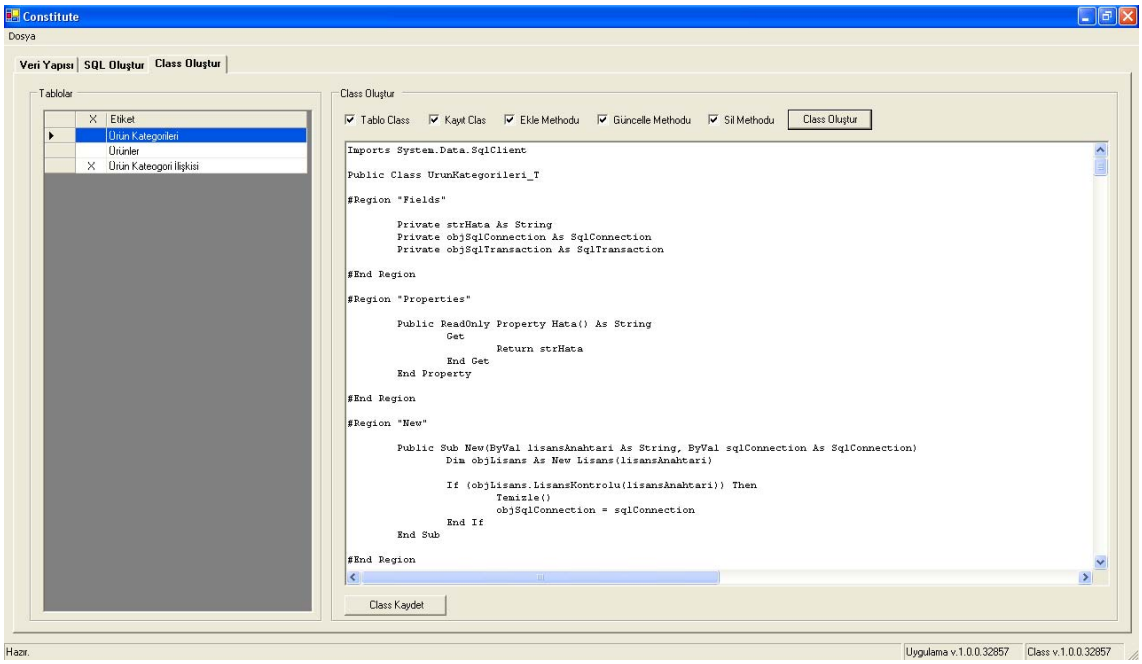
  - **Güncelle Metodu:** Tabloda varolan bir verinin güncellenmesi için kullanılacak sınıf kodlarının üretilmesi işlemini gerçekleştirir.

  - **Sil Metodu:** Tabloda bulunan bir verinin silinmesi için kullanılacak class kodlarının üretilmesi işlemini gerçekleştirir.

.NET Platformu kullanılarak geliştirilen tüm uygulamalarda Microsoft Ara Dili (MSIL) kullanıldığı için uygulama geliştirecek olan kişinin tercih ettiği yazılım dili sınıf kodlarının kullanımı açısından önemli değildir. Class Oluştur bölümünde üretilen tüm sınıf kodları .NET ortamında geliştirilecek tüm uygulamalarda kütüphane olarak projeye eklenerek rahatlıkla kullanılabilmektedir. Bu bölümde oluşturulan sınıf yapıları için [13]' de belirtilen kaynaktan faydalanılmıştır.

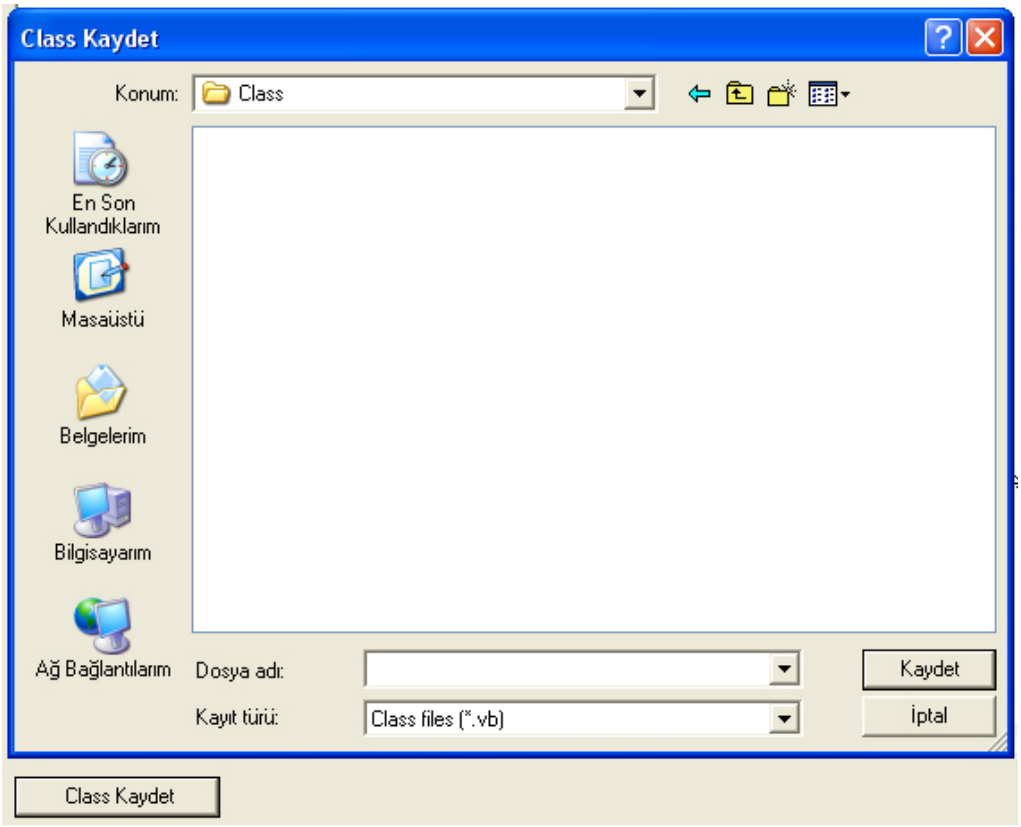
5.3.1 Sınıf Oluştur Örnek Proje Tasarımı

Bu bölümde SQL Oluştur bölümünde tasarımına başlanan Örnek Proje' ye devam edilmiştir. Projede oluşturulmuş tabloların sınıf kodları ve bu kodların içerikleri ile ilgili bilgiler bulunmaktadır. Ürün Kategorileri Tablosu için üretilmesi istenen sınıflar seçilirek Class Oluştur denilirse Şekil 5.19' da verilen ekran görüntüsü elde edilir.



Şekil 5.19 Ürün Kategorileri Tablosu Class Oluştur Ekran Görünümü

Üretilen sınıf kodlarının geliştirilecek projelerde kolay kullanımının sağlanması için yazılım içerisine “Class Kaydet” özelliği eklenmiştir. Şekil 5.19’da görülen Class oluştur bölümünde, pencerenin alt kısmında bu özelliğin kullanımı için Class Kaydet adıyla bir buton bulunmaktadır. Bu butona tıklanıldığında sınıf bilgisinin “.vb” uzantılı olarak kaydedilmesi işlemine olanak sunan yeni bir pencere açılır. Bu pencerede kaydetme işleminin yapılması istenen lokasyon bilgisi ve isim verilerek kayıt işlemi tamamlanır. Şekil 5.20’ de bu pencenin ekran görünümü verilmiştir.



Şekil 5.20 Class Kaydet penceresi ekran görünümü

5.3.1.1 Ürün Kategorileri Tablosu

Bu bölümde projede oluşturulan tabloların sınıf kodlarının daha iyi anlaşılabilmesi için Ürün Kategorileri Tablosu için üretilen sınıf kodları incelenmiştir.

- **Tablo Class**

Tablo sınıf yapısında, ilk olarak veri katmanına erişimi sağlayacak veritabanı bilgileri ve hata kontrolleri yapılır. Aşağıda tablo sınıfında bu işlemlerin gerçekleşeceği alan tanımlamalarının yapıldığı kod bloğu verilmiştir.

```
Imports System.Data.SqlClient

Public Class UrunKategorileri_T

#Region "Fields"

    Private strHata As String
    Private objSqlConnection As SqlConnection
    Private objSqlTransaction As SqlTransaction

#End Region


#Region "Properties"

    Public ReadOnly Property Hata() As String
            Get
                    Return strHata
            End Get
    End Property

#End Region
```

Veritabanı ile hatasız bir bağlantı kurulduktan sonra tablo sınıfı içerisinde tanımlanan yükle metodu yardımı ile seçilen tablonun bilgilerinin çekilmesi işlemi yapılır. Bu bilgiler SQL oluştur bölümünde yine belirtilen tablo için hazırlanmış olan saklı yordamlar ile iletişime geçilerek sağlanır. Aşağıda bu işlemlerin gerçekleştirildiği yükle metodunun bir kısmını oluşturan kod bloğu verilmiştir.

```
Try
                                Dim cmdSelect As New SqlCommand
                                cmdSelect.Connection = objSqlConnection
                                cmdSelect.Transaction = objSqlTransaction
                                cmdSelect.CommandType =
CommandType.StoredProcedure
                                cmdSelect.CommandText =
"spsUrunKategorileri"

                                Dim paramKolonlar As New
SqlParameter("@Kolonlar", SqlDbType.NText)
                                paramKolonlar.Value = kolonlar
                                cmdSelect.Parameters.Add(paramKolonlar)

                                Dim paramKosullar As New
SqlParameter("@Kosullar", SqlDbType.NText)
                                If ((IsNothing(kosullar)) OrElse (kosullar = ""))
Then
                                        paramKosullar.Value = DBNull.Value
                                Else
                                        paramKosullar.Value = kosullar
                                End If
                                cmdSelect.Parameters.Add(paramKosullar)

                                Dim paramParametreler As New
SqlParameter("@Parametreler", SqlDbType.NText)
                                If ((IsNothing(parametreler)) OrElse
(parametreler = "")) Then
                                        paramParametreler.Value =
DBNull.Value
                                Else
                                        paramParametreler.Value = parametreler
                                End If
                                cmdSelect.Parameters.Add(paramParametreler)

                                Dim adpSelect As New SqlDataAdapter
                                adpSelect.SelectCommand = cmdSelect
                                Dim dt As New DataTable
                                adpSelect.Fill(dt)
```

Veri erişim katmanı olarak hazırlanan tablo sınıfının kullanımında hatalı bir işlem yapılırsa, uygulama geliştiren kişi bir hata mesajı ile uyarılır. Bu hata mesajının belirlendiği kod bloğu aşağıdaki gibidir.

```
            Catch ex As Exception
                    strHata = "Ürün Kategorileri Tablosu Yüklenemedi!" &
Chr(13) & Chr(10) & Chr(13) & Chr(10) & ex.Message
                    Throw New OrnekProjeException(strHata)
            End Try
```

- **Kayıt Class**

Bu sınıf veritabanında belirtilen tablo üzerinde veri ile ilgili bir iş yapılmak istenirse erişim için kullanılır. Daha önce "SQL Oluştur" bölümünde oluşturulan saklı yordamlar ile bağlantıyı kurar. Belirtilen tablonun içeriğinin alan ve özellik bilgilerini oluşturur. Kullanılacak Ekle, Güncelle ve Sil metotlarının ön tanımlamalarını içerir. Dikkat edilmesi gereken en önemli husus, veri üzerinde yapılacak çalışmada kullanıcı tarafından yapılabilinecek olası hataların belirlenen özellik tanımlamalarında yapılan kontroller ile önüne geçilmesidir. Bu tanımlamalara Ürün Kategorileri Tablosunda bulunan "Adi" kolonun özellik bilgisinin verildiği aşağıdaki kod bloğu ile örnek verebiliriz.

```
 If ((IsNothing(Value)) OrElse (IsDBNull(Value))) Then
            strHata = "Adı için değer girilmesi zorunludur!"
Throw New OrnekProjeException(strHata)
            ElseIf (CStr(Value).Length > 64) Then
                strHata = "Adı için girilen değer en fazla 64 karakter     olabilir!"
                    Throw New OrnekProjeException(strHata)
            ElseIf (Not (CStr(Value).Length > 0)) Then
      strHata = "Adı için değer girilmesi zorunludur!"
           Throw New OrnekProjeException(strHata)
            Else
                    paramAdi.Value = Value
            End If
```

- **Ekle Metodu:**

Kayıt sınıfında yapılan tanımlamalardan sonra SQL oluştur bölümünde meydana getirilen "ekle" saklı yordamıyla bağlantı kurulur. Bağlantı sonrasında veri girişi her kolon için kullanılan ayrı parametreler üzerinden aktarılarak yapılmış olur. Aşağıda bu parametlerin metot içerisindeki kullanımını gösteren kod bloğu verilmiştir.

```
cmdEkle.Parameters.Add(paramFkUrunKategorisiID)
cmdEkle.Parameters.Add(paramDurum)
cmdEkle.Parameters.Add(paramKodu)
cmdEkle.Parameters.Add(paramAdi)
  If (IsNothing(paramAciklama.Value)) Then
          paramAciklama.Value = DBNull.Value
  End If
cmdEkle.Parameters.Add(paramAciklama)
cmdEkle.Parameters.Add(paramKosul)
              Dim intEtkilenenKayitSayisi As Integer =
cmdEkle.ExecuteNonQuery()
cmdEkle.Parameters.Clear()
```

- **Güncelle Methodu**

Çalışma mantığı ekle metodu gibidir. SQL oluştur bölümünde oluşturulan güncelle saklı yordamıyla bağlantı kurulur. Bağlantı sonrasında veri güncelleme işlemi parametreler üzerinden aktarılarak yapılmış olunur. Kod olarak tanımlanması ekle metodu ile benzer yapıdadır.

- **Sil Methodu**

"Ekle" ve "Güncelle" metoduna benzer yapıda oluşturulmuştur. "Sil" saklı yordamıyla bağlantı kurularak veritabanında silme işleminin yapılması sağlanmaktadır. Kod olarak tanımlanması "ekle" metodu ile benzer yapıdadır.

6. SONUÇ

Bu çalışmada çok katmanlı mimari yapı kullanılacak sistemler için veri ve veri erişim katmanlarını üretecek uygulama geliştirilmiştir. Hazırlanan uygulamanın birçok yararı bulunmakta olup, bunlar aşağıda sıralanmaktadır:

* Veritabanı uygulamalarında yapılan bağlantıların ve veri ulaşım katmanlarının yazılım tarafından oluşturulması ile olası hataların engellendiği gözlenmiştir.
* Yazılım uygulamaları geliştirilirken yapılan çalışmaların, belirli bir sisteme dayandırılması sağlanmıştır.
* Oluşturulacak programların yapısında bulunan alt programların, modüllerin ve objelerin belirli standartlarda üretilmesine olanak sağlanmıştır.
* Tekrarlanılarak kullanılan kod yapılarının istenilen özellikler doğrultusunda üretilmesi ile yazılım oluşturulurken meydana gelebilecek hataların önlenmesi sağlanmış ve zaman kaybının önüne geçilmiştir.
* Geliştirilen yazılımın kullanılması ile belli bir yapıda üretilen kodlar, yazılım projelerinde görev alan tüm kişiler tarafından rahatlıkla anlaşılabilir hale getirilmiştir.
* Olası eleman değişimlerinde, ayrılanın ürettikleri diğer çalışanlar tarafından rahat anlaşıldığından zaman kaybının önüne geçilmiştir.
* Üretimin herhangi bir aşamasında projeye katılan kişilerin projeye uyum sağlama sürecinin hızlanması sağlanmıştır.
* Yapılan uygulamalarda değişiklik veya geliştirilme istenmesi durumunda, tüm yazılımın yeniden oluşturulması yerine, lüzum görülen bölümün düzenlenmesi sağlanmıştır.

Bu çalışma daha önce yapılan çalışma [14] ile kıyaslandığında veritabanına erişimi sağlayan Veri erişim katmanına ek olarak veritabanının oluşturulmasını sağlayan veri katmanının da üretilmesine imkan sağlanmıştır. Veri katmanının eklenmesi, katmanlı mimarinin doğuş nedeni olan farklı bileşenlerin birbirinden ayrılarak kullanılması ilkesine daha uygun bir yapı sunmaktadır. Veri katmanının üretimi, veritabanı oluşturulurken yaratılan isimlerde Türkçe karakterlerin ve boşlukların kullanılmasından

doğacak hataların önüne geçmekte, veri girişlerinde belirlenecek kriterler sayesinde istenmeyen değer girişlerini engellemektedir. Ayrıca en önemlisi veri katmanını oluşturan saklı yordamların kullanımının kullanıcı bazlı yetkilendirilmesi ile veri güvenliği sağlanmaktadır.

Hazırlanan çalışmanın olumlu yönleri göz önünde bulundurulursa, .NET Platformunda uygulama geliştiren ve SQL veritabanı sistemini kullanan yazılımcılar için uygun bir çözüm olduğu görülmektedir.

Gelecekte benzer konuda çalışma yapmak istenirse, çalışma kapsamında ele alınmayan verinin düzenlenmesi ve temizlenmesi işlemleri geliştirilen uygulamaya eklenebilir. Ayrıca SQL haricindeki diğer veritabanı sistemleri için de uygulamaya yeni özellikler katılabilir.

KAYNAKLAR

[1] Sarıdoğan, M.Erhan, (2004), Yazılım Mühendisliği, Papatya Yayıncılık Eğitim, İstanbul

[2] http://www.microsoft.com/turkiye/

[3] http://msdn.microsoft.com/en-us/library/0b0thckt.aspx

[4] İnan, M. Yüksel ve Demirli, Nihat, (2004), Zirvedeki Beyinler – 12, Visual.Basic.NET 2003, Prestige Education Center Yayınları, Ankara

[5] http://www.bidb.itu.edu.tr/?d=496

[6] Chartier, Robert, "Application Architecture: An N-Tier Approach - Part 1", http://www.15seconds.com/issue/011023.htm

[7] Demirli, Nihat ve İnan, M. Yüksel, (2006), Palme Zirvedeki Beyinler – 31, Visual C#.NET 2005, Palme Yayıncılık, Ankara

[8] http://msdn.microsoft.com/en-us/library/8bs2ecf4(VS.71).aspx

[9] Müezzinoğlu, Alev, (2004), Zeybeklerin "SQL" sorgulama analizi, Gazi Üniversitesi Eğitim Bilimleri Enstitüsü, Yüksek Lisans tezi, 166 sayfa, Ankara

[10] http://www.netogretim.com/dokumangoster.aspx?id=46&d=SQL-Nedir?

[11] http://msdn.microsoft.com/en-us/library/aa258271(SQL.80).aspx

[12] http://msdn.microsoft.com/en-us/library/ms191240.aspx

[13] http://msdn.microsoft.com/en-us/library/aa581778.aspx

[14] Küçükeren, Gökçe, (2006), Veri erişim katmanı kod üretici, Işık Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans tezi, 88 sayfa, İstanbul

ÖZGEÇMİŞ

Ad Soyad	Ersin Artan
Doğum Tarihi	25.05.1983
Doğum Yeri	İstanbul
Lise	1994 – 2001 Edirne Anadolu Lisesi
Üniversite	2001 – 2005 Marmara Üniversitesi T.E.F Bilgisayar ve Kontrol Öğretmenliği.
Staj Yaptığı Yerler	ROBOTICS / PALMTURK – İstanbul – (24 iş günü) SÜZER HOLDING – İstanbul - (18 iş günü) TURKENT A.Ş. – İstanbul – (15 iş günü)
Çalıştığı Yerler	Satko Teknoloji Sistemleri Sanayi ve Ticaret A.Ş. – İstanbul – (1 yıl) Belgesistem Ltd. – İstanbul – (4 ay) Ofisteknik A.Ş. – İstanbul – (9 ay – devam ediyor)