PROJE TASARIMI VE YÖNETİMİ İÇİN VERİTABANI SİSTEMİ

Nefize ALTUN

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Haziran – 2004

JÜRİ VE ENSTİTÜ ONAYI

Nefize ALTUN'un **Proje Tasarımı ve Yönetimi için Veritabanı Sistemi** başlıklı **Bilgisayar Mühendisliği** Anabilim Dalındaki, Yüksek Lisans tezi 28 Haziran 2004 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	İmza
Üye (Tez Danışmanı)	**: Doç.Dr.AHMET BABANLI**	
Üye	**: Prof.Dr.ALİ GÜNEŞ**	
Üye	**: Y.Doç.Dr.C.HAKAN KAĞNICIOĞLU**	

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 26.08.2004... **tarih ve** ..28/3... **sayılı kararıyla onaylanmıştır.**

Enstitü Müdürü

Prof. Dr. Altuğ İFTAR
Fen Bilimleri Enstitüsü
Müdürü

ÖZET

Yüksek Lisans Tezi

YAZILIM PROJELERİ VERİTABANI YÖNETİM SİSTEMİ

NEFİZE ALTUN

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç.Dr.Ahmet BABANLI
2004, 107 sayfa

Günümüz yazılım projelerini, geçmişte yapılan yazılım projelerinden ayıran en önemli özellik, yazılım üretiminin vazgeçilmez bir parçası haline gelen yazılım standartlarının uygulanmasıdır. Bu tez çalışmasında yazılım kalite standartlarının oluşma nedenleri, tarihçeleri incelenmiş ve günümüz koşullarında etkin bir kalite denetimini sağlayan IEEE/EIA 12207'nin içeriği ele alınarak bu yazılım yaşam döngüsü süreçlerine uygun çalışan bir kurumun çalışmalarının daha iyi denetlenmesini sağlamak amacı ile bir program hazırlanmıştır. Bu program ile IEEE/EIA 12207 yazılım yaşam döngüsü süreçlerine uygun proje yönetiminin sağlanması amaçlanmıştır.

Program, Oracle veritabanı yönetim sistemi ile hazırlanmış bir veritabanından ve bu veritabanına erişim sağlayan Visual Basic 6.0 ile hazırlanmış bir arayüz programından oluşmaktadır. Programın hazırlanması sırasında veritabanına erişim yöntemleri incelenmiş, veriye erişim hızının arttırılması amacı ile izlenebilecek yöntemler araştırılmıştır. Yapılan çalışmalar sonucunda, prosedürel yapıları SQL ifadeleriyle birleştiren bir dil olan PL/SQL ile depolanmış prosedürler hazırlanmıştır.

Anahtar Kelimeler: IEEE/EIA 12207 Yazılım Yaşam Döngüsü Süreçleri, Yazılım Geliştirme Standardı, Veritabanı Yönetimi, Konfigürasyon Yönetimi, Kullanıcı Arayüzü.

ABSTRACT

Master of Science Thesis

SOFTWARE PROJECTS DATABASE MANAGEMENT SYSTEM

NEFİZE ALTUN

Anadolu University
Graduate School of Natural and Applied Sciences
Computer Engineering Program

Supervisor: Assoc.Prof.Dr. Ahmet BABANLI
2004, 107 pages

The most distinctive feature of the contemporary software projects from the historical ones is that of applying the software standards, which are considerable piece of software development. The formation and the history of the software quality standards are examined in this thesis and IEEE/EIA 12207 content is handled, which obtain an effective quality control under current conditions. A software program is prepared by using IEEE/EIA 12207 content, intending to inspect an association's work and project management convenient with the software life cycle processes.

The program is consist of a database prepared by Oracle Database Management System and a user interface program prepared by Visual Basic 6.0 that obtain database access. The database access methods are examined and the data access rate increasing methods are investigated during the program preparation. Finally store procedures were prepared by PL/SQL, which is a database programming language that combines procedures with SQL structures.

Keywords: IEEE/EIA 12207 Software Life Cycle Processes, Software Development Standard, Database Management, Configuration Management, User Interface.

ÖNSÖZ VE TEŞEKKÜR

İlk yazılım geliştirme standartlarının askeri yazılım projelerinin güvenliğinin ve kalitesinin arttırılması amacı ile ortaya çıktığı bilinmektedir. Standartların geliştirilmesi sırasında oraya çıkan ilk kanı, standartların kesin ve katı kurallardan oluşturulması gerekliliğiydi. Bir standardın kuralları ne kadar kesin olursa standart dışına çıkma olasılığı o kadar azalır ve yazılımın kalitesi ve güvenliği arttırılmış olurdu. Ancak standartlar da çeşitli geliştirme aşamalarından geçmiş ve zamanla bu kanının yanlış olduğu ortaya çıkmıştır. Çünkü bir standardın katı kurallar içermesi, o standardı kullanan kuruma, gittikçe altından kalkamayacağı ağır yükler getirmektedir.

Günümüzde ise yazılım üreten kurumlar kendi yapılarına en uygun standardı seçmekte ve seçtiği bu standardı kendi yapısına uyarlayarak çalışma usüllerini belirlemektedirler. Bu sayede kendilerini gereksiz bir iş yükünün altına sokmadan, ürettikleri yazılımın kalite standartlarına uygun olmasını sağlamaktadırlar.

Bu tezin anahtar kelimelerinden biri olan IEEE/EIA 12207 bir yazılım geliştirme standardı değil, kurumların kendi standartlarını geliştirmesine imkan veren yazılım yaşam döngüsü süreçlerini ve faaliyetlerini anlatır.

Hazırlanan Yazılım Projeleri Takip Programı(YPTP), kendine IEEE/EIA 12207 yazılım yaşam döngüsü süreçlerini baz alarak çalışma usüllerini oluşturan bir kurumun proje yönetimini sağlamak amacı ile geliştirilmiştir.

Bu tez çalışması süresince beni yönlendirerek yardımlarını esirgemeyen danışmanım Sayın Doç.Dr. Ahmet BABANLI'ya, tecrübe ve bilgilerinden yüksek oranda faydalandığım bölüm başkanım Sayın Prof.Dr. Ali GÜNEŞ'e, tez konusu ve ihtiyaçların belirlenmesinde bana yardımcı olan Sayın Hv.Müh.Yzb. Tamer EREN'e teşekkür ederim.

Ayrıca destek ve anlayışıyla sürekli yanımda olan eşim Özgür'e ve anneme teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

CSC : Computer Software Component

CSCI : Computer Software Configuration Item

CSU : Computer Software Unit

DBMS : Database Management System

DDL : Data Definition Language

DML : Data Manipulation Language

HTTP : Hypertext Transfer Protocol

IEEE/EIA : The Institute of Electrical And Electronics Engineers/ Electronic Industries Association Engineering Department

LOB : Large Object

NF : Normal Form

PL/SQL : Procedural Language Extensions to Structured Query Language

SGA : System Global Area

UML : Unified Modeling Language

UTL : Utility

VTYS : Veritabanı Yönetim Sistemi

YPTP : Yazılım Projeleri Takip Programı

1. GİRİŞ

Dünyanın en önde sektörlerinden biri haline gelen yazılım sektöründe artık amaç; müşteri ihtiyaçlarını tam olarak karşılayan, eksiksiz, hatasız yazılım ürünleri ortaya çıkarabilmek olmuştur. Tecrübeler yazılım sektöründe yapılan hataların ciddi zaman ve maliyet kaybına neden olduğunu göstermektedir. ABD'nde 1970'li yıllarda kamu tarafından sipariş yolu ile satın alınan yazılım ürünlerinin ancak %5'i kabul edilip kullanılabilmiştir. Yaşanan yazılım krizi 1980'li yıllarda artmış ve 1990'lı yıllara kadar da büyük projelerde başarı oranı ve kalite düşük düzeylerde seyretmiştir. Elde edilen tecrübeler ve çalışmalar sonucunda, yazılım kalitesini sağlamak amacı ile çeşitli yazılım kalite standartları ortaya çıkmıştır. Yazılım ciddi bir iştir ve başından sonuna kadar kontrol altında tutulması gerekmektedir. Standartlar yazılım kalite güvenliğini sağlayarak bunu yapmaktadırlar. Artık büyük yazılım kuruluşları ve müşteriler bu konuda bilinçlenmişlerdir. Ürünün standartlara uygunluğu konusunda daha titiz davranmaktadırlar.

Özellikle 90'lı yıllarda önemi giderek artan "Yazılım Kalite Sağlama" etkinlikleri günümüzde yazılım üretiminin vazgeçilmez bir parçası olmuştur. Artık kalite unsuları içermeyen yazılım üretiminden söz edilememektedir. En yakın anlatım biçimi ile Yazılım Kalite Sağlama üretim süreci boyunca ara ürünlere ilişkin kalite standartları geliştirmek ve geliştirmenin bu standartlara uygunluğunun denetlenmesi olarak tanımlanır. Kalite sağlama ayrıca sonuç ürünün belirlenen kalite kriterlerine uygunluğunun sağlanması amacını da taşır.

Standartlar; kalite sürecinin gerçeklenmesinde temel oluşturur. Standartlarda yazılım yaşam döngüsü süreçleri için ortak bir çerçeve tanımlanır. Çalışmaların rolleri ve arayüzleri olduğu gibi dokümantasyon tipleri ve içerikleri de açıklığa kavuşturulur. Gereken görevleri, adımları, dayanakları ve gözden geçirmeleri belirler. Geçmişteki problemlerden ve farkedilmeyen tehlikelerden uzaklıştırdığı gibi büyük bir yazılım ekibinin çalıştığı projelerde sistem ve yazılım mühendislerinin bir bütün halinde çalışmasını sağlar. Projenin takibi kolaylaşır hatayı zamanında bulma olasılığı artar ve müşteri isteklerini tamamiyle karşılayacak bir ürünün ortaya çıkması sağlanmış olur. Projede çalışan ekibin değişmesi durumda aksaklıkların ortaya çıkmasını engeller.

Askeri projelerde yazılımın güvenirliliğinin daha üst seviyede olmasından dolayı ilk geliştirilen standartlar askeri yazılım ürünleri için ortaya çıkmıştır. Zamanla standartlar da gelişme aşamalarından geçmiş ve yaşanan tecrübeler ile iyileştirilmiştir. Önceleri daha katı kurallardan oluşan standartlar zamanla yerini şirket yapısına, amacına uygun olarak biçimlendirilebilecek bir içeriğe ulaşmıştır.

Artık günümüzde yazılım şirketleri kendine en uygun standardı seçmekte ve onu kendi yapısına uyarlayarak kendi çalışma usüllerini belirlemektedir. Burada önemli olan standardın her bir bileşenini körükörüne almak yerine şirketin kendi ihtiyaçlarını karşılayacak şekilde bu standartların uyarlanmasıdır. Aksi taktirde yazılım standartları projenin ilerlemesini yavaşlatan, atlanması zor engeller haline gelir. [2]

Her yazılım birimine uyan hazır tablet şeklinde bir standart yada prosedür seti yoktur. Her organizasyonun kendi ihtiyaçları ile buluşacak şekilde, kullanmayı düşündüğü standardı kendine uyarlaması gereklidir. Şayet herhangi bir süreç değerlendirme tekniği kullanılırsa iyileştirme fırsatları listesi tanımlanmış olur. Bu alanlardan her birindeki görevlerin nasıl yerine getirileceğini dokümante edilerek kusurlar iyileştirilmeye başlanabilir. Dikkatlice seçilir ve zekice uygulanırsa yazılım prosedürleri ve standartları organizasyonun yazılım geliştirme ve bakım etkinliklerini sağlayarak çalışmaları kolaylaştırır.

Kurumlar seçtikleri ve kendilerine uyarladıkları yazılım kalite standartlarını daha kolay uygulayabilmek ve kendi içlerinde bunu denetleyebilmek için zamanla yardımcı programlara ihtiyaç duymuşlardır. Bu programlar proje yönetimini sağladığı gibi üretilen dokümanların, yapılan toplantıların da yönetimini yapmaktadırlar. Başlarda yazılım konfigürasyon yönetimini sağlayan bu tip programlar tek bir yazılım kalite standartının şartlarını sağlayacak şekilde hazırlanmışlardı. Bu tip programlar sistem ve yazılım ihtiyaçlarını, sistem ve yazılım testlerini ve de dokümantasyonu yöneterek bir grup halinde çalışan proje mühendislerinin arasında daha iyi bir iletişim sağlamış olur.

IEEE/EIA 12207'de son yıllarda oluşturulmuş herhangi bir yazılım modeline, tasarım yada programlama diline bağlı olmayan ve belgelemeyi kolaylaştıran bir yazılım yaşam döngüsü sürecidir. Bu çalışmada "IEEE/EIA 12207 Yazılım

Yaşam Döngüsüne" uygun olarak çalışarak yazılım üreten kişi yada şirketlerin, yazılım projelerinin takibinin yapılması amaçlanmaktadır.

"Yazılım Projeleri Takip Programı(YPTP)" olarak adlandırılan bu çalışmanın amacı; IEEE/EIA 12207 yazılım yaşam döngüsü süreçlerine uygun olarak gerçekleştirilmekte olan yazılım projelerinin takibini sağlamak ve geliştirilmekte olan yazılım projelerinin standarda uygunluğunu kontrol etmektir. YPTP; yazılım projelerindeki ihtiyaçları, testleri, tasarım spesifikasyonlarını ve bunlar arasındaki izlenebilirlik ve arabirimleri yöneten bir veritabanı programıdır. Ayrıca bu program mühendislik grubu içindeki ortaklaşa yürütülen mühendislik geliştirme çalışmalarını desteklemek amacı ile tasarlanmıştır. Böylelikle proje mühendisleri arasında daha iyi bir iletişim sağlanmış olur. YPTP programı yetkilendirilmiş tüm kullanıcılar için ulaşılabilirdir ve kullanıcıların son gelişmelere göre güncelleştirilmiş bilgiye erişimini sağlar. İhtiyaçların; geliştirmenin tüm aşamaları içinde tamamlanma safhalarını izleyerek yönetimini temin eder. Sistem ihtiyaçları; yazılım mühendisliği, donanım mühendisliği ve geliştirme disiplinlerine yönlendirilir. Yazılım ve sistem test yönetimi icra edilir. Bilgi tamamlanması ve yönetimi ile birlikte arabirim yönetimi gerçekleştirilir.

Bu tez gerçekleştirilirken veritabanı yönetim sistemi olarak Oracle8i ve arayüz programını oluşturmak için de Microsoft Visual Basic 6.0 programlama dili ve PL/SQL depolanmış prosedürleri kullanılmıştır.

Geliştirilen uygulamanın amacının ve kullanımının daha iyi anlaşılabilmesi için Bölüm 2'de problemin tanımı yapılmış, Bölüm 3'de IEEE/EIA 12207 Yazılım Yaşam Döngüsü Süreçleri hakkında detaylı bilgi verilmiştir.

YPTP programı tasarlanırken, kullanılabilecek veritabanı yönetim sistemleri üzerinde araştırmalar yapılmış ve Oracle veritabanı yönetim sisteminin kullanılmasına karar verilmiştir. Oracle veritabanı yönetim sistemi kullanılarak veritabanı tasarımı ve gerçekleştirimi yapılırken birden fazla kişinin aynı anda ve güvenli bir şekilde sadece yetkilendirildiği alanda çalışması sağlanabilir. Oracle VTYS, hazırlanan veritabanının ve arayüz programının modüler olmasına imkan verir. Geliştirilerek kullanımı düşünülen YPTP programı için bu özellikler birer avantajdır. Ayrıca Oracle veritabanı yönetim sisteminin sağladığı PL/SQL dili ile veri erişim hızı ve verinin güvenliği sağlanabilir.

Geliştirilen uygulamanın programlama tekniğinin anlaşılabilmesi için kullanılan Oracle veritabanı yapısı ve veritabanı programlama konularının bilinmesi gerekir. Bu nedenle bölüm 4'de ise Oracle veritabanının genel yapısı anlatılmış, veritabanı sistem mimarileri hakkında bilgi verilmiştir. Ayrıca ilişkisel veritabanlarında standart bir programlama dili olan SQL ve Oracle tarafından geliştirilen PL/SQL hakkında bilgi verilmiştir.

"Yazılım Projeleri Takip Programı" olarak isimlendirilen uygulamanın yapısı, tabloları, bu tabloların birbirleri ile ilişkileri, kullanılan arayüzler ve veriye erişim yolları Bölüm 5'de anlatılmış ve uygulama programın özellikleri, geliştirme safhası sırasında karşılaşılan problemlere bu bölümde yer verilmiştir.

Bölüm 6 ise uygulama programının sağladığı yararlar, kullanım alanları ve geliştirilmeye açık olup olmadığı konusuna ayrılmıştır.

2. PROBLEMİN TANIMLANMASI

Yazılım sektöründe artık amaç; müşteri ihtiyaçlarını tam olarak karşılayan, eksiksiz, hatasız yazılım ürünleri ortaya çıkarabilmek ve bunu yaparken de zaman ve maliyet kaybını en aza indirmektir. Bunu yapmak için yaşanan tecrübeler ve çalışmalar sonucunda yazılım geliştirme standartları ortaya çıkmıştır. Yazılım ciddi bir iştir ve başından sonuna kadar kontrol altında tutulması gerekmektedir. Bu amaçla bir yazılım standardını kendisine baz alarak kendi çalışma usullerini belirleyen ve yazılım üreten organizasyonlar bazı yardımcı alt programlara ihtiyaç duyarlar. Yardımcı program olmadan gerçekleştirilen projelerde, proje çalışanlarının iş yükü ve hatta projede çalışacak birey sayısı artmakta, proje elemanlarının proje ile ilgili güncel bilgilere erişimi zorlaşmaktadır. Bu tez çalışması da IEEE/EIA 12207 yazılım yaşam döngüsü süreçlerini kendisine baz alarak kendi çalışma usullerini belirlemiş organizasyonların proje takiplerini sağlamak amaçlı oluşturulmuş bir yardımcı programdır.

Bir yazılım projesinde etkin olarak çalışılabilmesi, oluşturulacak proje ekip yapısına büyük ölçüde bağlıdır. Ekip yapısı, projedeki iş yükünün işlevsel olarak tanımlanması ve projede çalışan kişiler arasında etkin iletişim oluşturulması amacıyla yapılır. Hemen hemen tüm proje yönetim metodojileri ekip yapısını önerir.

Proje ekip yapıları, hem projeyi yapan kuruluş hem de proje sahibi kuruluş tarafından oluşturulmalıdır. Yüklenici tarafında oluşturulacak yapı; temel olarak projenin geliştirilmesine, iş sahibi tarafında oluşturulacak yapı ise projenin koordinasyonuna yönelik olmalıdır.

Kurumlar kendilerine uyarladıkları yazılım kalite standartlarını herhangi bir yardımcı yazılım programı olmadan da uygulayabilirler; ancak bu, proje elemanlarının birbiri ile olan iletişimlerini geciktirerek zaman kaybına yol açabileceği gibi daha fazla hatanın yapılmasına ve projedeki risklerin artmasına yol açabilir. Projenin yaşam döngüsü boyunca ortaya çıkabilecek hatalar ve bu hataların düzeltilmesi yada yapılması gereken değişiklikler üzerindeki izlenilebilirlik azalabilir. Projeye yeni katılan bir kişinin projenin kendisinden önceki basamaklarını izleyip projeye adapte olması zorlaşabilir. Tüm bu nedenlerden dolayı artık günümüzde kurumlar kendilerine uyarladıkları yazılım

kalite standartlarını daha kolay uygulayabilmek ve gerçekleştirdikleri projelerin bu standarda uygun ilerleyip ilerlemediğini kontrol edebilmek için yardımcı programlar kullanmaktadırlar.

Yazılım Kalite Standartlarının uygulanmasını kolaylaştıran araç programların çoğu sadece bir standardı kendisine taban olarak almışlardır. Yazılım Projeleri Takip Programı; IEEE/EIA 12207'yi taban alıp kendisine uyarlayarak kendi çalışma usullerini hazırlamış bir şirketin, iş disiplinlerini denetleyecek, proje elemanlarının aynı dilden konuşarak projenin bakımını sağlayacak, projenin oluşturulan bu disipline uygun ilerleyip ilerlemediğini kontrol edecek bir yazılıma ihtiyaç duymasından kaynaklanmaktadır.

"Yazılım Projeleri Takip Programı", IEEE/EIA 12207 yazılım yaşam döngüsü süreçlerini kendisine uyarlamış bir kurumun belirlediği bu çalışma usullerini daha kolay yerine getirebilmesi için hazırlanmıştır. Bu nedenle IEEE/EIA 12207 standardının, bu amaçla hazırlanmış diğer araç yazılımların detaylı bir şekilde incelenmesi faydalı olacaktır.

3. IEEE/EIA 12207 YAZILIM YAŞAM DÖNGÜSÜ SÜREÇLERİ

IEEE/EIA 12207 belgelemeyi kolaylaştırır. Hangi belgelerin üretilmesi gerektiği ve içeriklerinde neler olacağı tarif edilir. Ancak MIL-STD-498'deki gibi kesin formlar ve katı kurallar yoktur. MIL-STD-498, 1994 yılında uygulanmaya başlanmış ABD Savunma Bakanlığı yazılım geliştirme ve belgeleme standartlarından biridir. MIL-STD-498 projenin yazılım kısmına odaklanmış ve projenin özellikle yazılım geliştirme etkinliklerini belirleyen bir standarttır. Takım çalışmasını destekler, herhangi bir tasarım yada programlama diline bağlı değildir. Ancak bununla birlikte hazılanması gereken dokümanlar ve içerikleri konusunda kesin kurallara sahiptir.[1]

IEEE/EIA 12207 etkin bir takım çalışmasının gerçekleştirilmesini sağlar. Her bir süreç için daha önce kullanılan standartların en iyi yönlerine işaret ederek referans verir, ancak bunların kullanılması için zorlamaz. Herhangi bir yazılım modeline, tasarım ve programlama diline bağlı değildir. Yazılım kalite faktörleri kullanımında esnekliğe sahiptir.

IEEE/EIA 12207'de süreçler birbiri ardınca değil, birbirleriyle içiçe geçmiş bir şekilde yada birbirleriyle paralel olacak şekilde gerçekleştirilir. Bir kurumun standartlaşması esnasında IEEE/EIA 12207'yi temel alması demek, daha önce de değinildiği gibi standardın, kurumun yapısına en uygun şekle dönüştürülmesi demektir. IEEE/EIA 12207'nin amacı da budur.

Katı kuralların olmaması kurumun kendine ait yazılım kalite sistemini kurarken esnek davranabilmesini sağlar. Mesela; proje gözden geçirme toplantılarının sayısı arttırılabilir yada azaltılabilir, üretilmesi zorunlu olan dokümanlar dışındaki diğer dokümanların üretilip üretilmeyeceğine projeye yada müşteri istediğine göre karar verebilebilir. Kurum, organizasyon yapısını kendi belirleyebilir ve kendi kalite güvencesini oluşturabilir. Doküman içerikleri standartta tanımlanmıştır ama içerik hazırlama standardı oluşturacak kuruma bırakılmıştır. Kurum doküman içeriğini, formatını kendisi belirleyebilir.[2,3,4]

3.1. Süreçler

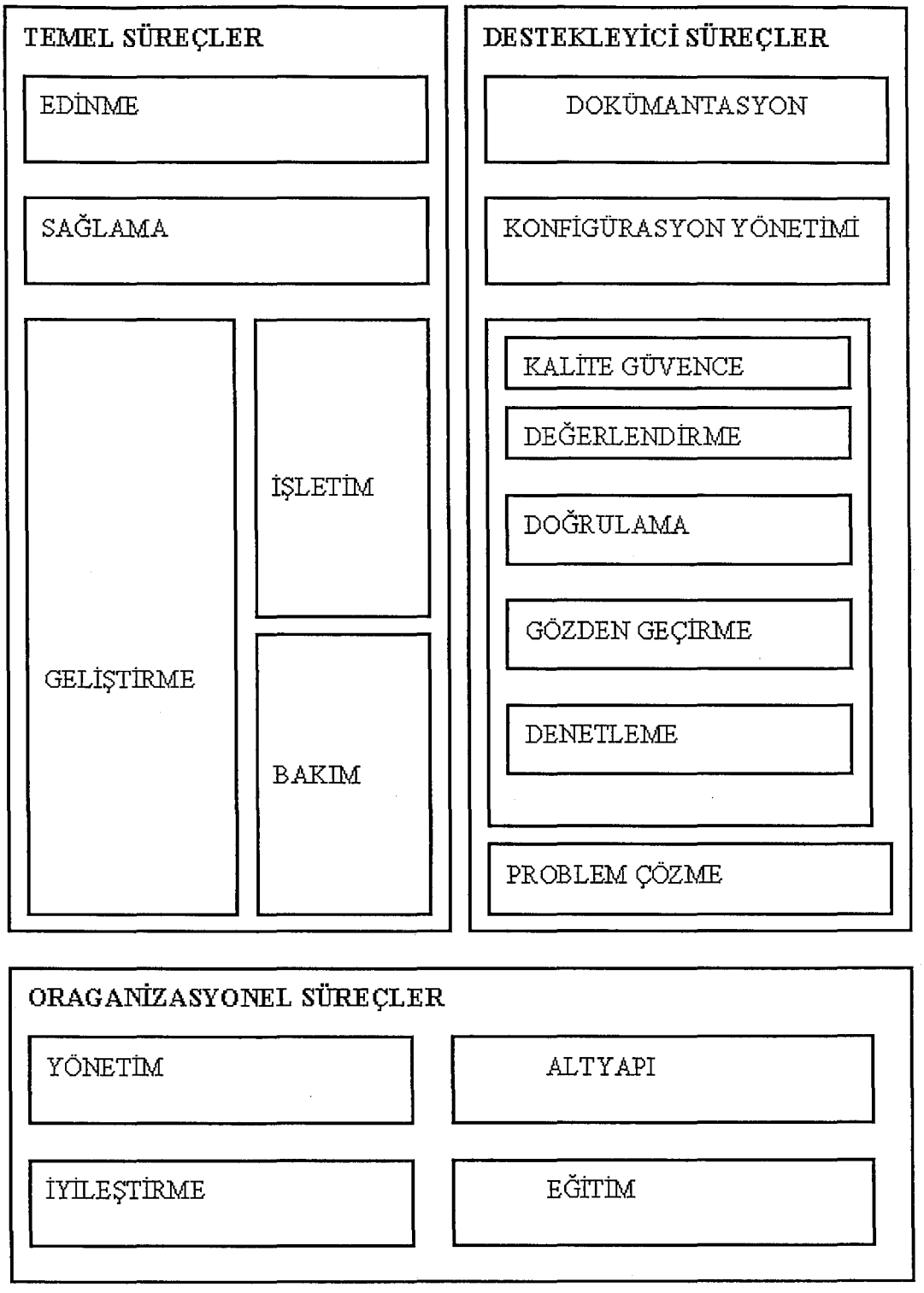
IEEE/EIA 12207'de; her süreç, etkinliklere, her etkinlik, faaliyetlere ayrılmıştır. Görevler bilgi öğelerini kullanırlar. IEEE/EIA 12207'nin genel bilgi

öğeleri; tanımlamalar, planlar, prosedürler, kayıtlar, raporlar, istekler ve spesifikasyonlardır. Yazılım ürünü ve dokümanlar bilgi öğelerinin birer parçasıdır. IEEE/EIA 12207 içerisinde süreçler; temel süreçler, destekleyici süreçler ve organizasyonel süreçlar olarak sınıflandırılmaktadırlar. Bu sınıflandırma Şekil 3.1'de verilmektedir.

Temel Süreçler; yaşam döngüsü boyunca organizasyon elemanlarının ne yaptığını tanımlar. Bunlar Edinme, Sağlama, Geliştirme, İşletim ve İdame süreçleridir.

Destekleyici süreçler; bir başka süreç tarafından kullanılan, çalıştırılan yardımcı süreçlerdir. Bu süreçler ise; dokümantasyon, konfigürasyon yönetimi, kalite güvence, değerlendirme, doğrulama, gözden geçirme, denetleme ve problem çözme süreçleridir.

Organizasyonel yaşam döngüsü süreçleri ise temel süreçlerin gerçekleşmesinde etkili olan süreçlerdir. Bunlar; yönetim, altyapı, iyileştirme ve eğitim süreçleridir. [2,3,4]



Şekil 3.1 IEEE/EIA 12207'de süreçler ve faaliyetler

3.2. Temel Süreçler

3.2.1. Edinme ve sağlama süreci

Edinme(Acquisition) süreci, müşteri aktivitelerini ve görevlerini içerirken sağlama(Supply) süreci de projeyi yapacak organizasyonun aktivite ve görevlerini içerir. Her iki süreçde de gerek kontrat gerekse protokol aracılığı ile yapılacak yazılı anlaşmada tarafların görev ve sorumluluklarının belirlenmesi sağlanır. Böylelikle yazılım üretiminin başlangıcından sonuna kadar her iki tarafın haklarının korunması sağlanmış olur.

IEEE/EIA 12207, müşteri ile işi yapacak organizasyon arasındaki ilişkiyi başlangıçtan yani kontrat aşamasından itibaren kontrol altına alınmasını sağlar. İhtiyaç makamının belirlediği gereksinimlere yönelik olarak aşağıda belirtilen tanımlama ve analizlerin bir kısmı veya hepsi ihtiyaç makamı tarafından yapılır veya talep edildiğinde organizasyon tarafından gerçekleştirilir, müşteri tarafından onaylanır:

- Piyasadan temin edilen ürünle ihtiyacın karşılanması,
- Organizasyon tarafından belirlenen kriterlere göre yeni ürün geliştirilmesi,
- Müşteri tarafından belirlenen kriterlere göre yeni ürün geliştirilmesi,
- Yukarıdaki üç maddenin ortak kullanımıyla ihtiyacın karşılanması,
- Mevcut ürün veya hizmetin geliştirilerek ihtiyacın karşılanması,
- Kullanıcı veya işletme seviyesinde güvenlik,
- Ürünle ilgili kritik tasarım kriterleri,
- Test,
- Uygulanacak standartlar,
- Hazırlanacak dokümanlar vb.

Organizasyon, gerekli görüldüğü taktirde müşteriden sistem ihtiyaçlarını, yazılım ürünleri listesini, tanımları ve şartları, alt kontraktörlerin kontrolünü, teknik özellikleri içeren bir şartname talep edebilir. Daha sonra şartname organizasyon tarafından incelenir, uygun görüldüğünde kontrat/protokol çalışmalarına başlanır. Bu kapsamda müşteri ve organizasyon; ihtiyaçların, proje takviminin, maliyet temini sağlanacak yazılım ürün ve hizmetlerinin, piyasadan

temin edilecek ürünlerin garanti koşulları ve lisans haklarının, ürün kalitesinin, güvenliğinin, kontrat ve ürün değişikliklerinde izlenecek yolun, alt kontraktörlerin kontrol yöntemlerinin, organizasyon tarafından sağlanacak ürün ve hizmetlerin garanti koşullarının, lisans haklarının, ihtiyaç makamına sağlanacak bilgi ve dokümanların, ihtiyaç makamının ulaşabileceği bilgi ve dokümanların, dokümanların içeriği ve formatların, ihtiyaç makamının onayına sunulacak dokümanların, dokümanların hangi safhada hazırlanacağının, gizlilik derecesinin, kabul kriterlerinin, uygulanacak test yöntemleri ve karşılıklı mutabakatın tanımlandığı kontratı hazırlar. Kontrat/protokol imzalandıktan sonra organizasyon tarafından gerekli planlama yapılır. Eğer proje kapsamında bir alt kontraktörle çalışma gereksinimi duyulursa, alt kontraktör seçimi ve yönetimi belirlenir.

Daha sonra projenin özelliğine göre yazılım geliştirme modellerinden biri (Waterfall model, Incremental model, Evolutionary modeli, Reengineering vb.) seçilir ve tüm bunları içeren bir plan oluşturulur. Organizasyon, hazırlanan plana göre projenin uygulamasına geçer. Proje ihtiyaçlarında, işletme politikalarında ve prosedürlerde değişiklik olduğunda bu planda değişiklik yapılır ve değişiklikler ihtiyaç makamına bildirilir. Böylelikle müşteriden habersiz iş yapılması engellenmiş olur. Kontrat aşaması tamamlandıktan sonra geliştirme sürecine göre yazılım ürünü geliştirilir, işletim sürecine göre yazılım ürünü çalıştırılır ve de bakım sürecine göre yazılım ürününün bakımı yapılır. Organizasyon, proje faaliyetlerinin ve ortaya çıkan ürünlerin istenilen kalitede olduğunu, teknik performansın, harcamaların, proje takviminin, problem belirleme, kayıt altına alma, analiz ve çözüm yöntemlerinin kontratta belirtilen süreç çerçevesinde izler ve kontrol eder.

Gözden geçirme ve değerlendirmeler sırasında organizasyon planında belirtildiği gibi kontrat gözden geçirilmesi, ihtiyaç makamı ile yapılacak toplantılar, iletişim, kabul gözden geçirmeleri, kabul testlerinin gerçekleşmesini sağlar. [2,3,4]

3.2.2. Geliştirme süreci

Bundan sonraki adımda IEEE/EIA 12207'nin geliştirme(development) süreci başlar. Geliştirme süreci şu işlem basamaklarından oluşur :

- Sistem İhtiyaçlarının Analizi,
- Sistem Yapısal Tasarım,
- Yazılım İhtiyaçlarının Analizi,
- Yazılım Yapısal Tasarım,
- Yazılım Detay Tasarım,
- Yazılım Kodlama ve Test,
- Yazılım Entegrasyonu,
- Yazılım Yeterlilik Testi,
- Sistem Entegrasyonu,
- Sistem Yeterlilik Testi,
- Yazılım Yükleme,
- Yazılım Kabulü,
- Sistem Kabulü.

Seçilen modele göre işlem basamaklarının sırası değiştirilebilir ve bazı basamaklar atlanabilir. Geliştirme süreci içerisinde, yazılım entegrasyonu ve sonrasında tespit edilen aksaklıklar, problem çözümlenmesi sürecine göre giderilir. Belirlenen bir formata göre yazılım tanıtma numarası verilir.

3.2.2.1. Sistem ihtiyaçları analizi

Bu kapsamda sistemin fonksiyon ve kapasitesinin, organizasyon ve kullanıcı ihtiyaçlarının, güvenlik ve gizlilik ihtiyaçlarının, ergonomik ihtiyaçların, uygulama ve bakım ihtiyaçlarının, tasarım kısıtlamalarının ve yeterlilik ihtiyaçlarının, donanım ihtiyaçlarının belirlenmesi yapılır ve dokümante edilir. Sistem ihtiyaçlarının analizi sırasında; ihtiyaç makamının isteklerinin sistem ihtiyaçları içinde izlenebilirliği, sistem ihtiyaçlarının, mşterinin isteklerine uygunluğu, test edilebilirlilik, sistem yapısal tasarım fizibilitesi, işletim ve bakım fizibilitesi kriterleri dikkate alınır. Ayrıca sistem ihtiyaçları değerlendirilerek kayıt altına alınır.[2]

3.2.2.2. Sistem yapısal tasarım

Bu kapsamda sistemin üst seviye yapısı oluşturulur. Bu yapıda sistemin donanım ve yazılım birimleri, sistem ihtiyaçlarının donanım ve yazılım birimleri arasındaki ilişkisi belirlenir.

Yapılan tasarım dokümante edilir. Tasarım ile ihtiyaç, donanım ve yazılım ilişkileri; sistem ihtiyaçlarının izlenebilirliği, yapılan tasarım ile ihtiyaç, donanım ve yazılım ilişkilerinin sistem ihtiyaçlarına uygunluğu, yazılım birimlerinin kendileriyle ilgili ihtiyaçları karşıladığının fizibilitesi, işletim ve bakım fizibilitesi kriterlerine göre gözden geçirilir ve yapılan değerlendirmeler kayıt altına alınır. [2]

3.2.2.3. Yazılım ihtiyaçları analizi

Bu kapsamda her bir yazılım birimi için performans, fiziksel karakteristikler, çevre koşullarını içeren fonksiyonel ve kapasite özellikleri çıkartılır, her bir yazılım biriminin dış arayüzleri, yeterlilik ihtiyaçları, güvenlik özellikleri, gizlilik özellikleri, ergonomik özellikler, veri tanımlamaları ve veritabanı ihtiyaçları, uygulama ve bakım alanlarında kullanılacak yazılım ürününün yükleme ve kabul ihtiyaçları, kullanıcı dokümanları, kullanıcı uygulama ve işletme ihtiyaçları belirlenir ve dokümante edilir.

Yapılan analiz, sistem ihtiyaçları ve sistem tasarımının izlenebilirliği, sistem ihtiyaçları ile yazılım ihtiyaçlarının uygunluğu, yazılım ihtiyaçlarının iç uygunluğu, test edilebilirlik, yazılım tasarımının fizibilitesi, uygulama ve bakım fizibilitesi kriterlerine göre gözden geçirilir. Yapılan değerlendirmeler kayıt altına alınır. Teknik personelin ve yöneticilerin katıldığı teknik gözden geçirme toplantısı düzenlenir.[2]

3.2.2.4. Yazılım yapısal tasarım

Bu kapsamda ihtiyaçlar, her bir yazılım birimi için, üst seviye yapının oluşturulduğu ve yazılım bileşenlerinin belirlendiği yapıya dönüştürülür. Bütün ihtiyaçların yazılım bileşenleri tarafından yerine getirildiğinden ve yazılım yapısının sistem yapısına uygun olduğundan emin olunur.

Yazılım birimleri ve bileşenleri için arayüz tasarımı yapılır ve dokümante edilir. Veritabanı üst seviye tasarımı yapılır. Kullanıcı dokümanlarının taslak tasarımları yapılır. Yazılım entegrasyonunu içeren bir takvim ve testleri, doğrulamaları içeren bir plan hazırlanır.

Yazılım birimleri yapısı, arayüz ve veritabanı tasarımları yazılım birimi ihtiyaçlarının izlenebilirliği, yazılım birimlerinin dış uygunluğu, yazılım bileşenlerinin iç uygunluğu, kullanılan tasarım metotları ve standartların uygunluğu, detay tasarımının fizibilitesi, uygulama ve bakım fizibilitesi kriterlerine göre değerlendirilir ve değerlendirme sonuçları kayıt altına alınır. Teknik personelin, yöneticilerin ve müşterinin katıldığı ön tasarım gözden geçirme toplantısı düzenlenir.[2,3]

3.2.2.5. Yazılım detay tasarım

Bu kapsamda her bir yazılım birimine ait yazılım bileşeni için detaylı tasarım yapılır. Yazılım bileşenleri kodlanabilecek, derlenebilecek, test edilebilecek yazılım ünitelerine dönüştürülür. Yazılım bileşenlerinin yazılım ünitelerine dönüştürüldüğünde bütün ihtiyaçları karşıladığından emin olunur ve bu tasarım dokümante edilir.

Yazılım birimleri, bileşenleri ve üniteleri için arayüz tasarımı yapılır ve bu bilgiler dokümante edilir. Veritabanı detay tasarımı yapılır ve gerekirse kullanıcı dokümanları güncelleştirilir. Yazılım ünitelerini de içerecek şekilde ihtiyaç limitlerini belirterek, yazılım entegrasyonu için takvim oluşturulur.

Yazılım detay tasarım ve test ihtiyaçları; yazılım birimi ihtiyaçlarının izlenebilirliği, yapısal tasarımla dış uyumluluk, yazılım bileşenleri ile yazılım üniteleri arasındaki iç uyumluluk, kullanılan tasarım metotları ve standartların uygunluğu, test fizibilitesi, uygulama ve bakım fizibilitesi kriterlerine göre değerlendirilir. Değerlendirme sonuçları kayıt altına alınır. Teknik personelin, yöneticilerin ve ihtiyaç makamının katıldığı kritik tasarım gözden geçirme toplantısı düzenlenir. [2,3]

3.2.2.6. Yazılım kodlama ve test

Bu kapsamda her bir yazılım birimi, veritabanı ve bunların test prosedürleri ile testler için gerekli veriler tasarlanır ve dokümante edilir. Oluşturulan bütün hedef kodları bilgisayar ortamında saklanır. İlgili kayıtlar ve prosedürler hazırlanır. Her bir yazılım ünitesi ile veritabanı için hazırlanan dokümanlara göre test yapılır, sonuçlar raporlanır. Eğer gerekirse kullanıcı dokümanları güncelleştirilir.

Yazılım kodu ve test sonuçları; yazılım birimi tasarım ve ihtiyaçlarının izlenebilirliği, yazılım biriminin tasarımının ve ihtiyaçlarının dış uyumluluğu, yazılım üniteleri ihtiyaçları arasındaki iç uyumluluk, testlerin üniteleri kapsaması, kullanılan kodlama metotları ve standartların uygunluğu, yazılım entegrasyon ve testlerin fizibilitesi ve bakım fizibilitesi kriterlerine göre değerlendirilir ve sonuçlar dokümante edilir. [2]

3.2.2.7. Yazılım entegrasyonu

Bu kapsamda yazılım birimleri, yazılım bileşenleri ve ünitelerinin entegrasyonu için plan ve prosedürler hazırlanır. Yazılım birimleri ve bileşenleri entegrasyon planına göre entegre edilir. Uygulama sonuçları raporlanır. Yazılım yeterlilik testlerine yönelik olarak yazılım yeterlilik test ve değerlendirmelerini içeren prosedürler hazırlanır. Yazılım entegrasyonu; tasarım, kod, testler, test sonuçları ve kullanıcı dokümanları sistem ihtiyaçlarının izlenebilirliği, sistem ihtiyaçlarının dış uyumluluğu, iç uyumluluk, testlerin yazılım birimleri ihtiyaçlarını kapsaması, kullanılan test metotları ve standartların uygunluğu, beklenen sonuçlara ulaşma, yazılım yeterlilik testlerinin fizibilitesi, uygulama ve bakım fizibilitesi kriterlerine göre değerlendirilir ve bu değerlendirmeler dokümante edilir. Teknik gözden geçirme toplantısı düzenlenir. [3]

3.2.2.8. Yazılım yeterlilik testi

Bu kapsamda önceden hazırlanan prosedürlere göre testler yapılır ve sonuçlar raporlanır. Gerekirse kullanıcı dokümanları güncelleştirilir. Yazılım yeterlilik testi; tasarım, kod, testler, test sonuçları ve kullanıcı dokümanlarının testlerin yazılım birimleri ihtiyaçlarını kapsaması, beklenen sonuçlara ulaşma, sistem

entegrasyonu ve testlerinin fizibilitesi, uygulama ve bakım fizibilitesi kriterlerine göre değerlendirilir ve değerlendirme sonuçları dokümante edilir.

Denetlemeler gerçekleştirilir ve sonuçları raporlanır. Eğer donanım ve yazılımın her ikisi de tasarım veya entegrasyon aşamasında ise bu denetlemeler sistem yeterlilik testlerine kadar ertelenebilir. Denetlemeler başarı ile tamamlanmışsa ve gerekliyse sistem entegrasyonu, sistem yeterlik testi, yazılım yükleme veya yazılım kabulü için yazılım ürünü güncellenir. [2,3]

3.2.2.9. Sistem entegrasyonu

Sistem entegrasyonunda, yazılım konfigürasyon birimleri, donanım konfigurasyon birimleri ve ilgili diğer birimlerle beraber sisteme entegre edilir. Sistem entegrasyon test ve doğrulamaları içeren prosedürler hazırlanır. Sistemin tüm ihtiyaçlarını gerçekleştirecek şekilde tasarlanıp tasarlanmadığı test edilir ve sonuçlar raporlanır. Sistem yeterlilik testlerine yönelik prosedürler hazırlanır ve sistemin, sistem yeterlilik testlerine hazır olduğundan emin olunur.

Entegre edilen sistem; testlerin sistem ihtiyaçlarını kapsaması, kullanılan test metodlarının ve standartların uygunluğu, beklenen sonuçlara ulaşılması, sistem yeterlilik testlerinin fizibilitesi, uygulama ve bakım fizibilitesi kriterlerine göre değerlendirilir ve değerlendirmeler dokümante edilir. [2,3]

3.2.2.10. Sistem yeterlilik testi

Bu kapsamda sistem entegrasyonu sırasında hazırlanan prosedürlere göre testler yapılır ve sonuçlar raporlanır. Sistem yeterlilik testleri; testlerin sistem ihtiyaçlarını kapsaması, beklenen sonuçlara ulaşma, uygulama ve bakım fizibilitesi kriterlerine göre değerlendirilir ve değerlendirmeler dokümante edilir.

Denetleme gerçekleştirilir ve sonuçlar raporlanır. Denetlemelerin başarı ile tamamlanmasından sonra yazılım yükleme ve yazılım kabulü için yazılım ürünü güncellenir ve hazırlanır. [2,3]

3.2.2.11. Yazılım yükleme

Yazılım yükleme kapsamında; yazılım ürününün hedef ortamda yüklenebilmesi için gerekirse bir plan hazırlanır. Yükleme planına göre yazılım

ürününün yüklemesi yapılır ve yazılım kodunun, veritabanının; başlatılması, işletilmesi ve sonlandırılmasının kontratta belirtildiği şekilde olduğundan emin olunur. Yükleme sırasındaki işlemler dokümante edilir. [2,3]

3.2.2.12. Yazılım kabulü

Yazılım kabulü kapsamında; yazılım programının müşteri tarafından yapılacak yazılım kabul gözden geçirme ve testlerine gerekli destek verilir. Bu faaliyetlerde bu ana kadar yapılmış olan gözden geçirmeler, denetlemeler, yazılım yeterlilik testleri ve sistem yeterlilik testlerinin sonuçları dikkate alınır. Kabul gözden geçirme sonuçları raporlanır. Kontratta belirtildiği gibi yazılım ürünü sonuçlandırılır ve dağıtımı yapılır. [2,3,4]

3.2.3. İşletim süreci

İşletim(Operation) süreci kullanıcının; sistemin işletilmesi sırasında ihtiyaç duyacağı danışmanlık hizmetlerini kapsar. Bu hizmetler teknik danışman tarafından, kontratta belirtilmesi halinde verilir ve bu hizmetlerle ilgili bilgileri içeren bir plan hazırlanır. Problemlerin kayıt altına alınması, çözümü, takibi ve geri besleme yöntemleri yapılır. Yazılım ürününün işletme ortamındaki testleri, problem raporları ve modifikasyon taleplerinin bakım sürecine aktarılması ve yazılım ürününün kullanıcıya verilmesi işletme sürecini anlatan plana göre yapılır. Bu süreçte verilecek hizmetler işletimsel testler, danışma ve yardım sağlanması, söz konusu faaliyetler ve takip eden çalışmaların izlenmesi ve kayıt altına alınması, sistem işletimi, kullanıcı desteğidir. [2,3,4]

3.2.3.1. İşletimsel testler

İşletimsel testler yapılır ve yazılım kodunun, veritabanının; başlatılması, işletilmesi ve sonlandırılmasının hazırlanan işletme süreç planına uygunluğu takip edilir. [2]

3.2.3.2. Sistem işletimi

Kullanıcı dokümanlarına göre sistem işletilir. [2]

3.2.3.3. Kullanıcı desteği

Kullanıcı desteği kapsamında; kullanıcının talebi üzerine ihtiyaç duyulan danışma ve yardım sağlanır. Kullanıcı talepleri eğer gerekli görülür ise teknik danışman tarafından bakım sürecine aktarılır. [2]

3.2.4. Bakım süreci

Bakım(Maintenance) süreci; yazılım ürününün bir iyileştirme, adaptasyon ihtiyacı veya bir problem sonucunda, kodda ve ilgili dokümantasyonda meydana gelebilecek modifikasyon ihtiyacının ortaya çıkması ile başlar. Bakımın nasıl yapılacağını anlatan bir plan hazırlanır. İşletme sürecinden gelen problem ve modifikasyon talepleri, geliştirme sürecinden gelen problem ve modifikasyon talepleri, taşıma talepleri ve sonlandırma talepleri yapılır. İşletme sürecinden gelen kullanıcı yardım talepleri kayıt altına alınır ve saklanır. Problem ve modifikasyon analizleri yapılarak bir mühendislik değişiklik teklifi hazırlanır. Bakım personeli; hangi dokümanın hangi aşamada güncelleştirileceğini; hangi yazılım ünitesi ve versiyonun modifiye edileceğini belirler. Tüm bu bilgiler de hazırlanacak mühendislik değişliklik teklifinde yer alır. Bu teklifi incelemek için oluşturulmuş olan bir kurul, toplanarak hazırlanan bu mühendislik değişiklik teklifini inceler ve uygulanıp uygulanmayacağına karar verir. Kurulun alacağı sonuç kullanıcıya bildirilir. Değişikliğin uygulanmasına karar verilmesi durumunda düzeltici işlemi içeren bir rapor hazırlanarak ilgili geliştirme süreci basamakları takip edilir.

Modifiyeden etkilenen ve etkilenmeyen bölümler için test ve değerlendirme kriterleri belirlenir ve bunların sonuçları dokümante edilir. Yazılım yüklenmesinden sonra kabule hazırlık gözden geçirme toplantısı düzenlenerek kullanıcıya bilgi verilir ve yazılımın kabulü, dağıtımı yapılarak açılmış olan düzeltici işlem raporu kapatılır.

Herhangi bir geliştirme alt sürecinde karşılaşılan problemler verilen düzeltici işlem raporuyla kayıt altına alınır ve problemler bilgisayar ortamında takip edilir. Problemi tespit eden kişi, problemi bu rapor üzerinde tarif eder, problemin kategorisini belirler. Tespit edilen kategoriye göre proje yazılım grubunun lideri; problemi, çözümü için ilgili mühendise aktarır. İlgili mühendis düzeltici işlemi

yapar ve rapor üzerinde ilgili bölümleri doldurur. Problemi tespit eden kişi, yapılan düzeltici işlemle problemin giderildiğini test ederek raporu kapatır. [2,3,4]

3.2.4.1. Yazılımın başka bir ortama taşınması

Yazılımın başka bir ortama hangi koşullarda ve nasıl taşınacağını tanımlamak için bir plan hazırlanır. Taşıma planları ve faaliyetleri ile ilgili kullanıcılara bilgi verilir. Yeni çalışma ortamına geçiş esnasında eski çalışma ortamının kullanılması gerekiyorsa kontratta belirtildiği şekilde gerekli olan eğitimler sağlanır. Taşıma zamanı geldiğinde bütün ilgililer uyarılır ve eski çalışma ortamı ile ilgili dokümanlar, kayıtlar ve kod arşive kaldırılır. Yeni ortama geçiş etkilerinin değerlendirilmesi için gözden geçirme toplantısı yapılır. Kontratta belirtilen yöntemlere göre eski çalışma ortamına ait verilere ulaşım imkanı sağlanır. [2,3,4]

3.2.4.2. Desteğin sonlandırılması

Desteğin sonlandırılması kapsamında; bakım ve işletme ile ilgili aktif desteğin sonlandırılması konusunda bir plan hazırlanır. Kullanıcılar, desteğin sonlandırılması plan raporu vasıtasıyla yapılan faaliyetler ve planlarla ilgili bilgilendirilir. Yeni yazılım ürününe geçiş esnasında eski ürünün kullanılması gerekiyorsa kontratta belirtildiği şekilde gerekli olan eğitimler sağlanır. Sonlandırma zamanı geldiğinde bütün ilgililer uyarılır ve eski ürün ile ilgili dokümanlar, kayıtlar ve kod arşive kaldırılır. Kontratta belirtilen yöntemlere göre eski ürüne ait verilere ulaşım imkanı sağlanır. [2,3,4]

3.3. Destekleyici Süreçler

Destekleyici süreçler; yazılım yaşam döngüsüne destek veren süreçlerdir. Bu süreçler diğer süreçlerle içiçe girmiş durumdadır. Kontrat aşamasından yazılımın teslim edilmesi ve bakımına kadar olan süre boyunca standartta belirlenmiş zamanlarda gerçekleştirilirler. [2,3,4]

3.3.1. Dokümantasyon

Dokümantasyon(Documentation) süreci, yazılım yaşam döngüsü içine yayılmış bir süreçtir. Kayıt altına alınması gereken bilgiler ve bu bilgilerin hangi

doküman, kayıt yada plan içerisinde tanımlanması gerektiği belirtilir. Ayrıca her dokümanın, planın ve kaydın oluşturulması yada güncelleştirilmesi gereken zaman verilir.

Dokümantasyon, uygulanması zor ve zaman alıcı ama olmazsa olmaz bir süreçtir. Yapılacak ve yapılan tüm işlemlerin, planların kayıt altına alınması, yaşam döngüsü boyunca projenin kontrolünü kolaylaştırır. Hatalar daha çabuk bulunup düzeltilebilir. Projeye yeni katılan birinin projeye daha çabuk adapte olmasını, müşterinin projeyi takibini ve yazılım teslim edildikten sonra yazılımın bakımını kolaylaştırır. [3]

3.3.2. Konfigürasyon yönetimi

Konfigürasyon yönetimi(Configuration Management) süreci; yazılım yaşam döngüsü boyunca diğer süreçlerle paralel ve etkileşimli olarak devam eden yönetimsel bir süreçtir. Konfigürasyon yönetimi, geliştirilen ürünün yönetimi içindir. Ürünün ve onu oluşturulan parçaların belirlenmesini sağlarken her bir parçanın değişiklik kontrollerini de sağlar. Bu süreçte konfigürasyon öğelerinin nasıl belirleneceği, öğelerin değişiklik kontrollerinin nasıl ve ne zaman yapılacağı, değişiklik isteklerinin ne zaman olabileceği ve bu isteklerin kayıtlarının nasıl tutulacağı anlatılmaktadır. [3]

3.3.3. Kalite güvence

Kalite güvence(Quality assurance) sürecinin amacı; bir sürecin yada projenin iş ürünlerinin ve etkinliklerinin uygulanabilir, tüm standart, prosedür ve isterlerle uyumlu olduklarının güvenceye alınmasıdır. Bu süreç; doğrulama, değerlendirme, gözden geçirme, denetleme ve problem çözme gibi diğer destekleyici süreçlerin sonuçlarını kullanabilir. Kalite güvencenin diğer birimlerden bağımsız olması gerekmektedir. Kalite güvence sürecinde çalışacak ayrı bir birim olması gerekir ve bu birim yazılımı geliştirmekle sorumlu kişileri direk olarak kontrol edebilmelidir. Ayrıca bu birim projenin çeşitli basamaklarında projenin planlara ve standartlara uygun ilerleyip ilerlemediğini kontrol edebilmelidir. Kalite güvence, IEEE/EIA 12207'ye göre süreçlerin doğru bir şekilde gerçekleştirilip gerçekleştirilmediğinin

kontrolünden, ürün güvencesinden, süreç güvencesinden ve sistemlerin kalitesinin güvencesinden sorumlu olan bir süreçtir. [3]

3.3.4. Değerlendirme

İş ürünlerini doğrulama sürecidir. Değerlendirme(verification) sürecinin amacı; her iş ürününün isterlerini uygun olarak yansıttığını güvence altına almaktır. Değerlendirme, ürünü doğru olarak yapıyor muyuz sorusuna verilecek cevapları içerir. Yazılım yaşam döngüsünün hangi safhalarında, ne amaçla değerlendirmeler yapılacağı ve sonuçların nasıl kayıt altına alınacağını tanımlanır.

3.3.5. Doğrulama

İş ürünlerini sağlama(validation) sürecidir ve amacı her iş ürününün belli bir amaçla kullanımı için gerekli isterlerin yerine getirildiğinin onaylanmasıdır. Doğrulamada amaç; istenen özelliklere sahip, kullanılabilir bir yazılım/sistem oluşturup oluşturulamadığının değerlendirilmesidir. Standartta doğrulama sürecinin gereklerinin yazılım yaşam döngüsü içerisinde nasıl ve ne zaman yapılması gerektiğini, doğrulama için gerekli planların nasıl hazırlanacağı ve sonuçların nasıl kayıt altına alınacağı belirtilir. [3]

3.3.6. Gözden geçirme toplantıları

Yazılım yaşam döngüsünün belirli aşamalarında yapılacakları ve yapılanları gerek müşteri ile gerekse proje ekibi ile değerlendirip projenin gidişini kayıt altına almak için toplantılar(joint review) yapılması gerekir. Bu, herhangi bir aşamada hata yada eksiklik yapıldıysa bunların farkedilip düzeltilmesini sağlar.

Tüm bu toplantılarda görüşülen konular ve taraflara açılan işlem maddeleri kayıt altına alınarak; toplantı sonucu taraflarca imzalanır. Toplantı sonuç raporu organizasyon tarafından arşivlenir.

Yazılım yaşam döngüsü boyunca yapılacak toplantılar ön tasarım gözden geçirme toplantıları, kritik gözden geçirme toplantıları, kabule hazırlık gözden geçirme toplantıları ve proje gözden geçirme toplantılarıdır.

Ön tasarım gözden geçirme toplantıları; yazılım yapısal tasarımının tamamlanmasından sonra gerçekleştirilir. Toplantıya müşteri, ilgili teknik personel ve yöneticiler katılır.

Kritik gözden geçirme toplantıları; yazılım detay tasarımının tamamlanmasından sonra gerçekleştirilir. Toplantıya müşteri, ilgili teknik personel ve yöneticiler katılır. Bu toplantıda yapılan tasarımlar dondurulur.

Kabule hazırlık gözden geçirme toplantıları; kabulden önce yapılan, modifikasyonun gözden geçirilmesi amacıyla müşterinin, ilgili teknik personel ve yöneticilerin katılımıyla gerçekleştirilen toplantılardır.

Teknik gözden geçirme toplantıları; proje boyunca yapılan faaliyetlerin değerlendirilmesi için, projede görev alan teknik personel, gerekirse proje yöneticileri ve müşterinin katılımıyla yapılan toplantılardır.

Proje gözden geçirme toplantıları; kontratta belirtilen periyotlarda ve kritik gözden geçirme toplantısından sonra müşterinin, ilgili teknik personelin ve yöneticilerin katılımıyla gerçekleştirilen toplantılardır. [3]

3.3.7. Denetleme

Denetleme süreci yazılım yaşam döngüsü ile içiçe girmiş bir süreçtir. Ürün kalitesini denetlemek için yapılır. Bazı test dokümanlarının kontrolünün yapılmasını, prosedürlere uygunluğun değerlendirilmesini ve kayıtların arşivlenmesini denetler.

Proje denetlemeleri; proje süresince yapılan faaliyetlerin yapılmasının kontrolü amacıyla organizasyon, kalite mühendisleri ile yazılım grubu kalite mühendisleri tarafından gerçekleştirilir. Proje denetleme programı kritik gözden geçirme toplantısından önce organizasyon kalite mühendisleri tarafından yayınlanır. Proje denetlemeleri iki kapsamda yapılır.

Yazılım entegrasyon denetlemesi; yazılım yeterlilik testinden sonra, yazılım yeterlilik test dokümantasyon kontrolü, test sonuçları, prosedürlere uygunluk, kayıtların arşivlenmesi kriterlerine göre hazırlanmış çekliste göre yapılır.

Sistem entegrasyon denetlemesi; sistem yeterlilik testinden sonra, sistem yeterlilik test dokümantasyon kontrolü, test sonuçları, prosedürlere uygunluk,

kayıtların arşivlenmesi kriterlerine göre hazırlanmış çekliste göre yapılır. Tespit edilen her bir aksaklık için gerekli işlemler yapılır. [2,3,4]

3.3.8. Problem çözme

Problem çözme(problem resolution) süreci; problemleri analiz eden ve problemlerin çözülmesini sağlayan bir süreçtir. Geliştirme, işletim, bakım ve diğer süreçlerin gerçekleştirilmesi sırasında yapılabilir. Problemlerin kayıt altına alınıp çözülüp çözülmediğinden emin olunmasını sağlar. Süreç gerçekleştirimlerini ve problem çözümlerini içerir. Yazılım ürünlerinde yada faaliyetlerdeki problemlere çözüm arandığı bir süreçtir.

Problem veya modifikasyon talebinin mevcut sistem üzerindeki etkisi incelenir. Analiz konuları;

- Tip; düzeltici, geliştirici, önleyici veya yeni bir ortama adapte edici vb.
- Kapsam; modifikasyonun büyüklüğü, maliyet, zaman vb.
- Kritiklik; performansa olan etki, güvenlik, gizlilik vb.

Problem bakım personeli tarafından tekrarlanarak problemin varlığı doğrulanır. Bakım personeli tarafından problem, modifikasyon talebi, analiz sonuçları ve uygulama seçenekleri kayıt altına alınır. [2,3,4]

4. ORACLE VERİTABANI YÖNETİM SİSTEMİ

Oracle veritabanının, işletim sistemi tarafından bakıldığında, biri fiziksel diğeri mantıksal olmak üzere iki bölümü vardır. Fiziksel bölüm, işletim sisteminden görünen kısımdır. Bunlar veri dosyaları, kontrol dosyaları ve log dosyalarından oluşmaktadır.

Mantıksal bölüm; bir yada daha fazla tablo uzayı(tablespace) ve tablolar(tables), görüntüler(views), sıralar(sequences), eşanlamlar(synonyms), indeksler(indexes), kümeler(clusters), veritabanı bağlantıları(database links), depolanmış prosedürler(stored procedure), fonksiyonlar(functions) ve paketlerden(packages) yani şema nesnelerinden(schema objects) oluşmaktadır.

Fiziksel bölüm; işletim sistemi tarafından görülebilmesine rağmen, mantıksal bölüm ancak Oracle'a bağlanıp, SQL komutları çalıştırılarak görülebilmektedir. Yani, Oracle kurulu herhangi bir bilgisayarda, SQL bilgisi olmayan bir insan, Oracle'ın sadece fiziksel bölümünü görebilmektedir.

Oracle veritabanındaki her nesnenin bir sahibi vardır. Her kullanıcı bir yada daha fazla tablo uzayına sahip olabilir. Her nesne, ait olduğu kullanıcının herhangi bir tablo uzayında (mantıksal olarak) bulunur. Her tablo uzayı da, kendisine sahip olan kullanıcının nesnelerini tutmak için işletim sisteminde bir veya daha fazla veri dosyasına sahip olabilmektedir.

Sonuç olarak; veritabanındaki her nesnenin bir kullanıcısı vardır ve bu nesneler mantıksal olarak o kullanıcının sahip olduğu tablo uzaylarının herhangi birinin içerisinde, fiziksel olarak da o kullanıcının sahip olduğu tablo uzayının herhangi bir veri dosyasında bulunur. Ancak, veri dosyasının içerisine işletim sisteminden bu nesneyi bulmak için bakılamaz. Nesnenin sahibi ve mantıksal yeri "DML" (veri işleme dili) komutları ile bulunabilmektedir. [8]

4.1. Fiziksel Bölüm

Fiziksel bölüm; veritabanını oluşturan işletim sistemi dosyalarıdır. Bir Oracle veritabanı, fiziksel olarak bir yada daha fazla veri dosyası, iki ya da daha fazla log dosyası, bir ya da daha fazla kontrol dosyasından oluşur.[8]

4.1.1. Veri dosyaları

Veri dosyaları; veritabanındaki tüm verileri tutan dosyalardır. Tablo, indeks gibi mantıksal veritabanı yapılarının içerisindeki veriler fiziksel olarak veri dosyalarında tutulurlar. Bir veri dosyası kendisi için ayrılan alan dolduğunda, kendi sahip olduğu alanı arttırabilecek özelliklere sahiptir. Bir yada daha fazla veri dosyası mantıksal bir veritabanı depolama ünitesi olan bir tablo uzayını oluştururlar.

Normal veritabanı işlemleri boyunca bir veri dosyası içerisindeki veriler okunur ve Oracle için ayrılan belleğe getirilirler. Örneğin; bir kullanıcının veritabanındaki bir tablonun verilerine erişmek istediğini varsayalım. Eğer istenilen veriler bellekte yer almıyorsa, ancak o zaman uygun veri dosyasından okunur ve belleğe getirilirler.

Değişikliğe uğrayan yada yeni eklenen veriler, veri dosyalarına hemen yazılmazlar. Sabit diske erişimi azaltmak ve böylece sistemin performansını artırmak için veriler bellek havuzunda tutulur ve gerektiğinde hepsi birden uygun veri dosyalarına kaydedilirler. Bunu Oracle'ın arka planda yapılan işlemleri belirler.[8]

4.1.2. Kontrol dosyaları

Tüm Oracle veritabanları kontrol dosyasına sahiptir. Bir kontrol dosyası veritabanı adı, veri dosyaları ve log dosyalarının adı, diskteki yeri, veritabanının oluşturulma tarihi gibi veritabanı ile ilgili bilgileri tutar.

Veritabanı oturumu açıldığında Oracle bu dosyayı kontrol ederek gerekli bilgileri alır. Eğer veritabanında yeni bir log yada veri dosyası oluşturulması gibi fiziksel bir değişme olursa, yapılan değişiklikler Oracle tarafından otomatik olarak kontrol dosyalarına yansıtılır.[8]

4.1.3. Log dosyaları

"Redo Log" dosyaları olarak bilinen bu dosyaların amacı; veriler üzerinde yapılan tüm değişiklikleri kaydetmektir. Eğer veri dosyalarına kalıcı olarak kaydedilmiş olan, değişikliğe uğramış kayıtlarda bir bozukluk olursa yapılan

değişiklikler redo log dosyalarından sağlanabilir ve işlemler kaybolmaz. Birden fazla tekrarlanan bozukluk durumlarında redo log dosyalarının da bozulmasını engellemek için Oracle farklı diskler üzerinde redo log dosyalarının birden fazla kopyasının alınmasına olanak sağlar.

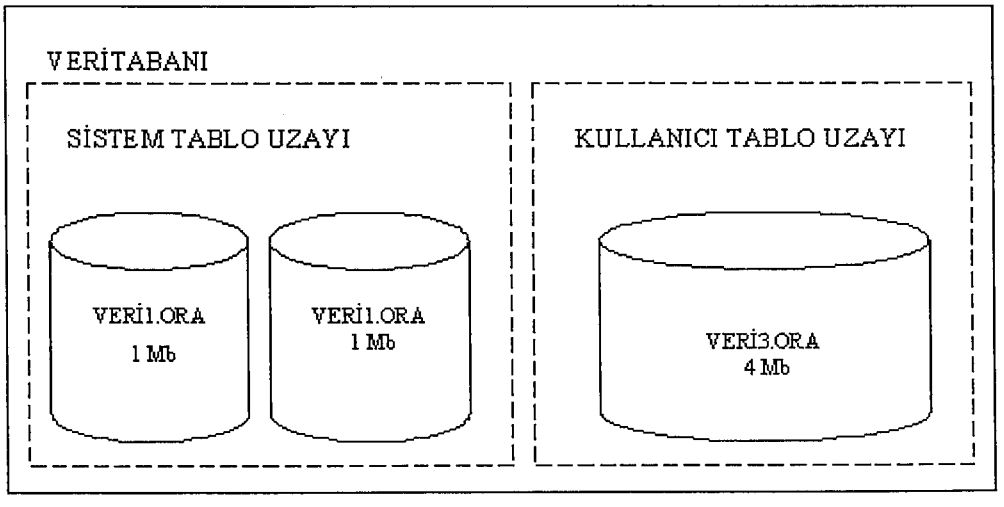
Bir veritabanı işlemi sırasında elektrik kesilirse, bellekteki veriler veri dosyalarına kaydedilmeyecek ve verilerin kaybolması durumuyla karşılaşılacaktır. Oracle veritabanı tekrar açıldığında redo log dosyalarında yapılan son değişiklikleri, veri dosyalarına yansıtılarak verilerin kaybolması engellenir.[8]

4.2. Mantıksal Bölüm

Oracle veritabanının mantıksal yapısı tablo uzaylarını, şema nesnelerini, veri bloklarını, genişlemeleri ve parçaları içerir. [7]

4.2.1. Tablo uzayı

Bir veritabanı, ilişkili mantıksal yapıların gruplanmasını sağlayan ve tablo uzayı olarak bilinen mantıksal depolama ünitelerine bölünmüştür. Şekil 4.1; veritabanı, tablo uzayı ve veri dosyaları arasındaki ilişkiyi açıklamaktadır.



Şekil 4.1 Oracle veritabanı yapısı

Buna göre:

- Bir veritabanı bir yada daha fazla tablo uzayına bölünmüştür.

- Tablo uzayı içerisindeki tüm mantıksal yapıları fiziksel olarak depolayabilmek için, her tablo uzayı bir yada daha fazla veri dosyasına sahip olabilir.
- Tablo uzaylarının toplam kapasitesi, sahip oldukları veri dosyalarının toplam kapasitesine eşittir.(Yukarıdaki şekil için sistem tablo uzayı 2MB, kullanıcı tablo uzayı 4MB).
- Tablo uzaylarının toplam kapasitesi veritabanının toplam kapasitesini belirler.(6 MB)

Bir tablo uzayı açık yada kapalı olabilir. Tablo uzayı kapalı olduğunda bu tablo uzayının içerisindeki nesnelere erişilemez. Bir tablo uzayı yönetim amaçlı olarak kapalı duruma alınabilir.[8]

4.2.2. Veritabanı şema nesneleri

Şema nesneleri mantıksal veri depolama yapıları olarak bilinir. Veritabanı üzerinde kullanıcının belirli işleri yapabilmesi için tanımlanan bu yapılar; tablolar, görüntüler, sıralar, eşanlamlar, indeksler, kümeler, veritabanı bağlantıları, prosedürler, fonksiyonlar ve paketlerdir. Bir şema ise bu nesnelerin oluşturduğu gruptur. [8]

4.2.2.1. Küme

Aynı anda sorgulanan birden fazla tablonun bir arada kaydedilmesine küme denir. Bu yapı, beraber sorgulanan tablolarda hız kazanmak için çok önemlidir. Örneğin; işçi ve bölüm adlarında iki tablo olsun. Her iki tablo da ortak olarak bölüm ve bölümno alanlarını içermekte olsun. Yani işçi tablosundaki bölüm, bölüm tablosundaki bölümno alanına karşılık gelir. Burada oluşturulacak bir kümede her iki tablo veritabanında yan yana getirilerek aynı veri bloğu içerisine kaydedilir. Böylece daha hızlı erişim sağlanır. [8]

4.2.2.2. İndeks

İndeksler; tablo ve kümeler için kullanılan veritabanı nesneleridir. Burada amaç; aranılan bir kayıda daha hızlı erişim sağlamaktır. Özellikle üzerinde çok arama yapılan alan veya alanlar üzerinde indeks oluşturmak çok etkilidir. İndeks

oluşturulduğunda ilgili tablonun kayıtları yer değiştirmez. Sadece ilgili kayıtların, kayıt numaraları olarak adlandırılan sıra tanıtıcıları(rowid) alınarak sıralama yapılır.

Bir tablo üzerinde oluşturulabilecek indeks sayısı sütunların kombinasyonları farklı olduğu sürece sınırsızdır ve bir sütun diğer sütunlarla değişik kombinasyonlarda kullanıldığı sürece birden fazla indeks içerisinde yer alabilir. Aynı sütun kombinasyonlarının indeksi, farklı indeks ismi kullanarak oluşturulmaya çalışılsa bile gerçekleştirilemez.

İndeksler mantıksal ve fiziksel olarak oluşturuldukları tablodan bağımsızdırlar. Eğer bir indeks silinirse, ilgili tablo zarar görmez, çalışmaya devam eder. Ancak; indeks olmadığı için veri erişim süresi artacaktır.

Oracle bir indeks oluşturulduğunda onu otomatik olarak kullanmaya başlar ve indeksin oluşturulduğu tablodaki silme, güncelleme ve ekleme işlemleri indekse otomatik olarak yansıtılır. [8]

4.2.2.3. Rol

Oracle veritabanında her nesnenin ait olduğu bir kullanıcı vardır. Bir kullanıcı bir başka kullanıcının nesneleri üzerinde işlem yapmak isterse buna hakkı olması gerekir. Bir nesne üzerinde işlem yapabilme yetkisine rol(role) denir. Örneğin; veritabanına bağlanma, tablo oluşturma, bir başkasına ait tablodan kayıt listeleme, bir başkasının prosedürünü çalıştırma birer roldür. Rol kullanıcılara atanmak suretiyle yapılır. Rolün kullanıcılara atanması iki şekilde olabilir. Birinci olarak bir tabloya kayıt ekleme, kayıt silme gibi roller bir kullanıcıya yada kullanıcılara ayrı ayrı atanır. İkinci şekilde ise; verilmek istenen haklar bir rol altında birleştirilir ve bu rol istenen kullanıcılara aktarılır.

Rol; sistem rolü ve nesne rolü olmak üzere ikiye ayrılır. Sistem rolü; veritabanı ile ilgili olarak önceden tanımlanmış roldür. Oracle'da 60'dan fazla sistem rolü tanımlanmıştır. Nesne rolü ise veritabanı nesneleri üzerinde işlem yapma hakkıdır. "Create Table", "Create TableSpace", "Drop Any Index" gibi roller, sistem rollerine örnek olarak verilebilir. Nesne rolleri ise; "Select", "Insert", "Update", "Delete", "Alter", "Index", "Execute", "References", "All" olmak üzere sekiz çeşittir. "All" ayrı bir rol olarak adlandırılmaz. Diğer tüm

rolleri kapsar. Bu roller sırayla kayıt seçme, kayıt güncelleme, kayıt silme, nesnelerin yapısını değiştirme, indeks oluşturma, alt program çalıştırma, yabancı anahtar tanımlayabilme işlemlerini içerirler. [8]

4.2.2.4. Geri alma parçası

Oracle; veritabanının güvenliği açısından "Select", "Insert", "Update", "Delete" gibi işlemlerin yedeğini almaktadır. Alınan bu yedeklerin konulduğu yerlere geri alma parçası(rollback) denir. Kullanıcı bu tip işlemleri yaptıktan sonra "Rollback" komutunu uygularsa, yaptığı değişiklikler geri alma parçalarından getirilir ve böylece kayıtlar eski haline dönmüş olur. Kullanıcı "Commit" komutunu verirse yaptığı değişiklikler geri alma parçalarından geri getirilemez. Her veritabanında bir yada birkaç tane geri alma parçası olabilir.[8]

4.2.2.5. Sıra

Tablolardaki kayıtlar için otomatik sıra numarası verilmesi isteniyorsa sıra nesnesi kullanılabilir. Sıra numarası veritabanı tarafından otomatik olarak üretilir. Özellikle çok kullanıcılı ortamlarda tekil olarak numara üretilmek istendiğinde çok kullanışlıdır. Birden fazla kullanıcı aynı anda böyle bir sayı üretmek isterse bunun program koduyla yapılması işlem hızını yavaşlatır. Çünkü bir kullanıcı, diğer kullanıcı işini bitirene kadar beklemek zorundadır. Sıra nesnesi bu işi otomatik olarak ve çok seri bir şekilde başarır.

Sıra numaraları, Oracle'da tanımlı 38 rakama kadar tamsayılardan oluşur. Bir sıra tanımlaması sıranın adını, artan yada azalan olacağını, iki sayı arasındaki fark miktarını içerir. Oracle tüm sıra numarası tanımlarını sistem tablo uzayının içerisindeki bir veri sözlüğü tablosuna kaydeder. Sistem tablo uzayı sürekli çalışır durumda olduğu için tüm sıra numaraları da aktiftir. Sıra numaraları tablolardan bağımsız olarak üretilir yani bir sıra numarası, bir yada daha çok tablo için kullanılabilir.[8]

4.2.2.6. Kayıtlı fonksiyonlar ve kayıtlı prosedürler

Bir grup SQL yada PL/SQL komutunun belli bir işi gerçekleştirmek için bir araya getirilip veritabanına kaydedilmesi kayıtlı prosedürler ve fonksiyonlar

sayesinde olur. Birden fazla uygulama programı içerisinde aynı işi yapan kodları sürekli yazmak yerine kayıtlı prosedür ve fonksiyonlar bir kere yazılır ve tüm uygulamalar tarafından kullanılır. SQL*Plus, Oracle Forms ya da Oracle Reports içerisinden bu prosedürler çağırılabilir. Prosedürler, geriye birden fazla değer döndürebilirken, fonksiyonlar sadece bir değer döndürürler.[8]

4.2.2.7. Eşanlam

Eşanlam; bir tablo, görüntü, sıra, prosedür, fonksiyon yada paket için "alias" olarak adlandırılan bir takma isimdir. Eşanlam bir takma isim olduğu için veri sözlüğü içerisindeki tanımının kapladığı yer haricinde, veritabanında yer kaplamaz. Eşanlamlar güvenlik ve daha rahat kod yazma amacıyla kullanılırlar. Bir eşanlam kullanıldığında ilgili nesnenin adı ve sahibi gizlenir ve SQL komutu içerisinde kullanımı kolaylaşır.

Eşanlamlar; genel(public) ve özel(private) olarak tanımlanabilirler. Genel olarak tanımlanan eşanlamlara tüm veritabanı kullanıcıları erişebilir. Özel olarak tanımlanan eşanlamlara ise sadece ilgili nesnenin sahibi ve sahibi tarafından hak verilmiş bir başka kullanıcı erişebilir.

Bir nesnenin adı değiştirilmek yada silinmek istendiğinde, bu nesneyi kullanan tüm uygulama programları değiştirilmek zorundadır. Oysa ki, bu nesnenin bir eşanlamı oluşturulursa ve uygulama programları bu eşanlamı kullanırsa, eşanlam üzerinde yapılacak değişikliklerle silme gibi işlemlerden uygulama programlarının etkilenmesi önlenebilir.

Bir kullanıcı bir başka kullanıcının nesnesini kullanmak istediğinde "kullanıcı_adı.nesne_adı" şeklinde bir yazım kuralına uymak zorundadır. Eğer genel olarak bir eşanlam tanımı yapılırsa tüm kullanıcılar direk eşanlam ismini kullanarak işlemlerini gerçekleştirebilirler.[8,9]

4.2.2.8. Tablo

İlişkisel veritabanı yönetim sistemlerinde veriler tablolar içerisinde yer alır. Her tablo bir isimle tanımlanır ve her biri bir "kayıt" olarak adlandırılan satırlar ile bu kayıtlardaki verilerin özelliklerini belirleyen sütunlardan oluşur. Her tablo bir yada daha fazla sütuna sahip olabilir. Her sütunun bir adı ve veri tipi vardır.

Bir tablo oluşturulduğunda Oracle, verileri depolamak için bir tablo uzayı içerisinde bir veri segmenti ayırır. Veri segmentinin değerleri değiştirilerek bir tablo için ayrılacak yer miktarı da değiştirilebilir. [8]

4.2.2.9. Görüntü

Görüntü, bir yada birkaç tablodan istenilen alanların alınmasıyla oluşturulan sanal bir tablodur. Görüntü, bu tablolar üzerinde gerçekleştirilen bir sorgu sonucu oluşturulur. Bir görüntü üzerinde silme, güncelleme gibi işlemler yapılamaz. Görüntü tabloların sadece o anlık değeridir ve veritabanında kendi tanımının kapladığı yer haricinde yer kaplamaz. [8]

4.2.3. Veri blokları, genişlemeler ve parçalar

Oracle veritabanında verilerin depolandığı en küçük birim, veri bloğu olarak adlandırılır. Bir veri bloğu veritabanının depolama alanı üzerindeki belli bir bayt sayısına karşılık gelir. Veri bloğunun hacmi veritabanı oluşturulurken belirlenir.

Veri bloklarının bir üst birimi genişleme olarak adlandırılır. Bir genişleme ardı ardına gelen belirli sayıda veri bloğundan oluşur.

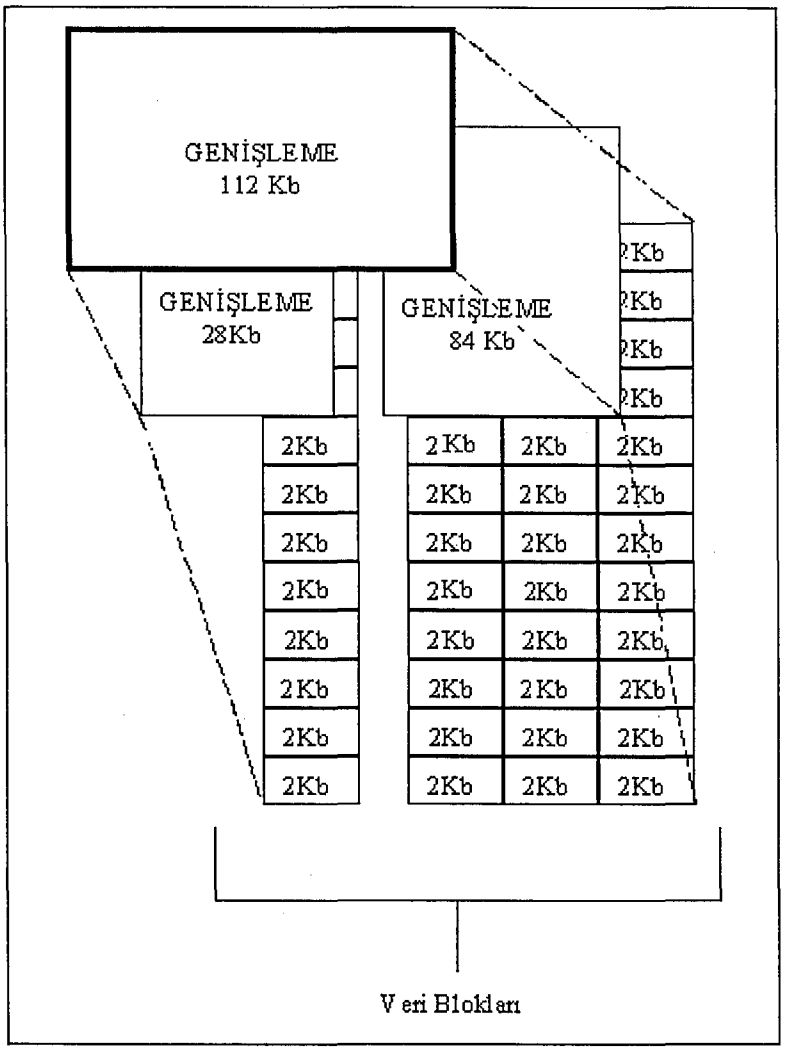
Genişlemelerin bir üst birimi de parçalardır. Parçalar belli bir mantıksal yapı için ayrılmış bir dizi genişlemeden oluşurlar. Farklı amaçlar için kullanılan parçalar da vardır. Bunlar; veri parçaları(data segments), indeks parçaları(index segments), geri alma parçaları (rollback segments) ve geçici parçalardır(temporary segment).

Bir tablo, bir veri parçasından oluşur. Tablonun verileri bu parça içerisindeki genişlemelere kaydedilir. Oracle8i ile birlikte gelen yeni bir özellik olan bölümlenmiş tablo yapısında ise her bölüm bir veri parçasına karşılık gelir. Bir küme de bir veri parçasından oluşur ve küme içerisindeki tüm tablolar o kümenin veri parçasında yer alır.

Her bir indeks de, bir indeks parçasından oluşur ve indeksin bütün verileri indeks parçasında yer alır. Bir veritabanında, veritabanı kurtarma işlemleri (database recovery) ve yapılan değişiklikleri geri alma işlemleri (rollback) için bir yada daha fazla geri alma parçası yer alır. Geçici parçalar Oracle tarafından SQL

komutları işletilirken ihtiyaç olduğunda oluşturulur ve SQL komutu işlemini bitirdiğinde bu parça tekrar sistemin kullanımına bırakılır. [9]

Şekil 4.2 veri bloklarını, genişlemelerini ve parça ilişkisini göstermektedir.



Şekil 4.2 Veri blokları, genişlemeler ve parçalar

4.3. PL/SQL

PL/SQL; C, Pascal gibi programlama dillerinde uygulanan prosedürel yapıları SQL ifadeleriyle birleştiren bir dildir. PL/SQL ile bu dillerde yazılan programlara benzer programlar yazılarak, bu programlar içerisinde SQL komutları kullanılabilir.

SQL komutlarının istemci tarafından çalıştırılması yerine PL/SQL ile yazılmış, depolanmış prosedürler vasıtasıyla çalıştırılması ağ trafiğinin azaltılması ve performans artışı yönünden oldukça faydalıdır.

PL/SQL kullanımının avantajları şöyle sıralanabilir :

Nesneye Yönelik Programlama : PL/SQL'de bulunan nesne tipi, nesneye yönelik programlar geliştirmeyi sağlar. Nesne tipi, C++ programlama dilindeki sınıf tipine benzer. Bu veri tipinin kullanımı ile kompleks uygulamaların, geliştirilme maliyeti ve geliştirilme süresi azaltılır. Nesne tipi, modüler, tekrar kullanılabilir ve kolay idame edilebilir programların yazılmasını mümkün kılmanın yanı sıra, programcıların aynı zamanda paralel olarak çalışmasını da sağlar. Nesne veri tipinin kullanılması ile uygulama detayları nesnelerin metodları içerisine saklanır. Böylece uygulama detayları istemci programları etkilenmeden değiştirilebilir.

Güvenlik : PL/SQL kullanılarak istemcilerin, veritabanına sadece çalıştırma yetkileri olan PL/SQL blokları ile ulaşmaları sağlanabilir. Örneğin, belirli bir kullanıcı grubuna, sadece bir tabloya kayıt ekleyen bir PL/SQL bloğuna yetki verilebilir. Böylece kullanıcıların bu tablo içerisindeki kayıtları görmesi ve silmesi engellenmiş olur.

SQL Desteği : PL/SQL bloklarında "Select", "Insert", "Update", "Delete" gibi tüm veri işlem komutları ve "Commit", "Rollback" gibi hareket komutları ile birlikte SQL fonksiyon ve operatörleri kullanılabilir. Ancak; "Create","Drop" gibi veri tanımlama komutlarının kullanımına izin verilmez. Buna rağmen veri tanımlama komutları dinamik SQL kullanılarak PL/SQL programları içerisinde işletilebilir.

Performans Artışı : PL/SQL kullanılmadığı durumlarda her bir SQL ifadesi ayrı ayrı Oracle veritabanına gönderilir. Bir SQL ifadesinin sonucuna göre yeni SQL ifadelerinin üretilerek veritabanına gönderilmesi gerekebilir. Bu durumda her bir SQL ifadesinin ayrı ayrı veritabanına gönderilmesi ağ trafiğini arttırarak performası azaltacaktır. PL/SQL ile mantıksal olarak bir bütün oluşturan yani bir iş yapılmasını sağlayan SQL ifadelerinin tümü bir PL/SQL bloğunda toplanabilir. Ayrı ayrı gönderilen SQL ifadeleri yerine, çalışmaya hazır bir PL/SQL bloğunun çağırılması çok hızlı ve etkili olacaktır.

PL/SQL programları bloklardan oluşmaktadır.

Declare

Bu kısımda tipler, değişkenler ve lokal alt programların tanımlamaları bulunur.

Begin

Bloğun ana kısmıdır. SQL komutlarını ve PL/SQL program yapılarını içerir.

Exception

Ana blokta bir hata oluşması durumunda işletilecek program parçalarını içerir.

End

PL/SQL bloğunun bittiğini gösterir.[6]

4.3.1. Paketler

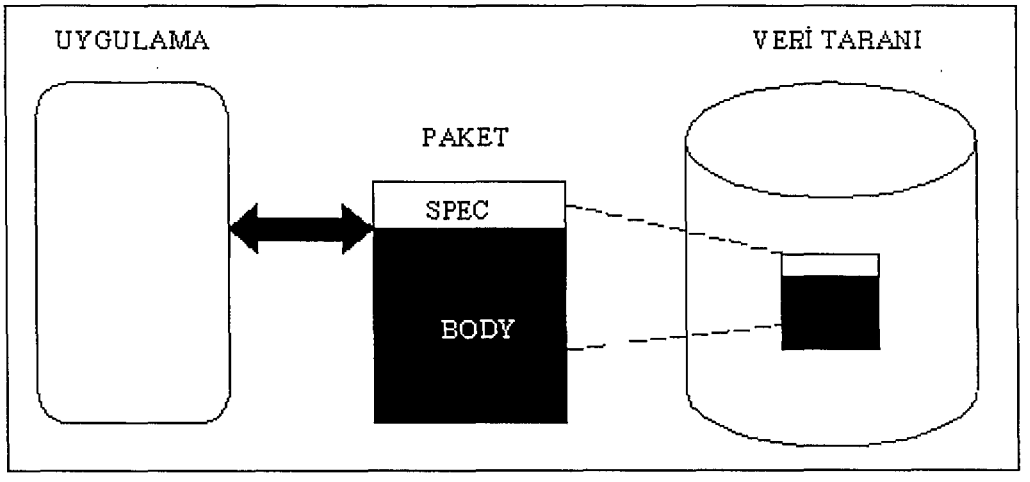
Paket, mantıksal olarak ilgili PL/SQL tiplerinin, değişkenlerinin ve alt programların toplandığı bir şema nesnesidir. Genelde paketler belirli bir tabloya, görüntüye uygulanacak tüm prosedürleri bir araya toplamak amacı ile oluşturulan koddur.

Organizasyonel yararlarına ek olarak, paket kullanımının iki yararı daha vardır. Bunlardan ilki, Oracle'ın prosedürlerden birini SGA hafıza bölgesine çekerken, bu prosedürün içinde bulunduğu tüm paketi SGA hafıza bölgesine çekmesidir. Kullanıcının başka bir prosedüre ihtiyacı olması durumunda tüm paketin zaten hafızada olmasından dolayı performans açık bir şekilde artar.

İkinci neden ise kullanıcı tarafından tanımlanmış veri tiplerinin prosedürde bir parametre olarak kullanılmasının tek yolunun paket kullanımı olmasıdır.

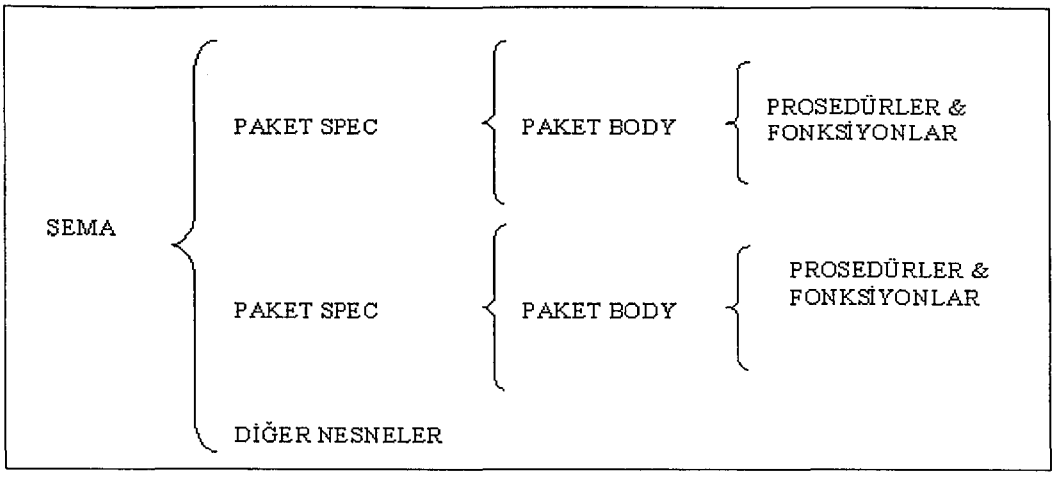
Paketler "Spec" ve "Body" olmak üzere iki kısımdan oluşur. "Spec" kısmı paketin diğer uygulamalarla olan arayüzünü oluşturduğu gibi tip, değişken, sabit, cursor, exception ve altprogram tanımlamalarını da içerir. "Body" kısmı ise arayüzde verilen prosedürlerin, fonksiyonların ve diğer tanımlamaların gerçeklenmesini içerir. Bu yapısı ile paket bir C modülüne benzer.

"Spec", uygulamaların public tanımlamalarını tutar. "Body" ise, "Spec"de bulunan public tanımların detaylarını ve uygulamaların sadece kendileri tarafından kullanılan ve başka uygulamaların kullanmasına izin verilmeyen private tanımlamaları içerir. Yani bir paketin "Spec"inde yer alan tanımlamalar görülebilir ve erişilebilir olmasına karşın, paketin "Body" kısmında yer alan tanımlamalar gizlidir ve erişilemez. Şekil 4.3, "Spec" ve "Body" den oluşan bir paketin arayüzünü göstermektedir.



Şekil 4.3 Paket arayüzü

Paketin "Spec" kısmı Şekil 4.4'de de görüldüğü gibi genel tanımlamaları içerir. Bu tanımlamaların kapsamı veritabanı şeması için local, paket için globaldir. Dolayısıyla tanımlanan öğelere herhangi bir başka paketten yada uygulamadan erişilebilir.



Şekil 4.4 Paket kapsamı

Eğer bir paketin "Spec"i sadece tipleri, sabitleri, değişkenleri ve kural dışı durumları tanımlıyorsa bu paket için bir "Body" tanımlamaya gerek yoktur.

"Spec" içerisinde tanımlanan tiplere, öğelere, alt programlara şu notasyonla erişilebilir:

paket_ismi.tip_ismi

paket_ismi.öğe_ismi

paket_ismi.altprogram_ismi

Paketin "Body"si içerisinde, "Spec" içinde yapılan tanımlamalar gerçekleştirilir. "Spec" içinde yer alan her alt program ve cursor'ın tanımlaması "Body" içerisinde yer almak zorundadır. Ancak "Body" içinde yer alan alt programlardan sadece "Spec" içinde tanımlanmış olanlara dışarıdan erişilebileceği unutulmamalıdır.

"Body" sadece o paket içinde kullanılmak üzere tanımlanmış tipleri ve öğeleri içerebilir. Tanımlamalarin kapsamı ise paket içindedir ve local'dir. Bu yüzden paketin "Body"si "Spec"den farklı olarak "Declarative Part" içerir.

Paket içindeki "Declarative Part" seçime bağlıdır. Bu alanda paket içerisinde önceden tanımlanmış değişkenlerin ilk değerlerinin atanması işlemleri yapılır.

Oracle VTYS ve yardımcı programları PL/SQL tabanlı uygulamaların oluşturulmasında yardımcı olması amacı ile üretilmiş özel paketleri içerir.

DBMS_STANDARD : Bu paket Oracle ile kullanıcı uygulaması arasındaki etkileşime yardımcı olmak amacı ile hazırlanmıştır. Mesela;

raise_application_error prosedürü ile kullanıcı tarafından tanımlanmış hata mesajları oluşturulabilir.

DBMS_ALERT : Veritabanında belirli bazı değerlerin değişmesi durumunda uygulamanın bundan haberdar olmasını sağlamak amacı ile hazırlanmış bir pakettir.

DBMS_OUTPUT : PL/SQL bloklarının ve alt programlarının daha kolay test edilmesini sağlamak amacı ile oluşturulmuş bir pakettir.

DBMS_PIPE : Farklı oturumların "pipe" üzerinden birbirleriyle haberleşmesini sağlamak amacı ile oluşturulmuş bir pakettir. Pipe, bir prosesin bir diğerine bilgi aktarması amacı ile kullanılan bir hafıza bölgesidir.

UTL_FILE : PL/SQL programlarının, işletim sistemi text dosyalarını okumasını ve yazmasını sağlamak amacı ile hazırlanmıştır.

UTL_HTTP : PL/SQL programlarda HTTP yapılmasını sağlar. Bu sayede internetten veri alınabilir yada Oracle Web Cartridge çağırılabilir.

DBMS_DDL : DDL komutlarının kullanımı için hazırlanmış bir pakettir.

DBMS_DESCRIBE: Kayıtlı prosedürlerin argümanlarının tanımlanması için hazırlanmış bir pakettir.

DBMS_JOB : Önceden tanımlanmış aralıklarda PL/SQL prosedürlerinin zaman çizelgesinin hazırlanması amacı ile oluşturulmuş bir pakettir.

DBMS_LOB : Büyük nesne işlemlerinin yapılabilmesi için hazırlanmış bir pakettir.

DBMS_SESSION: Uygulamaların oturum karakteristiklerinin kontrol edilebilmesi için hazırlanmış bir pakettir.

DBMS_TRANSACTION: Transaction kontrolünü sağlamak amacı ile oluşturulmuş bir pakettir.[7]

4.3.1.1. Paket kullanmanın avantajları

Modülerlik : PL/SQL olarak isimlendirilen modüllerin içerisinde alt programlar, tipler ve değişkenlerin tanımlanması her bir paketin daha kolay anlaşılabilmesini ve paketler arasındaki arayüzün daha basit, şeffaf ve iyi tanımlanabilmesini sağlar. Bu da uygulamanın geliştirilmesini kolaylaştırır.

Tasarımın basitleştirilmesi : Bir uygulama tasarlanırken başlangıçta tüm ihtiyaç olan arayüz bilgileri bir paketin "Spec" i içerisinde olacaktır. Bu sayede "Spec", "Body" olmaksızın kodu yazılabilir ve derlenebilir. Yani uygulama tamamlanıncaya kadar paketin "Body" sinin tanımlanmasına gerek yoktur.

Bilginin saklanması : Paketlerde; tipler, değişkenler ve alt programlar görülebilir ve erişilebilir olarak yada saklanabilir ve erişilemez olarak tanımlanabilir. Örnek olarak; bir paket üçü public ve biri private olmak üzere dört tane alt program içeriyor olsun. Paket private olan alt programın tanımını saklayacak ve böylelikle tanımlamadaki herhangi bir değişiklik uygulama programını değil, yalnızca paketin kendisini etkileyecektir. Bu da kodun bakımını ve geliştirilmesini kolaylaştırır.

Fonksiyonerliğin arttırılması : Public değişkenler ve cursor'lar oturum boyunca geçerliliklerini korurlar. Dolayısıyla, bunlar bütün alt programlar tarafından kullanılabilir ve erişilebilirler. Böylelikle bunlar veritabanı içerisinde saklanmaksızın veri akışını sağlarlar.

Performansın arttırılması : Paket içerisindeki bir alt program ilk defa çağırıldığında tüm paket hafızaya yüklenir. Aynı paket içerisindeki bir başka alt program çağırıldığında ayrı bir hafıza bölgesine ihtiyaç yoktur. Ayrıca paketler arasındaki ardı ardına gelen ilişkiler durdurulmuştur, bu da gereksiz olarak tekrar derlemeyi engeller. Bir paketin "Body"si içerisindeki alt programın değiştirilmesi onu çağıran pakette bir değişikliğe neden olmaz. Bu durumda sadece değişikliğin yapıldığı paketin "Body"sinin derlenmesi yeterlidir. [7]

4.3.2. Saklanmış prosedürler

Özellikle veritabanından veri okuma yada veritabanına veri yazma işlemlerinde kullanılan saklanmış prosedürler; veritabanında saklanan kod bloklarıdır. Veritabanı dışındaki uygulamalar tarafından çağırılarak kullanılırlar ve bu uygulamalar ile veritabanı arasında bir arayüz oluştururlar.

Veritabanında yer alan bu işletilebilir kod parçaları sayesinde ağ üzerindeki trafik azaltılmış olur. Veritabanında yapılacak sorgular kolaylaşır ve uygulama programları doğrudan sorgu sonucunu elde edebilirler. Microsoft Access gibi dosya tabanlı sistemlerde bir sorgu yapılmak istendiği zaman uygulama

programına tüm bir kayıt verilir. Oysaki saklanmış prosedürler; veritabanında gerekli sorguyu SQL komutları ile yapar ve sonucu tek bir kayıt ile uygulama programına verir.

Uygulama programı içerisinde işlem yapılmasını sağlayacak karmaşık bir kod parçacığı ağ üzerinde gereksiz kod akışına neden olmaktadır. Saklanmış prosedürler ağ üzerindeki kod akışını azaltır. Veritabanı üzerinde işletilebilir kod parçaları olması, hızın da artması anlamını taşır.

Kullanıcılara saklanmış prosedürler içerisinde yetki verilebilir. Bu şekilde her kullanıcının her veriye erişimi engellenmiş ve veri bu şekilde daha iyi korunmuş olur. Aynı saklanmış prosedür birden fazla kullanıcı tarafından kullanılabilir. Oracle bu durumlar için saklanmış prosedürün sadece bir kopyasını oluşturarak bu hafıza bölgesini kullanıcılara tahsis eder. Bütün kullanıcılar da bu kopyayı paylaşırlar. Bunun anlamı Oracle'ın kullanıcının hafızayı en etkin bir biçimde kullanmasını sağlamaktır.

4.3.3. Cursor

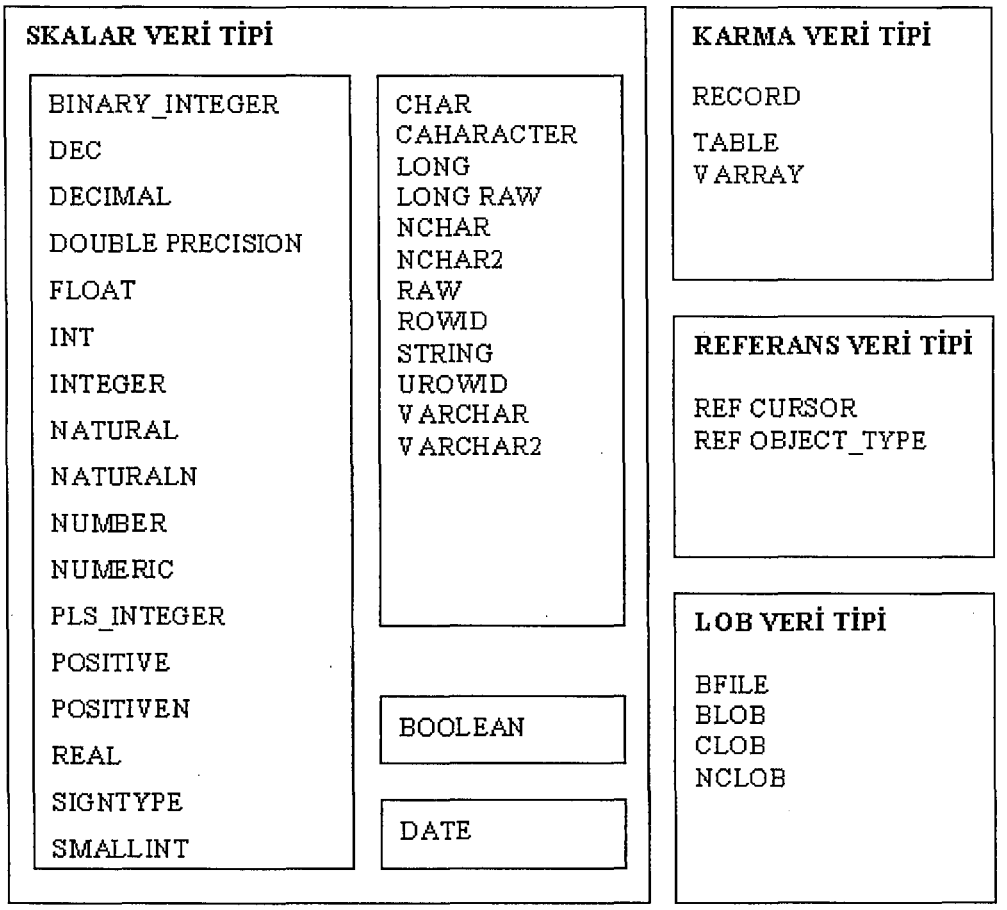
Bir SQL komutu kullanılarak yapılan sorgulamada, birden fazla kayıt yapılan sorgulamanın koşullarını sağlıyorsa; bu koşul yada koşullara uyan ilk kayıt sorgulama sonucunda döner. Uygulama programlarında bazen sorgulama koşullarına uyan tüm kayıtlara ulaşılmak istenebilir. İşte cursor sorgulama sonucunda oluşan bütün sonuç kümelerinin kullanıma açık hale getirecek mekanizmayı oluşturur ve böylece sonuç kümesi içerisinde yer alan bütün kayıtlar ardışık bir dosyadan okunuyormuşcasına kullanılabilir.

Cursor'lar kullanılarak sonuç kümesi içerisinde arzu edilen kayıtlar üzerinde pozisyon almak, mevcut pozisyondan itibaren bir yada daha fazla kayıt bilgisini elde etmek, sonuç kümesinin mevcut pozisyonunda bulunan kaydına ait bilgiyi değiştirmek ve diğer kullanıcılar için veri değişimi seviyesini belirlemek mümkündür.

PL/SQL'de cursor'lar "örtülü" ve "belirtilmiş" olmak üzere iki çeşittir. Sadece tek bir kayıt döndüren SQL ifadeleri için PL/SQL tarafından otomatik olarak örtülü cursor tanımlanır. Birden fazla kayıt döndüren SQL ifadeleri tanımlamak için ise belirtilmiş cursor tanımlamak gerekmektedir.

4.3.4. Veri tipleri

PL/SQL'de kullanılabilecek veri tipleri skalar, LOB, referans ve karma veri tipleri olmak üzere dört ana gruba ayrılabilir. PL/SQL'de veri tipleri Şekil 4.5'de verilmiştir.



Şekil 4.5 PL/SQL'de veri tipleri

5. YAZILIM PROJELERİ TAKİP PROGRAMI

YPTP programının geliştirilmesi sırasında Oracle8i veritabanı yönetim sistemi, Oracle8i'e ait yardımcı programlar ve Microsoft Visual Basic 6.0 programlama dili kullanılmıştır. Tüm tablolar, görüntüler Oracle8i veritabanı içerisinde saklanmaktadır. Veritabanı ve veritabanı üzerindeki saklanmış prosedürler sunucu bilgisayar üzerinde bulunurlar. Visual Basic ile yazılmış istemci üzerinde bulunan arayüz programı aracılığı ile bu saklanmış prosedürlere ve dolayısıyla da veritabanındaki bilgilere erişim sağlanmış olur. İstemci/sunucu mimarisine göre hazırlanmış olan YPTP uygulama programı kendisi için gerekli olan sorgulamaları yapar, verilere erişimi sağlayarak onları gerekli yerlerde kullanır ve saklanması gereken verileri de veritabanına kaydeder.

Veritabanının tasarımı yapılırken biçimsel tekniklerin gözardı edilerek kullanılmaması ileriki aşamalarda büyük hataların ortaya çıkmasına neden olur. Küçük veritabanı uygulamalarında dizayn tekniklerine uymamak çok önemli olmayabilir ancak; veritabanı büyüdükçe bu tekniklerin kullanılmaması; anlaşılması, ilişkilendirilmesi ve geliştirilmesi zor karmaşık bir veritabanının oluşmasına neden olur. Oysaki veritabanı tasarımında bir çok kişi çalışıyor olabilir, hatta bu kişilerin veritabanı tasarımı, bakımı ve uygulaması gibi farklı görevleri olabilir ve farklı konularda aynı alanda çalışacak bu kişilerin hepsinin amacına yönelik bir veritabanın oluşturulması gerekir. Veritabanını kullanacak programın daha sonraki dönemlerde yeni ihtiyaçlara cevap verebilecek bir program olması isteniyorsa veritabanı tasarımında daha dikkatli davranmak gerekir.

YPTP programının veritabanı tasarımı yapılırken formal teknikler uygulandığından dolayı bu tekniklere değinmek uygun olacaktır. Veritabanı tasarımını üç farklı faza ayırmak mümkündür. Bunlar :

Konseptör Tasarım : Kullanıcı topluluğu ile etkileşim sonucunda ortaya çıkan tüm ihtiyaç tanımlamaları alınarak donanımın ve yazılımın gereksinimlerinden bağımsız bir tasarım yapılmasıdır. Varlık-İlişki modeli veritabanı tasarımında çok popüler olan bir tekniktir.

İyi bir veritabanı tasarımı için veri modellemesi olarak adlandırılan bir süreç kullanılır. Veri modellemesi gerçek dünyanın nesnelerinin ve bu nesneler

arasındaki ilişkilerin soyut olarak gösterilmesini sağlayan bir veri modelidir. Konseptör fazında veri ihtiyaçları tanımlanır, veri akış diyagramı hazırlanır, eğer mümkünse durumlar arasındaki geçiş diyagramları hazırlanır, veriler varlık-ilişki diyagramlarında gösterilir, varlıkların özellikleri ile diyagram tanımlanır.

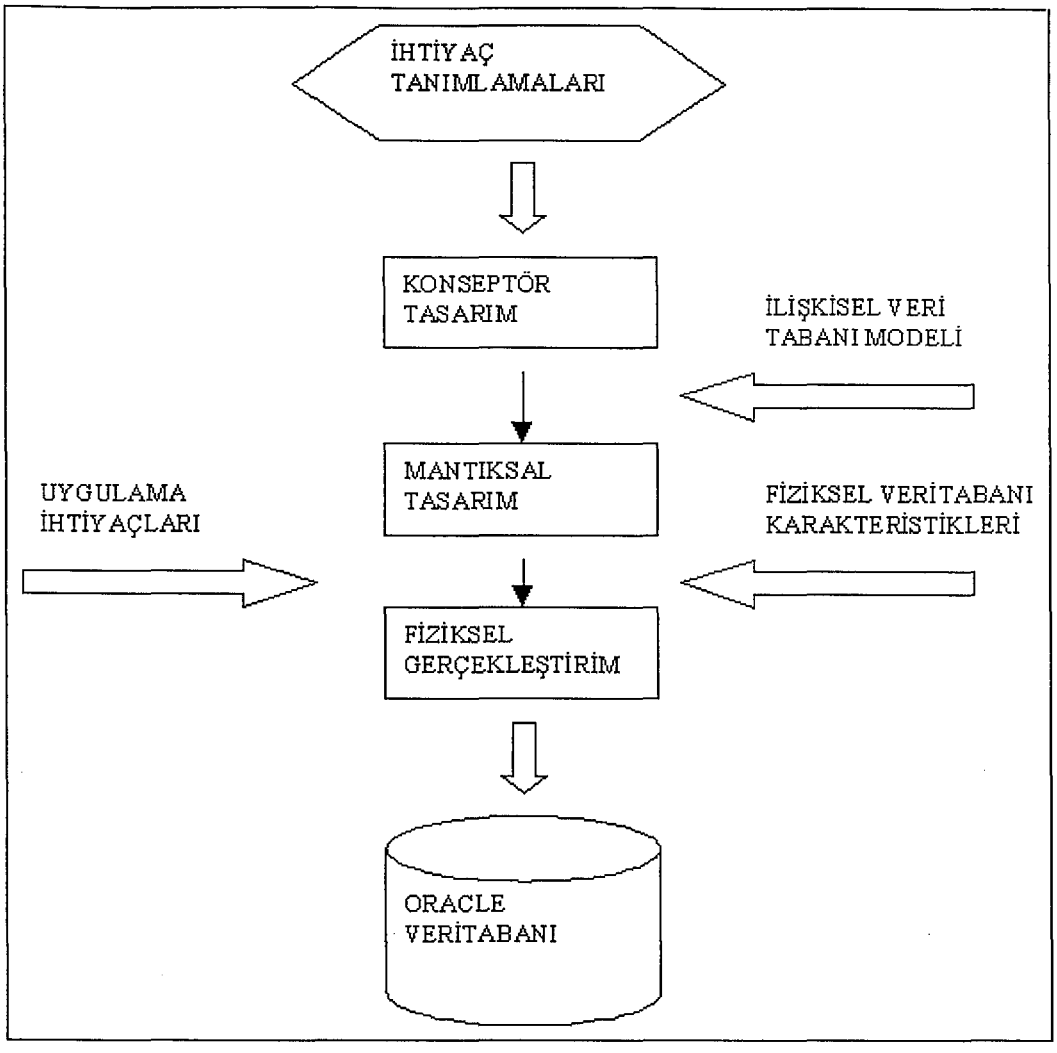
Varlık; gerçek dünyadaki nesnelerin veritabanında tanımlanmış şeklidir ve her bir nesneye o nesneyi tanımlayacak özellikler atanabilir. Özellikler; bir varlık hakkında korunması istenen ek bilgilerdir. Konseptör modellemede dikdörtgenler varlıkları, karolar ise varlıklar arasındaki ilişkileri gösterir. Özellikler bir varlığın daha detaylı olarak tanımlanmasını sağlar. Özellikler tanımlanırken varlıkları tanımlayan ve varlıklar hakkında saklanmak istenen tüm bilgilerin düşünülmesi gerekmektedir. Özellikler, varlıklara bağlı elips şekliyle gösterilir. Varlıklara ait özellikler o varlığı gösteren dikdörtgenin içine de yazılarak gösterilebilirler. Konseptör modelleme yardımı ile varlıklar, özellikler ve varlıklar arasındaki ilşkiler daha açık bir şekilde gözükür ve bu sayede tasarımcı birleştirilmesi, parçalanması gereken varlık yada ilişkileri daha iyi görerek tasarımını değiştirme ve iyileştirme imkanı bulabilir.

Veritabanı tasarımında gerçekleştirilen tüm fazlar şekil 5.1'de bir özet halinde verilmiştir.

Fiziksel Tasarım : Konseptör tasarım modelinin kullanılacak DBMS tipine uygun olarak mantıksal modele dönüştürülmesi işlemidir. Veri yedeklemesi ve bakım gereksinimlerinin minimum düzeyde göz önüne alınarak tasarım yapılmasıdır.

İyi bir konseptör modelleme yapıldıktan sonra mantıksal modellemeye geçilir. Mantıksal tasarımda;

- Kayıt tipleri tanımlanır
- Bu kayıtlar içindeki alanlar tanımlanır
- Herhangi bir verinin bağımlılıkları tanımlanır
- Veritabanı normalleştirilir
- Anahtarlar tanımlanır



Şekil 5.1 Veritabanı tasarım süreci

Varlıklar, konseptör tasarımdan direk olarak mantıksal tasarıma geçirilebildiği halde bu, ilişkiler için geçerli değildir. İlişkiler mantıksal tasarıma aktarılırken yeni özelliklerin ve varlıkların oluşturulması gerekebilir. Bu yeni özellikler genellikle bir tablodan bir başkasına referans istemi ortaya çıktığında oluşur ve bunun içinde bir dış anahtar tanımlanması gerekir.

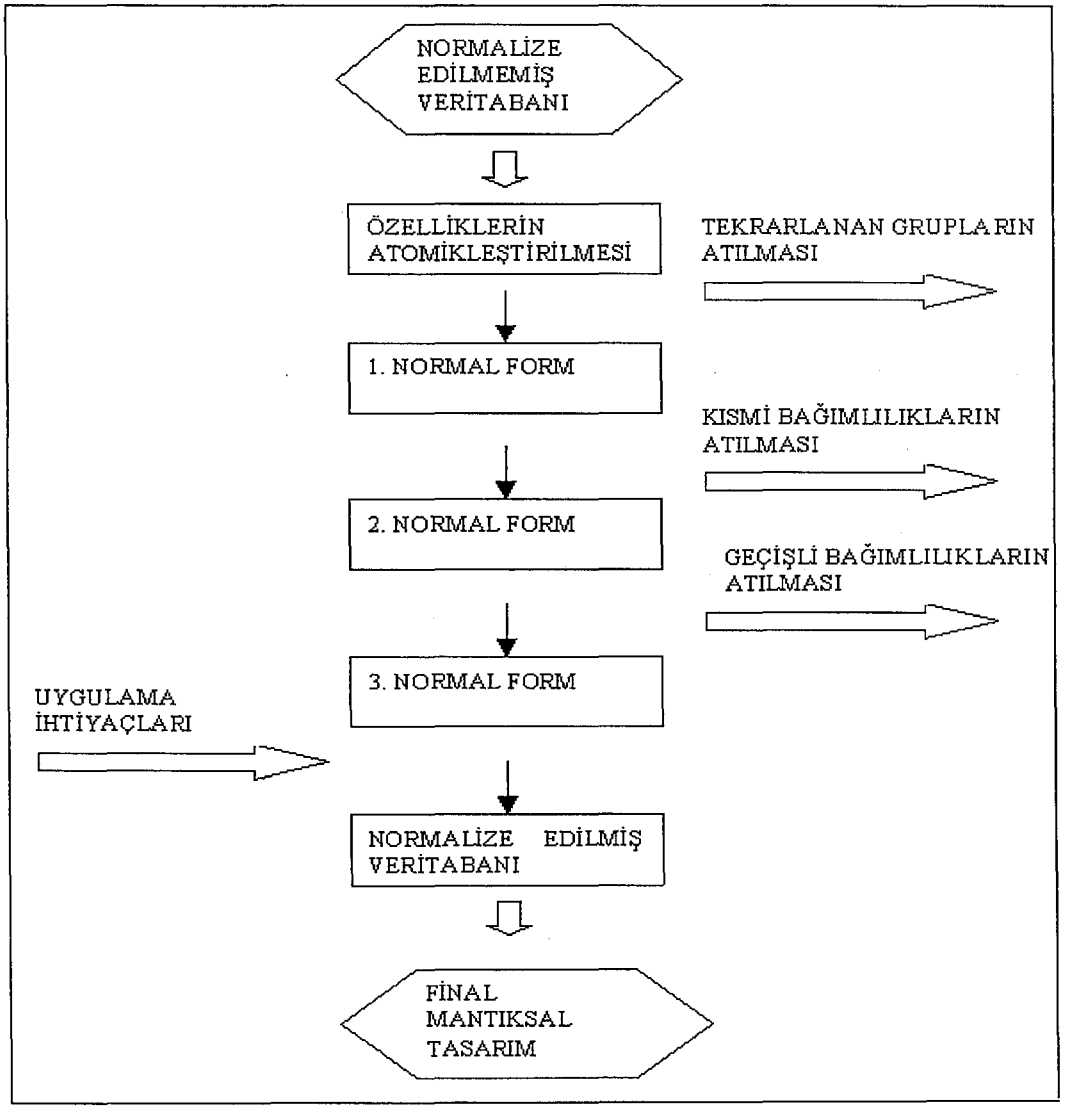
Normalleştirme basit ilişki grupları içinde karmaşık hale gelen tasarımları azaltmayı sağlayan bir işlemdir. Normalleştirme, veritabanının;

- Bakımını kolaylaştırır
- Minimum iş yükü ile veritabanının büyütülmesini sağlar
- Verinin gereğinden fazla tekrarlanmasını engeller
- Verilerdeki tutarsızlığı azaltır

- Veritabanında herhangi bir değişiklik gerektiği zaman bu değişikliğin minumum sayıda varlığı, özelliği ve ilişkiyi etkilemesini sağlar

Normalleştirmenin sağlanması için veritabanında bütün özellikler atomik olmalı, tekrarlanan alanlar, kısmi ve geçişli bağımlılıklar olmamalıdır. Normalleştirme ve gerçekleştirilen işlemler Şekil 5.2'de verilmiştir.

Birincil normalleştirmede, bir tabloda yer alan tekrarlanan alanlar kaldırılmalı bunlar için farklı bir tablo tanımlanmalıdır. İkinci normalleştirmede kısmi bağımlılıklar atılarak farklı tablolar tanımlanmalı ve de üçüncü mormalleştirmede geçişli bağımlılıklar atılarak farklı tablolar tanımlanmalıdır.

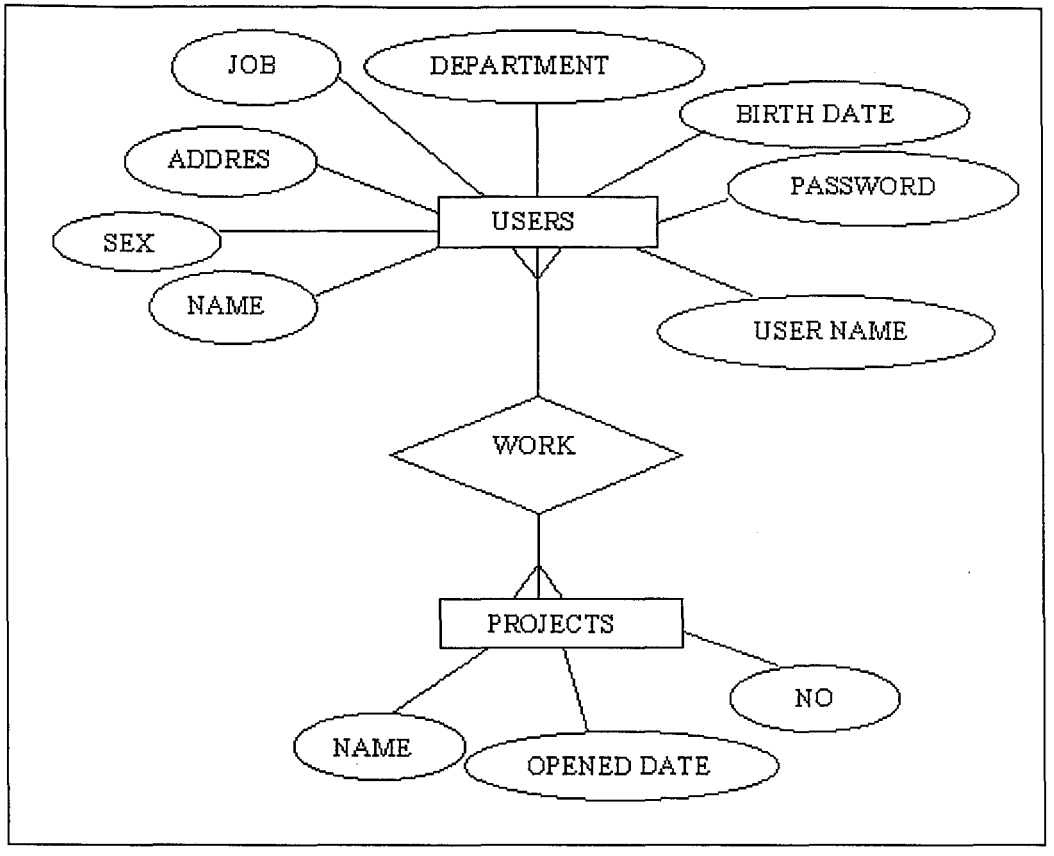


Şekil 5.2 Normalizasyon işlemleri

Fiziksel Gerçekleştirim : Veritabanı tasarımının son fazıdır. Veritabanının uygulama programına daha uygun hale getirilmesi için fiziksel veri tablolarının gerçekleştirilmesi ile yapılan tasarımlardır.

5.1. YPTP Konseptör Tasarımı

YPTP programı; IEEE/EIA 12207'ye uygun bir çok yazılım projesinin takibinin yapılması amacı ile oluşturulmuştur. Konseptör tasarımın yapılabilmesi için kullanıcı ihtiyaçlarının tam ve açık olarak tanımlanması gerekmektedir. Bu uygulama programının ilk amacı üzerinde çalışılmakta olan tüm projeleri ve hangi projelerde hangi elemanların çalıştığı bilgisinin saklanmasıdır. Konseptör tasarıma başlandığında ortaya çıkan ilk varlık bu nedenle projeler varlığıdır. Projeler için çalışan kişiler de ikinci varlığı ortaya koymaktadır. Her projenin bir ismi, numarası ve proje açılış tarihi vardır. Bunlar proje varlığına ait özelliklerdir. Kullanıcı nesnesinin özellikleri ise isim, cinsiyet, doğum tarihi, adres, meslek, çalıştığı bölüm, yaptığı iş ve programı kullanmak için sahip olduğu kullanıcı ismi ve şifresidir. Konseptör şemanın iki varlığı olan projeler, kullanıcılar ve bunların özellikleri Şekil 5.3'de gösterilmektedir.

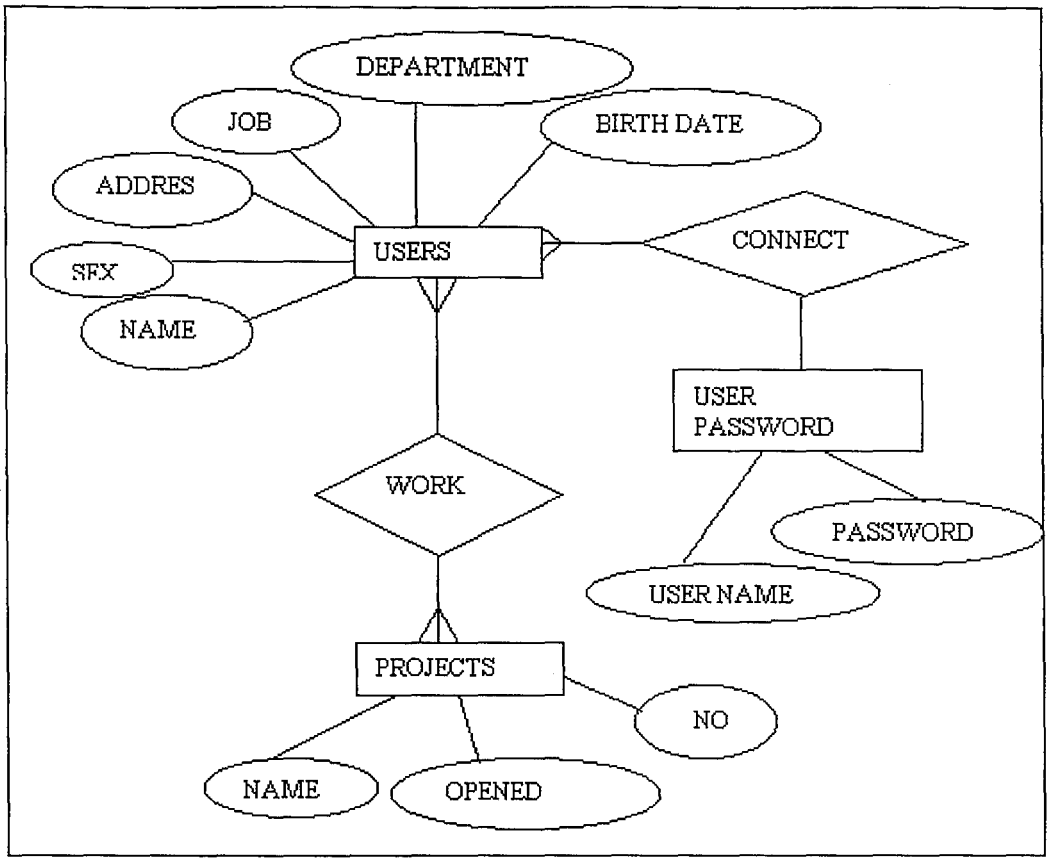


Şekil 5.3 Proje ve kullanıcı nesnelerinin konseptör tasarımla gösterimi

Şekil 5.3'de görüldüğü gibi proje nesnesi ile kullanıcı nesnesi arasında her iki istikamette çoklu ilişki bulunmaktadır. Çalışma olarak isimlendirilen bu ilişki bir projede birden fazla kişinin çalışabileceğini ve bir çalışanın farklı projelerde görev alabileceğini göstermektedir.

Kullanıcıların adres bilgisi, gerçekleştirilen konseptör tasarımda tek bir özellik olarak tutulabileceği gibi; mahalle, sokak, şehir, posta kodu gibi ayrı ayrı özellikler olarak da tutulabilirdi. Tek bir özellik olarak seçilmesinin nedeni uygulama programında bu bilgilere ayrı ayrı erişimin gerekli olmamasından kaynaklanmaktadır. Yani bu bilgi için türetilmiş nesneye ihtiyaç yoktur.

Tek başına Şekil 5.3'deki konseptör tasarım dikkatlice incelendiğinde ise kullanıcı adı ve şifresinin bir kullancıya bağımlı olduğu görülmektedir. Bu iki özellik için yeni bir zayıf varlık tanımlamak daha doğru olacaktır. Buna göre konseptör tasarımın yeni alacağı durum şekil 5.4'de olduğu gibi olacaktır.



Şekil 5.4 Proje ve kullanıcı nesnelerinin konseptör tasarımla gösterimi

Tasarımda yapılan bu değişikliğe göre kullanıcı şifresi adında yeni bir varlık oluşturulmuştur. Önceki tasarımda kullancıya ait olan kullanıcı ismi ve kullanıcı şifresi özellikleri ise bu yeni varlığa atanmıştır.

Yeni bir varlığın ortaya çıkması beraberinde yeni bir ilişkiyi getirmiştir. Çalışma olarak adlandırılan bu ilişki bir yönde çoklu ilişkidir. Her bir kullanıcı uygulama programını kullanmasını sağlayacak bir kullanıcı ismi ve kullanıcı şifresine sahipken, her kullanıcı şifresi ile isminin tek bir sahibi vardır.

YPTP'nin veritabanı oluşturulurken tüm ihtiyaçlar açık ve kesin bir şekilde tanımlanmış, daha sonra bu ihtiyaçlar doğrultusunda konseptör tasarım yapılmıştır. Ancak veritabanı yapısı çok geniş olduğundan yapılan konseptör tasarımların tümüne değinilmeyecek, sadece örnekler verilerek geçilen aşamalar anlatılacaktır.

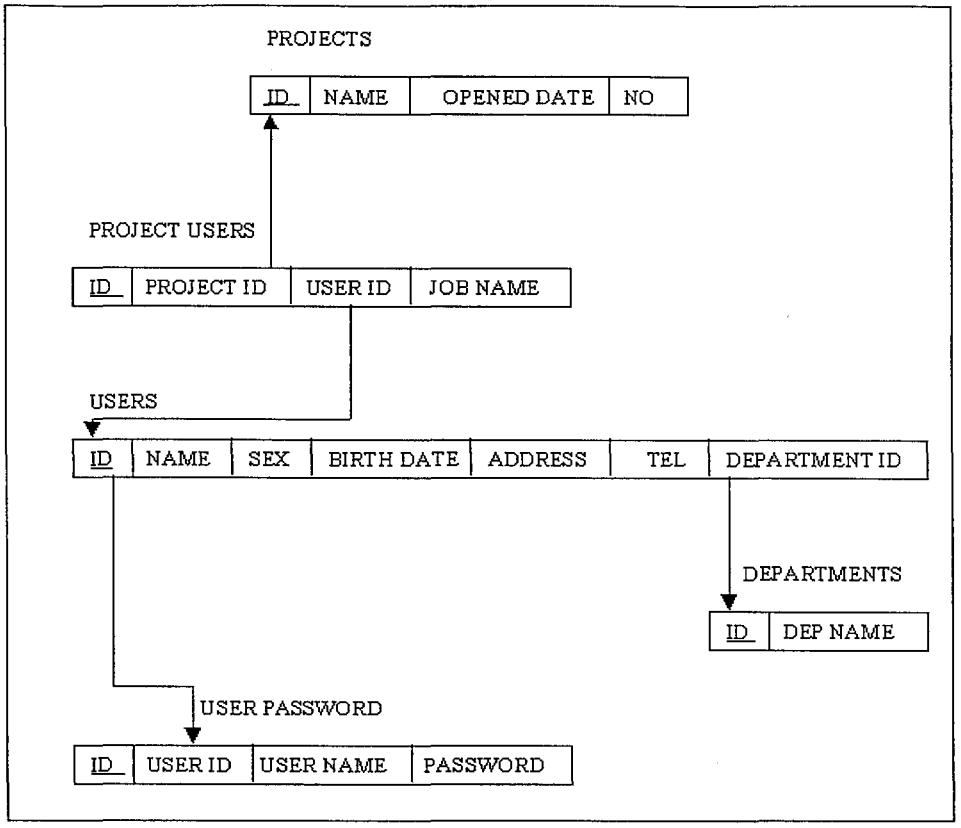
5.2. YPTP Mantıksal Tasarımı

Konseptör tasarımı yapılan proje, kullanıcı ve kullanıcı şifresi nesnelerinin konseptör tasarımdan mantıksal tasarıma geçişi yapılırken varlık sayısının aynı kalmasının beklenmesi hata olur. Daha önce de bahsedildiği gibi varlıklar direk olarak mantıksal tasarıma aktarılabilirken bu ilişkiler için mümkün olmaz ve yeni varlıkların tanımlanması gerekebilir.

Mantıksal tasarımda Şekil 5.5'de de görüldüğü gibi varlıklar kayıtlara çevrilir ve varlıklara ait özelliklerin her biri de bu kayıtların alanları olarak tanımlanır.

Konseptör tasarımda proje ve kullanıcılar arasındaki çalışma ilişkisi mantıksal tasarıma proje kullanıcıları ismiyle tanımlanan bir kayıt olarak geçmiştir. Bu kayıt kullanıcılarla projeleri birbirine bağlayan bir kayıt olmakla birlikte aynı zamanda kullanıcının yaptığı işi de bir özellik olarak üzerinde taşımaktadır.

Mantıksal tasarımda kayıtlar arasındaki bağımlılıkların oluşturulması ve bunu sağlayacak anahtarların tanımlanması gerekir. Şekil 5.5'de görüldüğü gibi her bir kayıt için ID isminde birincil anahtar tanımlanmıştır. Projeler kaydındaki birincil anahtar, proje kullanıcıları kaydında tanımlanan PROJECT_ID ismindeki yabancı anahtarla bağlantılıdır. Bu sayede her bir kullanıcının hangi projede ve hangi görevle çalıştığı bilgisi tutulmaktadır. Kullanıcı şifreleri kaydında da aynı durumla karşılaşılmaktadır. Burada tanımlanan USER_ID ismindeki yabancı anahtar kullanıcı kaydındaki birincil anahtara bağlanmıştır. Böylece hangi kullanıcı isminin ve kullanıcı şifresinin hangi çalışana ait olduğu saklanabilmektedir.



Şekil 5.5 Proje ve kullanıcı nesnelerinin mantıksal tasarımla gösterimi

Mantıksal tasarımda bağımlılıklar da tanımlandıktan sonra oluşturulan tasarım normalleştirilmelidir. Konseptör tasarımda çalışanlara ait bir özellik olarak tanımlanan bölümler mantıksal tasarıma aktarıldığı zaman bilgi tekrarlanması oluştuğundan dolayı bunun 1NF'e uygun olmadığı görülmektedir. Bu nedenle bölümler, mantıksal tasarıma aktarılırken ayrı bir kayıt olarak tanımlanmıştır. Kurumun içindeki tüm bölüm isimlerini tutan bu kayıt ID ismiyle tanımlanan birincil anahtarı sayesinde kullanıcılar kaydına bağlanmıştır. Bu sayede projede çalışan her bir kişinin çalıştığı bölüm bilgileri tutulabilmektedir. Mantıksal tasarım 2NF ve 3NF formlarına göre incelendiğinde bu formlara uygun olduğu görülmüştür.

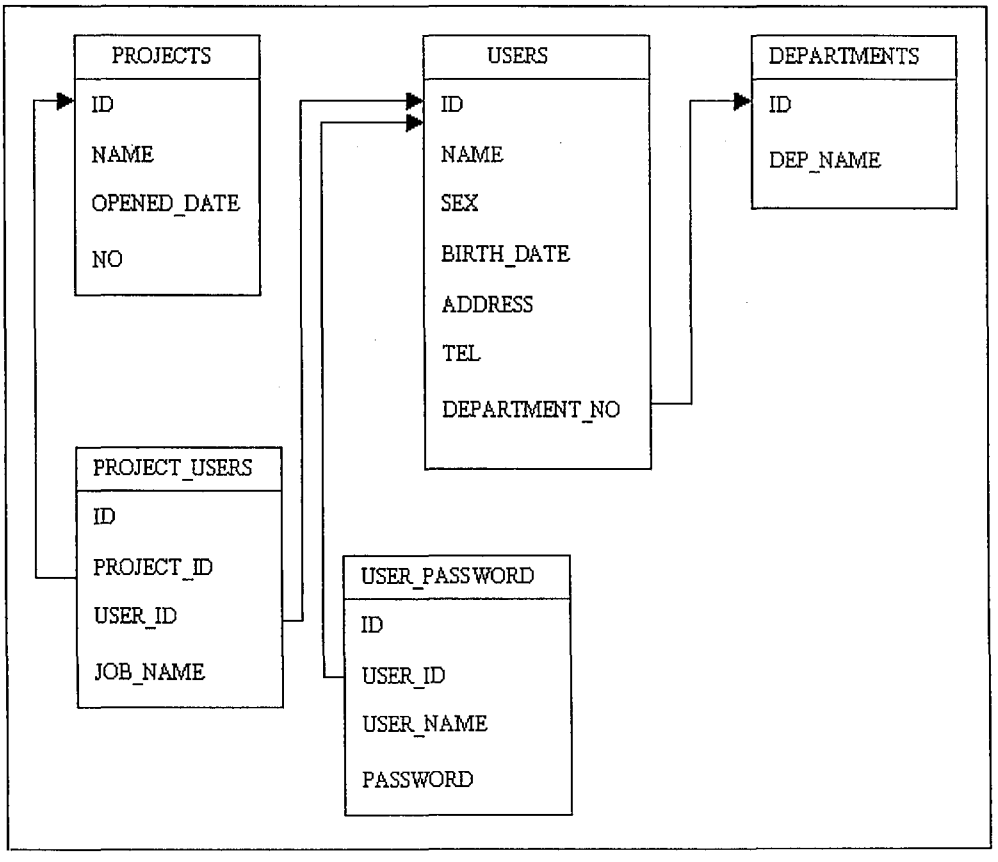
5.3. YPTP Fiziksel Gerçekleştirim

Fiziksel gerçekleştirim, veritabanı tasarımının en son aşamasıdır. Fiziksel gerçekleştirim sırasında Oracle8i veritabanı yönetim sistemi ve araç yazılımları kullanılarak, mantıksal tasarımda hazırlanan kayıtlar birer tabloya dönüştürülür.

Mantısal tasarımda belirlenen özellikler bu aşamada her bir tablonun alanı olarak tanımlanır, sınırları ve tipleri belirlenir. Alanlardan uygun olanları birincil yada yabancı anahtar olarak tanımlanarak tablolar arasındaki ilişkiler tanımlanır. Bu tasarımda veritabanı YPTP uygulama programının fiziksel gerçekleştirimi, konseptör ve mantıksal tasarıma göre daha detaylı anlatılacak tabloların alanları, alanların tipleri ve tablolar arasında oluşturulan ilişkiler verilecektir.

Veritabanının fiziksel gerçekleştirim sırasında oluşturulan ilk tabloları; konseptör ve mantıksal tasarımı verilen, projeler, kullanıcılar, şifreler ve bölümler tablolarıdır.

Şekil 5.6'da görüldüğü gibi "Projects" tablosu açılan tüm projeleri; isimleri, açılış tarihlerini ve proje numarası bilgileri ile içerisinde tutmaktadır. "Users" tablosu ise tüm kullanıcıları isim, cinsiyet, doğum tarihi, adres, telefon numarası bilgileri ile saklamaktadır. "Departments" tablosunun içinde kurum içinde bulunan tüm bölümler, "User_Password" tablosu içerisinde kullanıcı adı ve



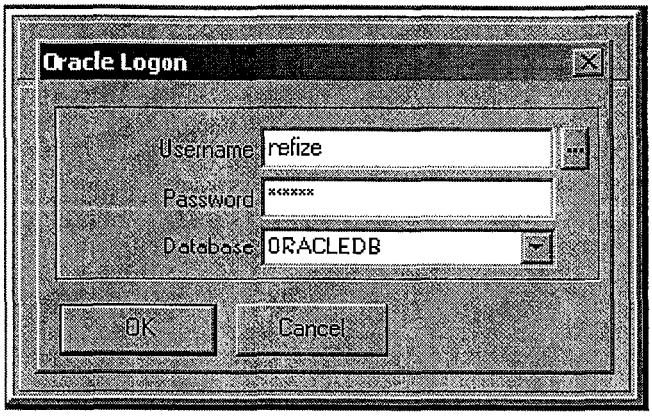
Şekil 5.6 Proje ve kullanıcı nesnelerinin fiziksel gerçekleştirimi

"Project_Users" tablosunda ise kullanıcı görev bilgileri saklanmaktadır. Bu bilgilerin her biri ait oldukları tablolardaki alanları ifade eder. "Project_Users" tablosu her bir projede çalışan elemanlar ve onlara ait bilgilere erişilmesini sağlar.

Tüm tablolarda "ID" isimli bir birincil anahtar tanımlanmıştır. Bu anahtarlar ait olduğu tablonun diğer tablolar ile bağlantı oluşturulmasını dolayısıyla aralarında bir ilişki kurulabilmesini sağlar. Şekil 5.6'ya bakıldığında "User_Password" ve "Project_Users" tablolarındaki "ID" isimli birincil anahtarların gereksiz olarak tanımlandığı düşünülebilir ancak unutulmamalıdır ki hazırlanan veritabanı geliştirilmeye açık olmalı ve her zaman yeni eklenebilecek bir tablo ile mevcut tablonun arasında bir ilişki kurma gereksinimine hazır olmalıdır. Birincil anahtarların bağlandığı "Project_ID", "User_ID" ve "Department_ID" alanları birer yabancı anahtar olarak tanımlanmıştır. Bu anahtarlar da daha önceden tanımlanan birincil anahtarlara referans verilerek iki tablo arasındaki ilişkinin oluşturulmasını sağlar.

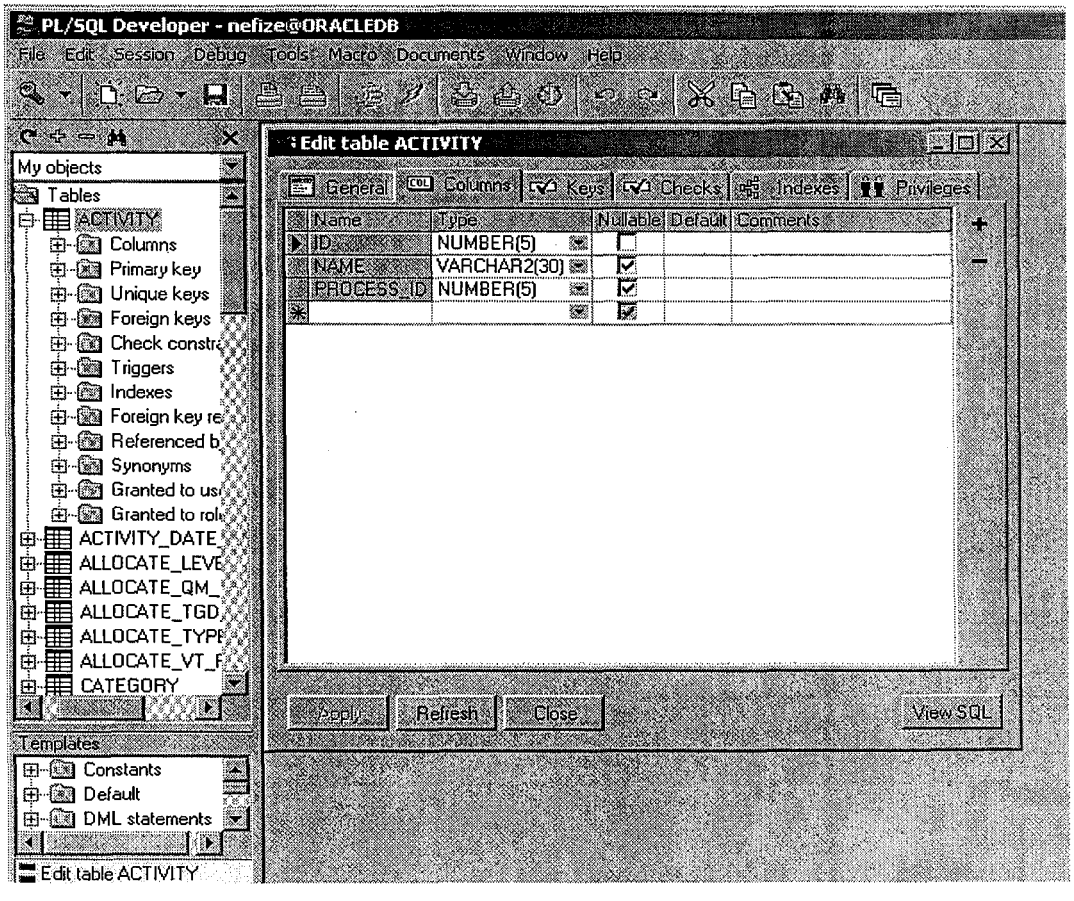
Fiziksel gerçekleştirimin yapılabilmesi için Oracle8i'ye ait bir araç program olan PL/SQL Developer 8.1 kullanılmıştır. PL/SQL Developer Oracle veritabanı üzerinde tabloları, görüntüleri, bunların alanlarını, alanların tip ve sınırları, kullanıcı tiplerini, indeks ve sıraları tanımlamayı sağlar. Bu araç yazılım sayesinde kullanıcı ayrıca veritabanını kullanacak kişilere yetki vererek rollerinin de tanımlanmasını sağlayabilir. Veritabanına gömülen saklanmış prosedür ve fonksiyonlar hazırlanabilir, PL/SQL kodlar kolayca yazılarak derlenip kullanılabilir hale getirilebilir. Adından da anlaşılabileceği gibi PL/SQL Developer, veritabanının geliştirilmesi sırasında kullanılabilecek ideal bir araç yazılımdır.

PL/SQL Developer, kullanıcının isteğine göre tüm sistem nesneleri yada sadece bu kullanıcı tarafından tanımlanan nesneleri gösterir. Oracle veritabanı üzerinde birden fazla veritabanının tanımlanmış olması durumunda Şekil 5.7'de olduğu gibi hangi veritabanına girilmek istendiğini soran bir açılış penceresi görülür. Yetkisi olması durumunda kullanıcı istediği veritabanına ve veritabanında tanımlanan tüm şema nesnelerine yetkileri sınırında erişebilir.



Şekil 5.7 PL/SQL giriş penceresi

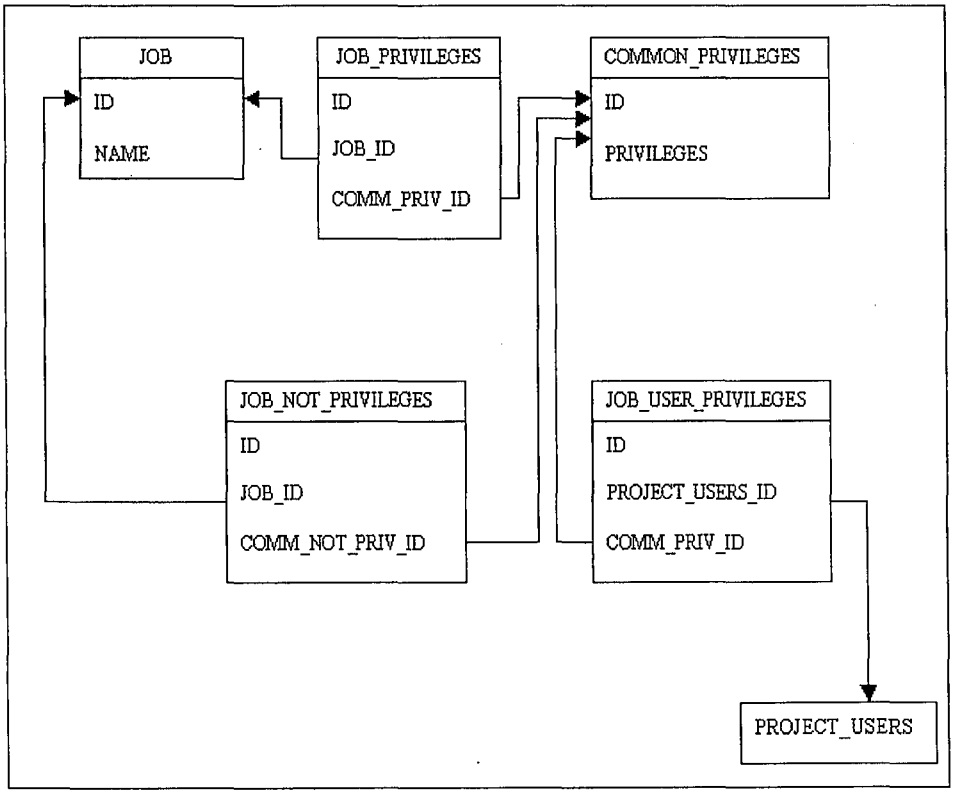
Şekil 5.8'de ise PL/SQL Developer'da tanımlanan bir tablo ve ona ait alanlar özellikleri ile birlikte görülmektedir.



Şekil 5.8 PL/SQL developer'da tablo ve alanların tanımlanması

PL/SQL Developer üzerinde bir tablo, görüntü, sıra yada indeks tanımlandığında otomotik olarak bu tanımlamanın SQL kodu oluşturulup kullanıcıya gösterilir. Kullanıcı isterse bu SQL ifadeyi elle de değiştirebilir.

YPTP program ihtiyaçlarından biri de projelerde istenilen kişilere istenilen yetkilerin proje yöneticileri tarafından tanımlanabilmesidir. Proje gruplarında proje teknik yöneticisi, yazılım grup lideri, sistem grup lideri, yazılım mühendisi, sistem mühendisi gibi görev tanımlamaları vardır. Bu kişilerin her birinin proje içinde yapması gereken görevler farklıdır. Yazılım mühendisi proje teknik yöneticisinin yaptığı bilgilere ulaşamayabilir, ikisinin de görev tanımları birbirinden çok farklıdır. Oluşturulan veritabanında bu gibi yetkilendirmeleri saklayacak tabloların tanımlanmasına ihtiyaç vardır. Konseptör ve mantıksal tasarımlarına değinilmeyecek olan bu tablo ilişkilerinin fiziksel gösterimi Şekil 5.9'da verilmiştir.



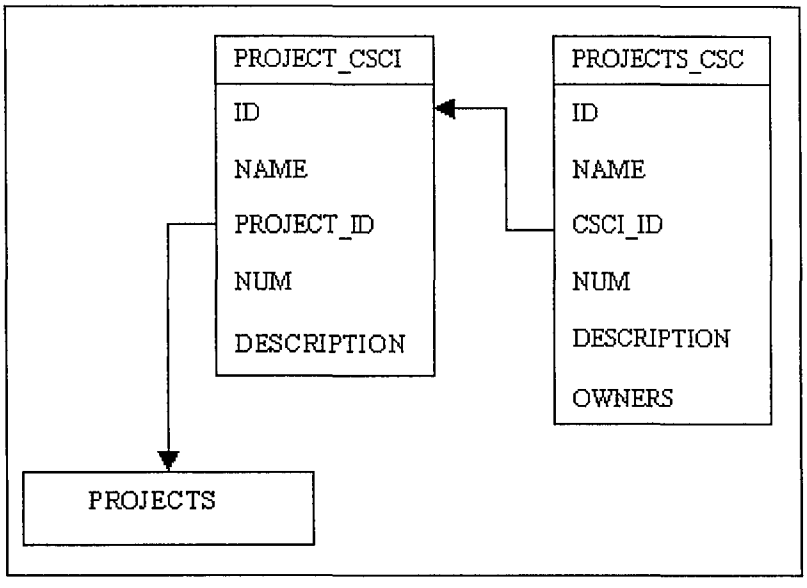
Şekil 5.9 Yetkilendirme için oluşturulan tablolar ve ilişkileri

Şekil 5.9'da görülen "JOB" tablosu bir yazılım projesinde yer alabilecek tüm görevleri içeren bir tablodur. Bu tablonun içeriği belirlenmiş ve fiziksel gerçekleştirim sırasında doldurulmuştur. Daha sonra hazırlanacak olan uygulama programı tablonun içeriğini değiştiremeyecektir. "JOB" tablosunun içinde teknik yönetici, sistem mühendisi, yazılım mühendisi, yazılım kalite mühendisi, yazılım grup lideri ve avyonik şube müdürü bilgileri saklıdır. "Common_Privileges" tablosu da "Job" tablosu ile aynı özelliklere sahip bir tablodur ve içinde kullanıcılara verilebilecek tüm yetkiler saklanmaktadır. YPTP'de tanımlanan ihtiyaçlarından biri, bazı görev sahiplerinin bazı yetkilere mutlaka sahip olacağı bununla birlikte bazılarına da asla sahip olamayacağı yönündedir. Bu sebeple Şekil 5.9'da görülen "Job_Privileges" ve "Job_Not_Privileges" tabloları tanımlanmıştır ve buradaki bilgiler de fiziksel gerçekleştirim sırasında doldurulmuştur. Mesela; bir teknik yönetici her durumda proje takvim güncellemesi yapabilirken, hiçbir durumda bir yazılım bileşeni tanımlayamaz yada güncelleştiremez. Bir proje görevlisinin her projede sahip olacağı tüm yetkiler "Project_Privileges" tablosunda saklanırken, hiçbir zaman kullanamayacağı yetkiler de "Job_Not_Privileges" tablosunda tanımlanmıştır. "Common_Privileges" tablosunda bulunan diğer yetkiler proje teknik yöneticileri ve avyonik şube müdürü tarafından proje çalışanlarına atanır ve bu sonuç atamalar da "Project_User_Privileges" tablosunda saklanır.

Her bir yazılım projesi bir yada daha fazla yazılım bileşeninden oluşur. CSCI olarak da isimledirilen yazılım bileşeni, konfigürasyonu takip edilen yazılım elemanıdır. Her bir yazılım bileşeni de bir yada daha fazla yazılım biriminden oluşur. CSC olarak adlandırılan yazılım bileşenleri ise sistem bazında tanımlanmış bir fonksiyonu yerine getiren yazılım üniteleri kümesidir. Yazılım bileşenlerinin içinde ise yazılım üniteleri yani paketler yer alır. YPTP programının ihtiyaçlarından birisi de bir yazılım projesinde bulunan CSCI ve CSC tanımlanabilmesi, saklanabilmesi ve değiştirilebilmesidir. Şekil 5.10'da gösterilen tablolar bunu sağlamak için tasarlanmıştır.

"Project_CSCI" tablosu her bir yazılım projesinde tanımlanan CSCI'ları isim, numara ve bu CSCI hakkındaki açıklama bilgileriyle birlikte tutar. "Project_CSC"

tablosu ise bir CSCI'a ait olan tüm CSC'leri isim, numara, açıklama ve sahibi olan kişi yani geliştiricisi bilgileri ile birlikte saklar.

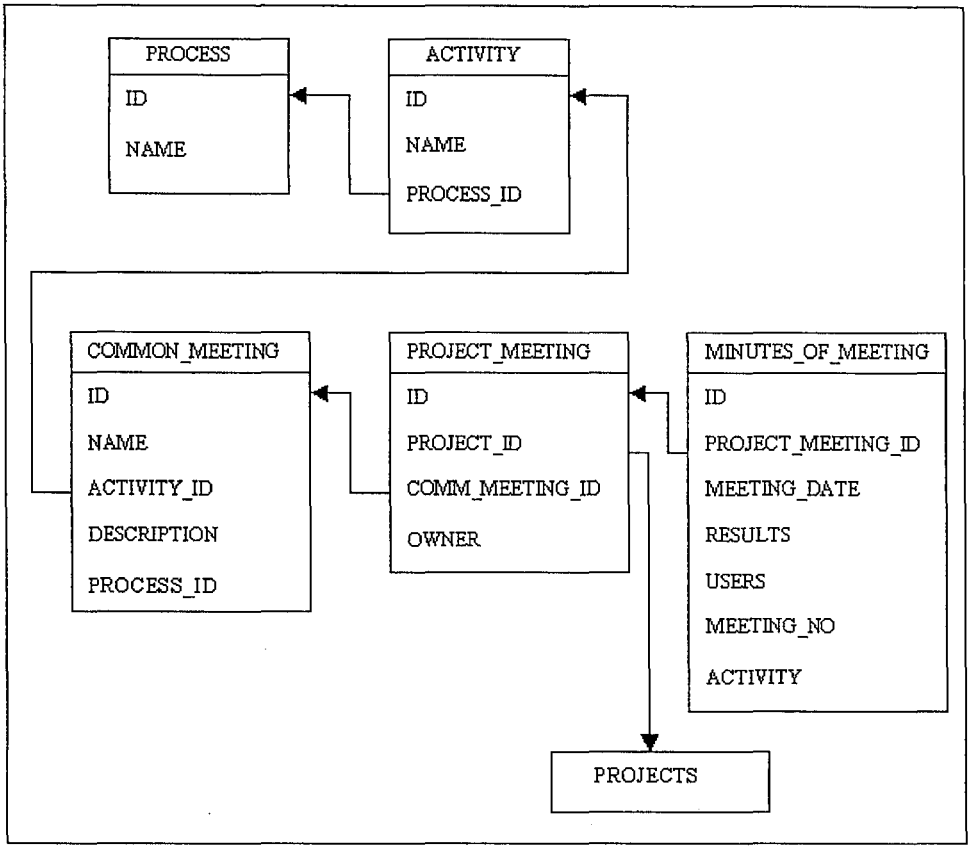


Şekil 5.10 Projeye ait cscı ve csc'ler için oluşturulan tablolar ve ilişkileri

IEEE/EIA 12207'de süreçler ve faaliyetler tanımlanmıştır ve yazılım projelerinin bu faaliyet ve süreçleri gerçekleştirmesi gerekmektedir. Bu bilgiler önceden tanımlı ve değişmeyecek bilgiler olduğu için Şekil 5.11'deki "Process" ve "Activity" tablolarının içeriği de fiziksel gerçekleştirim sırasında doldurulmuş ve projenin ilerleyen kısımlarında veritabanı sorumlusu dışında kimse tarafından değiştirilemeyecektir.

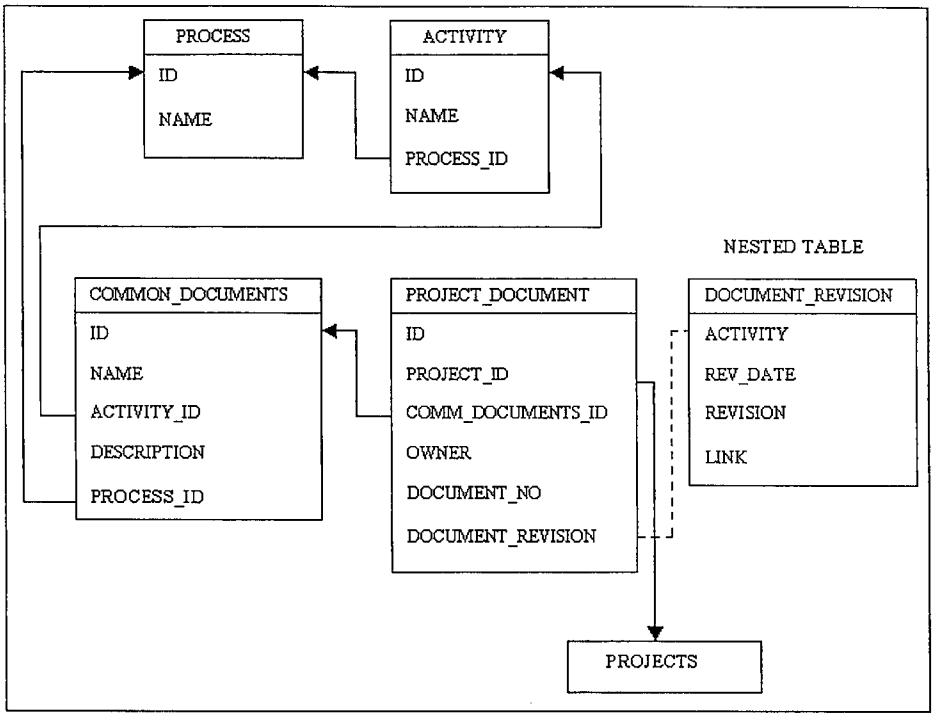
Yine uygulanan yazılım geliştirme standardına göre her projede yapılması gereken toplantılar vardır. Bu toplantılar isimleri, hangi süreç ve aktivite içinde yapıldıkları, kim tarafından yönetildikleri, toplantı tarihi, katılımcıları, toplantı numarası ve toplantı sonucunda alınan kararlar bilgileriyle şekil 5.11'de görüldüğü gibi saklanırlar. Buradaki "Common_Meeting" tablosu bir proje içinde düzenlenebilecek tüm tabloları içerir. Toplantılara ait bilgiler "Project_Meeting" ve "Minutes_Of_Meeting" olarak adlandırılan iki tabloda tutulmaktadırlar. Bu bilgilerin tek bir tabloda değil de iki farklı tabloda saklanmasının nedeni hem verilere erişim sırasında kolaylık sağlanması hem de bir toplantının gerekirse

tekrarlanabilmesinden kaynaklanmaktadır. Toplantıların tekrarlanması durumunda önceki toplantılar ve onlara ait tüm bilgiler de saklanmalıdır.



Şekil 5.11 Süreç, aktivite ve toplantılar için oluşturulan tablolar ve ilişkileri

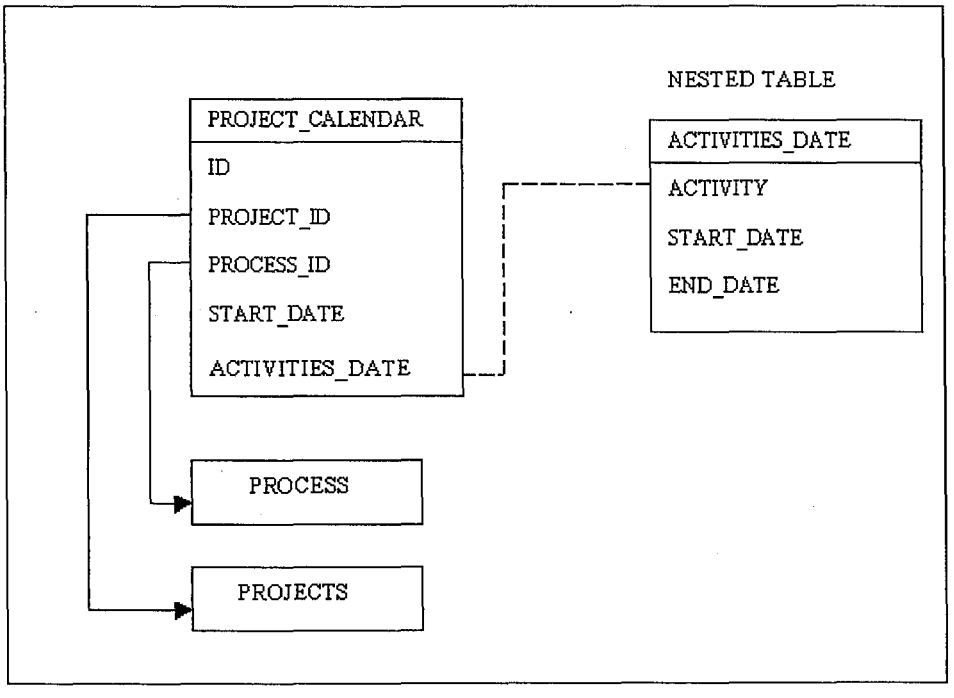
IEEE/EIA 12207 yazılım yaşam döngüsü süreçlerinin en önemli destek süreçlerinden birisi de dokümantasyondur. Yazılım projesi süresince üretilmesi gereken dokümanlar vardır. Her bir dokümanın üretiminden sorumlu olacak bir kişi dokümanın herkes tarafından erişiminin engellenmesine neden olacaktır. Buradan da anlaşılacağı gibi doküman veritabanı için bir varlıktır ve doküman varlığının diğer özellikleri doküman numarası, revizyonu, açıklaması, üretildiği faaliyet ve süreç, üretim yada değişiklik tarihi ve doküman revizyonudur. Şekil 5.12'de görüldüğü gibi doküman ile ilgili tüm bu bilgilerin saklanması için iki tablo ve bir tane nested tablosu hazırlanmıştır. "Common_Documents" tablosu, veritabanı tasarımı sırasında doldurulmuş hazır bir tablodur ve bir yazılım projesinde üretilebilecek tüm dokümanların isimlerini, o dokümana ait açıklamaları, dokümanın hangi süreç ve faaliyet içinde üretilmesi gerektiğini saklar.



Şekil 5.12 Proje dokümanları için oluşturulan tablolar ve ilişkileri

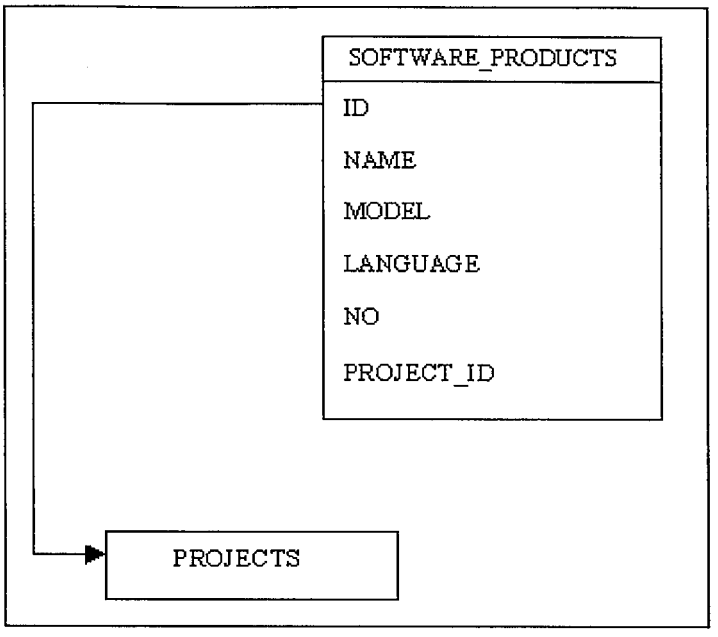
Tüm dokümanların, her bir yazılım projesinde üretilmesi zorunlu değildir. Bir yazılım projesinde bu dokümanların bir kısmı seçilerek üretilebilir. Bu nedenle oluşturulan "Project_Document" tablosu bir projede üretilecek dokümanları ve bu dokümanların sahiplerini saklar. Dokümanların bazı yazılım süreçlerinde güncellenmesi yada dokümana hatalı girilen bilgilerin değiştirilmesi nedeniyle doküman revizyonları oluşabilir. Doküman revizyonları, revizyon tarihleri ve hangi faaliyet içinde bu işlemin gerçekleştirildiği gibi bilgiler "Document_Revision" ismindeki nested tablosu içerisinde saklanmaktadır. Nested tablosunun alanları "Project_Document" tablosunun birer alanı olarak tanımlanmış olsaydı bir dokümanın her bir revizyonu için tekrarlanan satırlar ortaya çıkacaktı. Oracle'ın bu gibi tekrarlamaları engellemek amacı ile kullanıma sunduğu nested tablosu sayesinde herhangi bir anahtar tanımlamaksızın iki tablonun birbiri ile ilişkilendirilmesi mümkündür. Nested tablosu tanımlandığı tablo içerisindeymiş gibi gözükse bile aslında Oracle tarafından ayrı bir hafıza bölgesinde saklanmaktadır.

YPTP programı için tanımlanan ihtiyaçlardan bir diğeri ise her bir projeye ait süreç ve faaliyetlerin başlangıç ve bitiş tarihlerinin takvimlendirilerek saklanabilmesidir. Bu amaçla Şekil 5.13'de görülen tablo ve ilişkiler oluşturulmuştur. Burada takvim, veritabanı için bir varlıktır ve bu varlığın ait olduğu proses, faaliyet ve bunların başlangıç ve bitiş tarihleri de bu varlığın özellikleridir. Bir yazılım projesi, süreçlerden oluşur ve bazı süreçlerin içerisinde de gerçekleştirilmesi gereken aktiviteler bulunmaktadır. "Project_Calendar" tablosunun içerisinde bu nedenle oluşabilecek tekrarlamaları önlemek amacı ile "Activities_Date" adında bir nested tablosu oluşturulmuş ve aktivite ismi, aktivitenin başlangıç ve bitiş tarihi de bu nested tablonun içine yerleştirilmiştir.



Şekil 5.13 Proje takvimi için oluşturulan tablolar ve ilişkileri

Belirlenen ihtiyaçlara göre bir yazılım projesi açılırken oluşturulacak yazılım ürünü belirlenmelidir. Yazılımın tasarımı bu basamakla başlar. Yazılım ürününün ismi ve numarası belirlendikten sonra kullanılacak programlama diline ve yazılım modeline karar verilmelidir. Tüm bu bilgiler Şekil 5.14'de görülen "Software_Products" tablosunun alanlarını oluşturarak her bir projenin yazılım ürünü ve bu ürüne ait bilgiler saklanmış olur.



Şekil 5.14 Yazılım ürünü için oluşturulan tablolar ve ilişkileri

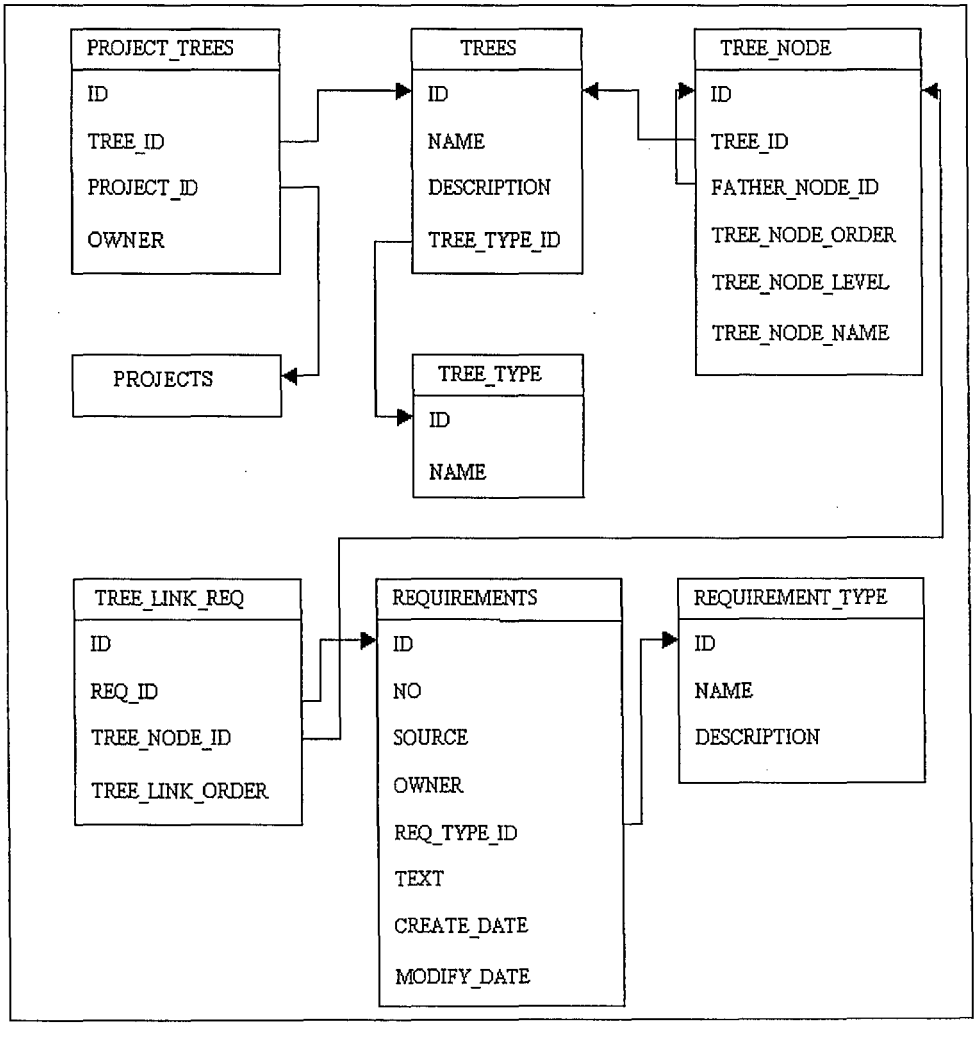
Veritabanının buraya kadar olan bölümünde genel anlamda standardın gerçekleştirilip gerçekleştirilmediğini kontrol eden tablolar ve ilişkileri tasarlanmıştır. Bundan sonraki tasarım ise daha çok YPTP özelleştiren tablolardan oluşmaktadır.

YPTP ihtiyaçlarından en önemlisi her bir yazılım projesine ait sistem ve yazılım ihtiyaçlarının tanımlanabilmesi ve bu ihtiyaçların izlenilebilirliğinin sağlanmasıdır. Yazılım projesine başlandığında ilk adım sistem ihtiyaçlarının tanımlanmasıdır. Sistem ihtiyaçları tanımlandıktan sonra yazılım ihtiyaçlarına geçilir. Bazı yazılım ihtiyaçları sistem ihtiyaçlarından doğar, bazıları ise yazılımın kendi tasarımı sonucunda ortaya çıkar. Eğer bir yazılım ihtiyacı bir sistem ihtiyacından doğuyorsa bu ilişkinin de veritabanında saklanması gerekmektedir.

İster sistem isterse yazılım ihtiyacı olsun tüm ihtiyaçlar bir numaraya, tanımlanma tarihi, içerik ve tanımlayan kişi bilgilerine sahiptir. Bu nedenle veritabanı içerisinde "Requirements" isimli bir tablo tanımlanmış ve yukarıdaki bilgiler de bu tabloya birer alan olarak eklenmiştir. Tanımlanan ihtiyacın ne tür bir ihtiyaç olduğunun saklanabilmesi için ihtiyaç tablosunda bir yabancı anahtar tanımlanmış ve bu yabancı anahtar veritabanı tasarımı sırasında içeriği doldurulmuş "Requirement_Type" adındaki tablo ile ilişkilendirilmiştir.

İhtiyaçların, doküman içeriklerinin ve testlerin izlenebilirliğini arttırmak için veritabanı içerisinde ağaç tanımlaması yapılmıştır. Kullanıcı, YPTP ile projeye ait çeşitli ağaçlar tanımlayabilecek ve bu ağaçların kollarını ekleyebilecek, çıkarabilecek yada değiştirebilecektir.

Şekil 5.15'de görüldüğü gibi her bir ağacın isim, açıklama ve ağaç tipi bilgileri bulunmaktadır. Ağaç tipleri, veritabanının tasarımı sırasında sabit olarak girilmiş bilgilerden oluşan "Trees_Type" tablosunda saklanmaktadır. Yabancı anahtarı ile "Trees" tablosuna bağlanan "Tree_Node" tablosu içerisinde ise bir ağacın kol bilgileri yer almaktadır. Bu tablo içerisinde diğer tablolardan farklı olarak tablo içerisinde tanımlanmış bir yabancı anahtar yine aynı tablonun birincil anahtarına bağlanmıştır. Tablonun kendisi ile kurduğu bu ilişki ağaç kollarının



Şekil 5.15 Projedeki ihtiyaç ağaçları için oluşturulan tablolar ve ilişkileri

birbirine bağlanmasını sağlamaktadır. Her ağaç bir yada daha fazla kola sahiptir, her kol da bir yada daha fazla kola sahip olabilir. Ağaç kolları tanımlanırken hangi ağacın bir kolu olduğu, bu kolun kol seviyesinin ve kol sırasının ne olduğu ve kolun babası olan kolun hangi kol olduğu bilgileri de saklanmalıdır.Tüm bu bilgilerin saklandığı tablo ise "Tree_Node" tablosudur.

YPTP, kullanıcının bir projede istediği kadar ağaç tanımlamasına olanak sağlar. Bu ağaçlar üretilecek dokümanların içeriklerini oluşturan bir ağaç olabileceği gibi, tanımlanan ihtiyaçları ve yapılacak testleri içerecek bir ağaç da olabilir. Bazen farklı projelerde aynı ağaç yada farklı ağaçlar için aynı kollar kullanılmak istenebilir. Bu durumda YPTP, projeler arasında ağaç yada ağacın kollarının kopyalanabilmesi imkanını kullanıcıya sunar. "Project_Trees" tablosu hangi projede hangi ağaçların tanımlı olduğu ve bu ağaçların sahibinin kimler olduğu bilgilerini saklamak için tanımlanmış bir ağaçtır.

Eğer bir ağacın ihtiyaçlar ağacı olarak tanımlanması söz konusu ise tanımlanan ihtiyacın hangi ağaca ve kola ait olduğu bilgilerinin, farklı bir tabloda saklanması gerekmektedir. Bu amaçla tanımlanan tablo ise "Tree_Link_Req" tablosudur. Tablo doğrudan bu bilgileri saklamamasına karşılık "Tree_Node" ve "Requirements" tabloları arasında ilgili ilişkileri kurarak hangi ihtiyacın hangi ağaçta tanımlandığının bulunmasını sağlar. Bu sayede, bir ihtiyaç birden fazla ağaca dahil edilebilir.

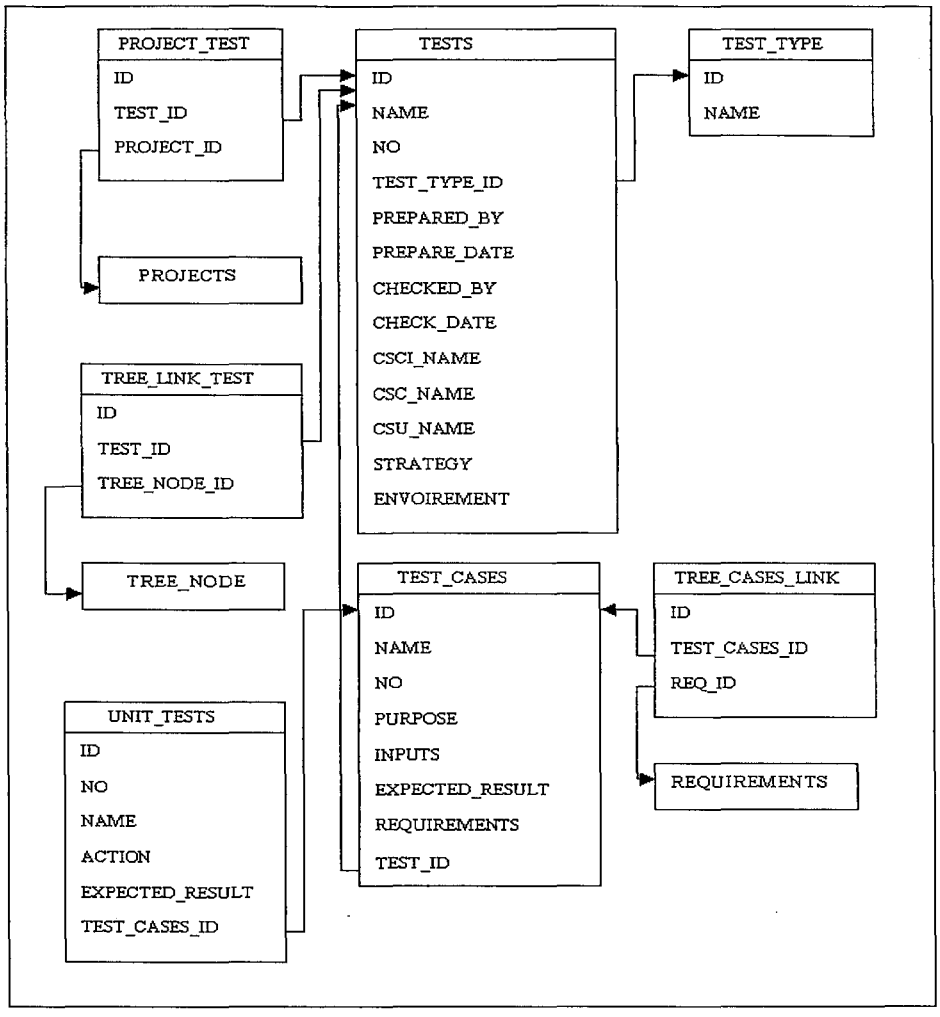
YPTP, ihtiyaçlarından bir diğeri ise yazılım projesinde yapılacak testlerin takibinin sağlanmasıdır. Yazılım projesi sırasında yapılması gereken testler kodlama testleri, yeterlilik testleri ve entegrasyon testleri olarak çeşitlilik gösterir. Türü ne olursa olsun gerçekleştirilmesi gereken ilk aşama testlerin hazırlanmasıdır. Testi hazırlayan kişi testlere birer isim, numara vererek test tipini belirlemelidir. Test ile ilgili takibin yapılabilmesi için de testi hazırlayan kişi, hazırlama tarihi, testin uygulanacağı CSCI, CSC yada CSU ismi, uygulanacak strateji, testin yapılacağı test ortamı gibi bilgilerin de kayıt altına alınması istenmektedir. Testi yapacak olan kişi, teste başladığında böylelikle testin yapılması gerektiği ortamı ve uygulanacak stratejiyi hazırlayabilir. Testlerin detaylandırılabilmesi ve bu detaylı testlerin daha kolay takip edilebilmesi için test basamakları ve birim testleri tanımlanmıştır. Her test bir yada daha fazla test

basamağından oluşur. Test basamaklarının daha fazla detaylandırılmasına ihtiyaç duyulduğu zaman birim testleri tanımlanır.

Her test basamağı; isim, numara, test amacı, test girdileri, beklenen test sonuçları, doğruluğu test edilen ihtiyaç numarası bilgilerine sahip olmalıdır. Bir test basamağının detaylandırılması istendiği taktirde, bu test basamağı içinde birim testleri tanımlanmalı ve her birim testi de isim, numara, yapılması gereken test hareketi ve beklenen test sonucu bilgilerine sahip olmalıdır.

Şekil 5.16'da görülen tablolar ve aralarında kurulan ilişkiler test tanımlamalarının, test takiplerinin yapılabilmesi ve gerektiğinde test formlarının veritabanından birer çıktı olarak alınabilmesi için tanımlanmıştır. "Tests" tablosu isim, numara, test tipi, testi hazırlayanın ismi, hazırlanma tarihi, test stratejisi, test ortamı, testin uygulanacağı CSCI, CSC yada CSU ismi, testi yapan kişinin ismi, testin yapıldığı tarih bilgilerini saklar. "Test_Type" veritabanının tasarımı sırasında içerikleri belirlenmiş bir tablo olarak test tiplerini saklar. Tanımlanan her bir testin hangi projeye ait olduğunu saklayan tablo ise "Project_Test" tablosudur. Veritabanında testlerin detaylandırılmasını sağlayan test basamakları bilgileri "Test_Cases" ve birim testleri bilgileri de "Unit_Tests" tablolarında saklanmaktadır. Tanımlanan ihtiyaçlar ve bu ihtiyaçların karşılanıp karşılanmadığını test eden test basamakları arasında izlenebilirliğin sağlanması için "Test_Cases_Link" isimli tablo tanımlanmıştır. Bu tablo "Test_Cases" ve "Requirements" tabloları arasında uygun ilişkilerin kurulmasını sağlayarak hangi testin hangi ihtiyacı karşıladığının bulunmasını sağlar.

Veritabanı içerisinde testler ve içerikleri birer ağaç olarak tanımlandığı için Şekil 5.16'da görülen "Tree_Link_Test" tablosu tanımlanmıştır. Bu tablo tanımlanan her bir testin hangi ağaca bağlı olduğunu direk olarak saklamasa da tanımlanan yabancı anahtarları sayesinde testin ait olduğu ağaç kolunun bulunmasını sağlar.



Şekil 5.16 Projedeki test ağaçları için oluşturulan tablolar ve ilişkileri

YPTP, ihtiyaçlarından bir diğeri, testler sırasında bulunan hataların saklanması ve düzeltilmesidir. Hataların takibinin yapılması için hata takip formları tutulur. Her hata düzeltildikten sonra tekrar test edilmeli ve hatanın giderildiğinden emin olunduğu zaman açılan bu formlar kapatılmalıdır. Veritabanında hatalar için tanımlanan tablolar, bu formda tanımlanacak tüm bilgileri içermeli ve saklamalıdır.

Açılan her hata formunun bir numarası olmalıdır. Hatayı bulan kişi kendi ismini, hatayı bulduğu tarihi, hatanın tipini, durumunu, katagorisini ve alt katagorisini, hangi sistem üzerinde görüldüğünü belirleyerek forma işler. Biriken hatalar yazılım yada sistem grubu lideri tarafından incelenerek önceliği belirlenir ve bu bilgi de açılan hata formuna kayıt edilir. Grup liderleri hatanın ilgili mühendise aktarılmasını sağlarlar. Aktarılan kişi, aktarılma tarihi, önerilen çözüm

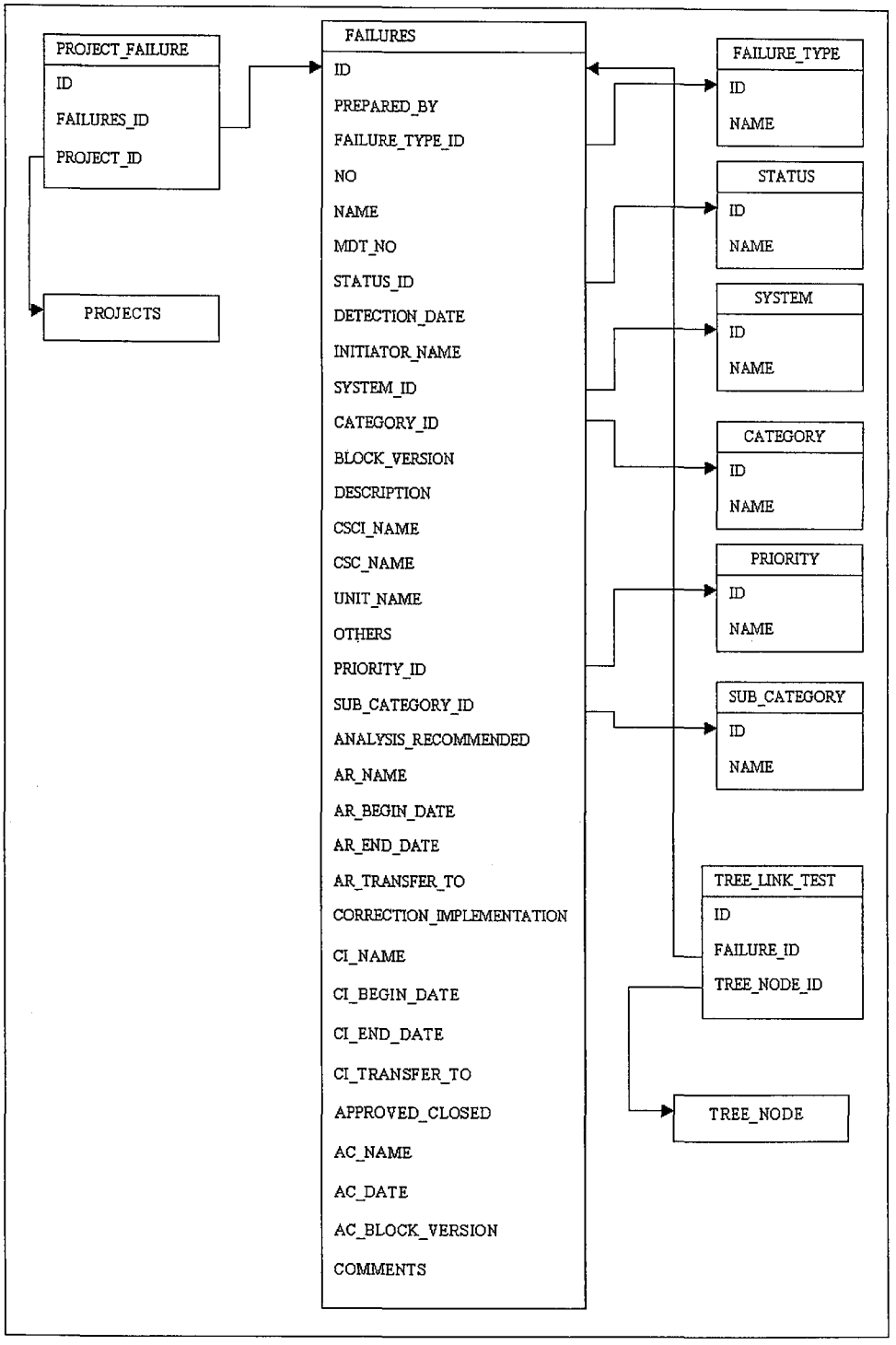
gibi bilgilerin de kayıt edilebilmesi için gerekli alanlar form üzerinde tanımlıdır. Hatayı çözmekle görevlendirilen mühendis yapılan işlemleri, çözülme tarihi gibi bilgileri forma ekleyerek düzeltmeden sonra hatanın tekrar gerçekleşip gerçekleşmediğinin test edilmesi için formun durumunu değiştirir.

Hata ilk tanımlandığında "Açık" durumundadır. Açık durumunda olan hatayı düzelten ilgili mühendis hatanın durumunu "Kontrol edilecek" olarak değiştirir. Yapılan testler sonucunda hatanın giderildiği gözlemlenirse hata "Kapalı", aksi taktirde ise hata durumu tekrar "Açık" olur. Bir hata farklı kişiler tarafından bulunarak iki kere açılmış olabilir. Bunun farkına varılması durumunda açılan hatalardan biri diğerini referans ederek kapatılır ve hata durumu "İptal" olarak belirlenir.

Problem ve modifiksayon olmak üzere iki çeşit hata tip vardır. Eğer mevcut yazılımda önceden tanımlanmış bir ihtiyacın gerçekleşmediği gözlemleniyorsa bu problem, yeni bir ihtiyacın gerçekleştirilmesi isteniyorsa bu modifikasyon olarak tanımlanır. Bir hata; sistemde, yazılımda yada test istasyonu gibi üzerinde çalışılan diğer ortamlarda bulunmuş olabilir. Bunlar hataların rastlanabileceği katogorilerdir. Bir yazılım projesinde hata projenin başından sonuna kadar gerçekleştirilen her aşamada olabilir. Bir hatanın projenin başlarında yapılması bundan sonra süre gelen aşamaların da etkilenmesine ve aynı hatanın gerçekleştirilen aşamalarda devam etmesine neden olabilir. İhtiyaçların doğru ve net tanımlanmamasından kaynaklanan bir ihtiyaç tanımlama hatası, buna bağlı olarak tasarımda, kodlamada ve de dokümantasyonda hatanın devam etmesine neden olur. Açılan bir hata formunda alt kategori olarak tanımlanan bu bilginin de mutlaka doldurulması gerekmektedir.

Kapsamlı ve büyük yazılım projelerinde bulunan hatalar yada sonradan tanımlanan modifikasyonlar birikerek takibi zor bir hal alabilir. Bulunan bu hataların düzeltilme sürecinin uzaması proje takviminin değişmesine yol açabilir. Bu nedenle her hata için bir yapılma önceliği tanımlanmalıdır ve proje takviminin bundan en az şekilde etkilenmesi sağlanmalıdır. Tanımlanan bu öncelikler hataların hangisinin daha evvel ilgili mühendislerce düzeltilmesi gerektiğini belirtir.

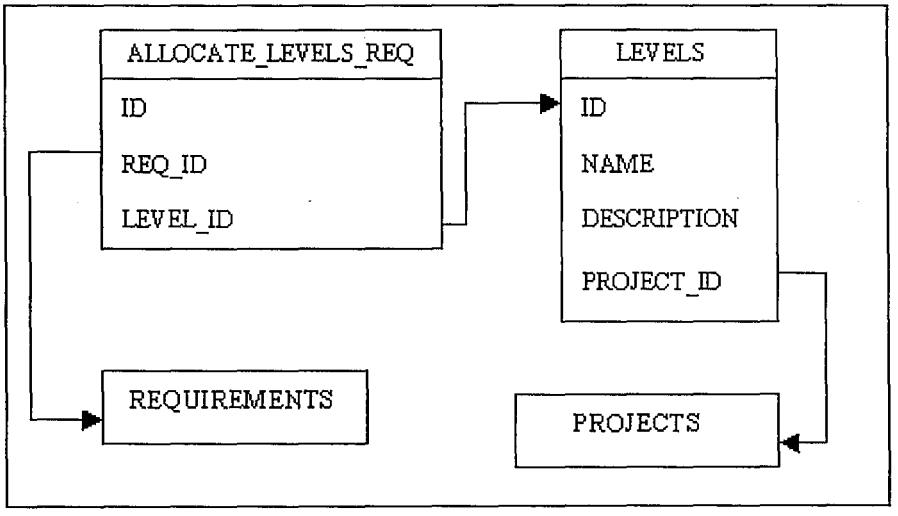
Tanımlanan bu ihtiyaçlar doğrultusunda Şekil 5.17'de görülen tablolar ve ilişkileri oluşturulmuştur. Projede bulunan bir hata ve bu hataya ait numara, isim, hatayı bulan kişi, hatanın bulunma tarihi, hata için önerilen çözüm ve diğer tüm bilgiler "Failures" tablosunun birer alanı olarak tanımlanmıştır. Bu tablo içerdiği



Şekil 5.17 Projedeki hatalar için oluşturulan tablolar ve ilişkileri

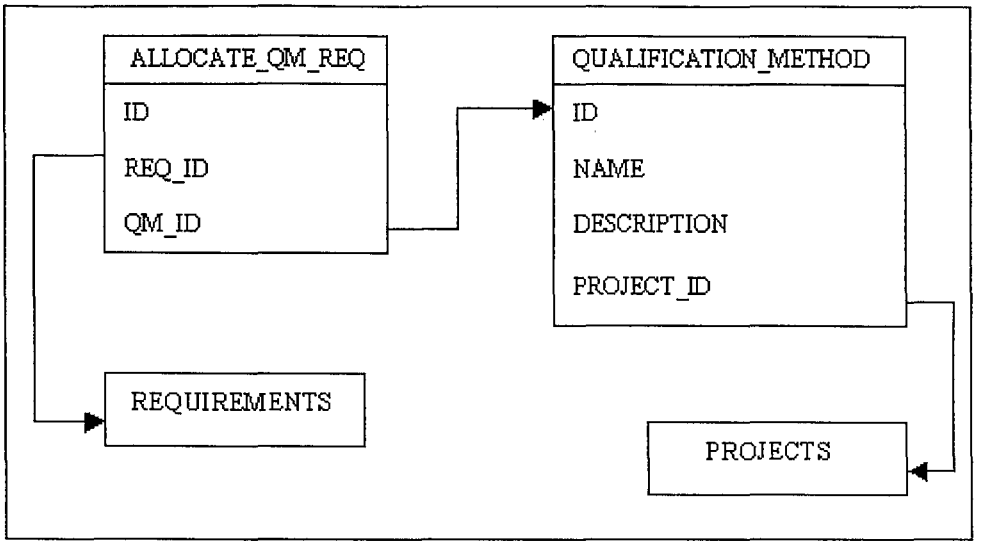
yabancı anahtarları aracılığıyla "Failure_Type", "Status", "Priority", "System", "Category" ve "SubCategory" tabloları ile ilişkilendirilmiştir. Adı geçen tablolar veritabanı tasarımı sırasında tanımlanan ihtiyaçlar doğrultusunda sabit olarak girilen bilgilerden oluşmaktadır. Bulunarak kaydedilen her bir hatanın hangi projeye ait olduğunun bulunması ise "Project_Failures" tablosu aracılığı ile olmaktadır.

YPTP, yazılım projesi kapsamında tanımlanan her bir ihtiyaç seviyesinin, kalite metodunun, tipinin ve üzerinde yapılması planlanan geçerlilik testlerinin kullanıcı tarafından tanımlanarak ilişkilendirilmesi istenmektedir. Kullanıcı, yazılım projesinde ihtiyacın tanımlanabileceği seviyeleri kendisi belirler, daha sonra da her bir ihtiyacın hangi sevide yazıldığını kayıt altına alır. Bu amaçla oluşturulan tablolar ve ilşkileri şekil 5.18'de verilmiştir.



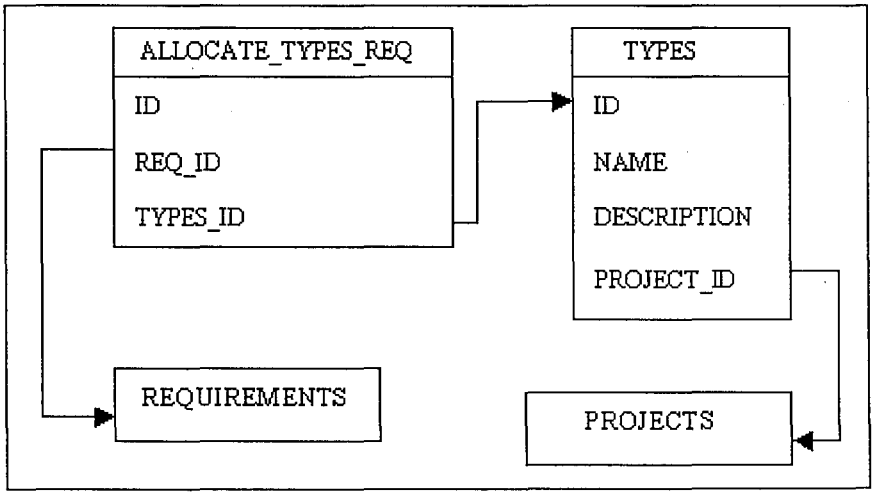
Şekil 5.18 Projedeki ihtiyaç seviyeleri için oluşturulan tablolar ve ilişkileri

Uygulanacak kalite metodları projeden projeye değişim gösterebilir. Kullanıcı bu amaçla her bir proje için kalite metodu tanımlar ve o projeye ait her bir ihtiyacın da hangi kalite metodu ile kontrol edildiğini belirler. YPTP'nin bu ihtiyacı doğrultusunda hazırlanan tablolar ve ilişkileri ise Şekil 5.19'da gösterilmiştir.



Şekil 5.19 Projedeki kalite metodları için oluşturulan tablolar ve ilişkileri

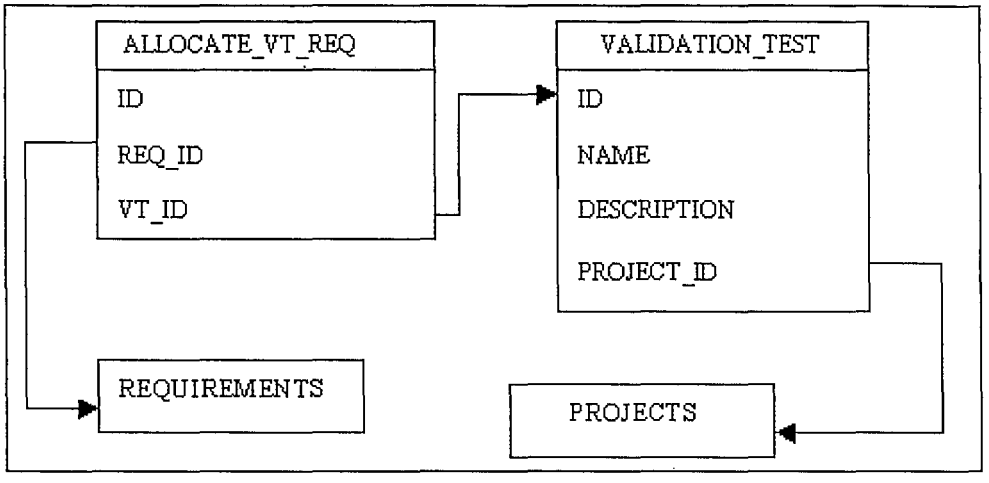
İhtiyaçlar; sistem tasarımı, yazılım gerçekleştirimi, yazılım testi, doküman şablonu yada tasarım modülü için tanımlanmış olabilir. Bu ve bunun gibi örnekler proje çalışanlarına göre çoğaltılıp azaltılabilir. Daha sonrada yazılan her bir ihtiyacın bu tiplerden hangisine referans ettiği belirtilir. Şekil 5.20'de bu yapının kurulması için oluşturulan tablolar ve ilişkileri görülmektedir.



Şekil 5.20 İhtiyaç tipleri için oluşturulan tablolar ve ilişkileri

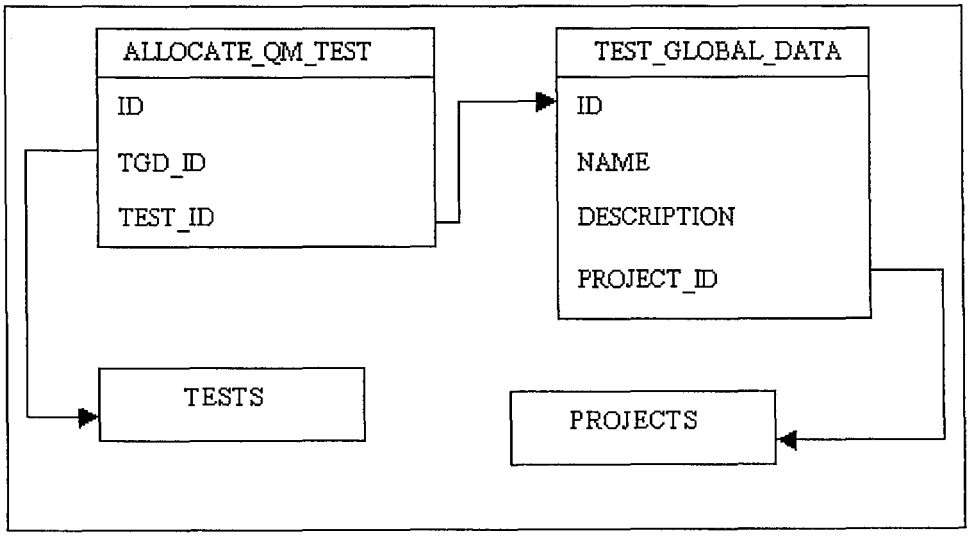
YPTP, gerekliliklerinden bir diğeri ise yazılım projesinde tanımlanacak her bir ihtiyacı test sonuçlarının nasıl değerlendirileceğinin tanımlanmasıdır. Projelere

göre tanımlanacak bu değerlendirme testleri için gerekli tablolar ve ilişkileri Şekil 5.21'de verilmiştir.



Şekil 5.21 İhtiyaç test sonuçlarının değerlendirilmesi için oluşturulan tablolar ve ilişkileri

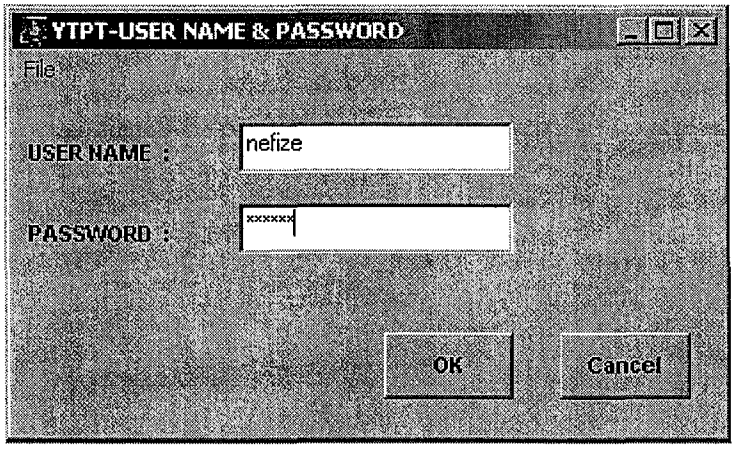
İhtiyaçlara ait genel dataların tanımlanabildiği gibi testler için de genel veri tanımlamalarına ihtiyaç duyulabilir. Şekil 5.22'de görülen tablolar ve aralarında kurulan ilişkiler de test genel verilerinin projelere özel tanımlanması ve tanımlanan her bir testin bu verilerden hangisini referans ettiğinin bulunması için oluşturulmuştur.



Şekil 5.22 Genel veri tanımlamaları için oluşturulan tablolar ve ilişkileri

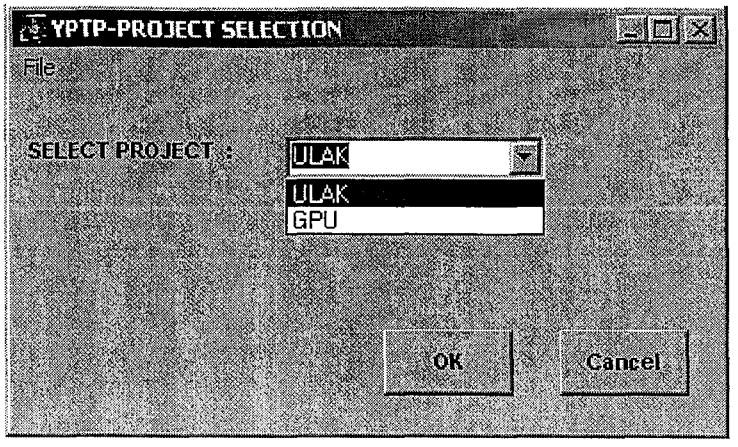
5.4. YPTP'nin Çalışması

YPTP, çalıştırılması ile Şekil 5.23'de görülen giriş sayfası ekranda görülür. Girilecek kullanıcı adı ve şifresi Oracle veritabanında tanımlı olan "User_Password" tablosuna girilmiş olmalıdır. Girilen kullanıcı adı ve şifresinden birinin hatalı olması durumunda program girilen bilgilerin hatalı olduğunu belirten bir uyarı mesajı verdikten sonra girilen bilgileri temizleyerek tekrar girilmesine imkan verir. "OK" butonu girilen bilgilerle işleme devam edilmesini, "Cancel" butonu ise programdan çıkılmasını sağlar.



Şekil 5.23 Giriş sayfası

Şekil 5.23'de doğru kullanıcı adı ve şifresi girilip ok butonuna basıldıktan sonra ekrana Şekil 5.24'de görülen proje seçim penceresi gelir. Bir önceki pencerede girilen kullanıcı adı ve şifresine sahip kişinin çalıştığı projeler bir liste halinde bu pencerede görülür. Bu basamakta hangi proje seçilirse bundan sonraki yapılan tüm işlemler o projeye ait olacaktır. Programın ilerleyen basamaklarında seçilen projenin değiştirilmesinin istenmesi durumunda bu pencereye geri dönülmesi gerekecektir.

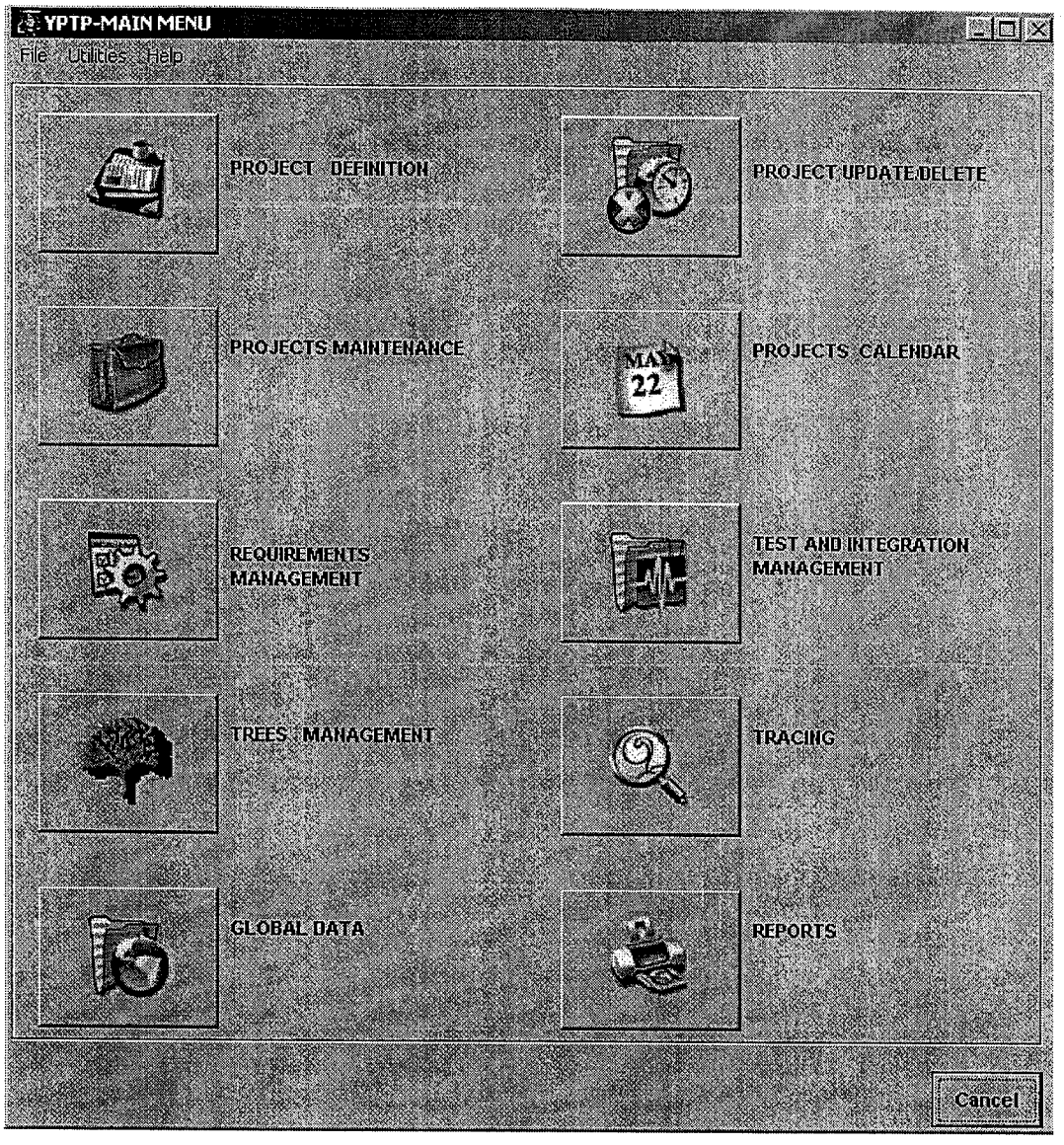


Şekil 5.24 Proje seçim sayfası

Üzerinde çalışılmak istenen proje ismi seçilip "OK" butonuna basıldıktan sonra şekil 5.25'de görülen ana menü penceresi ekrana gelecektir. Şekilde de görüldüğü gibi projenin üzerinde gerçekleştirilebilecek işlemler gruplandırılmıştır. Buna göre ana menüden seçilecek işlem basamağına göre; proje tanımlaması yapılabilir, tanımlanan tüm projeler, tanımlamaları ile birlikte görüntülenebilir, bilgileri güncellenebilir yada proje silinebilir, projenin takibi yapılabilir, takvimi hazırlanıp takibi yapılabilir. Projede tanımlanan tüm ihtiyaçların, testlerin ve bu amaçlarla kullanılan ağaçların yönetimi yapılabilir, projeye özel global veriler tanımlanabilir, ihtiyaçlar ve testler arasında izlenilebilirlik sağlanabilir ve YPTP aracılığı ile yapılan çoğu işlem sonucu rapor şeklinde alınabilir.

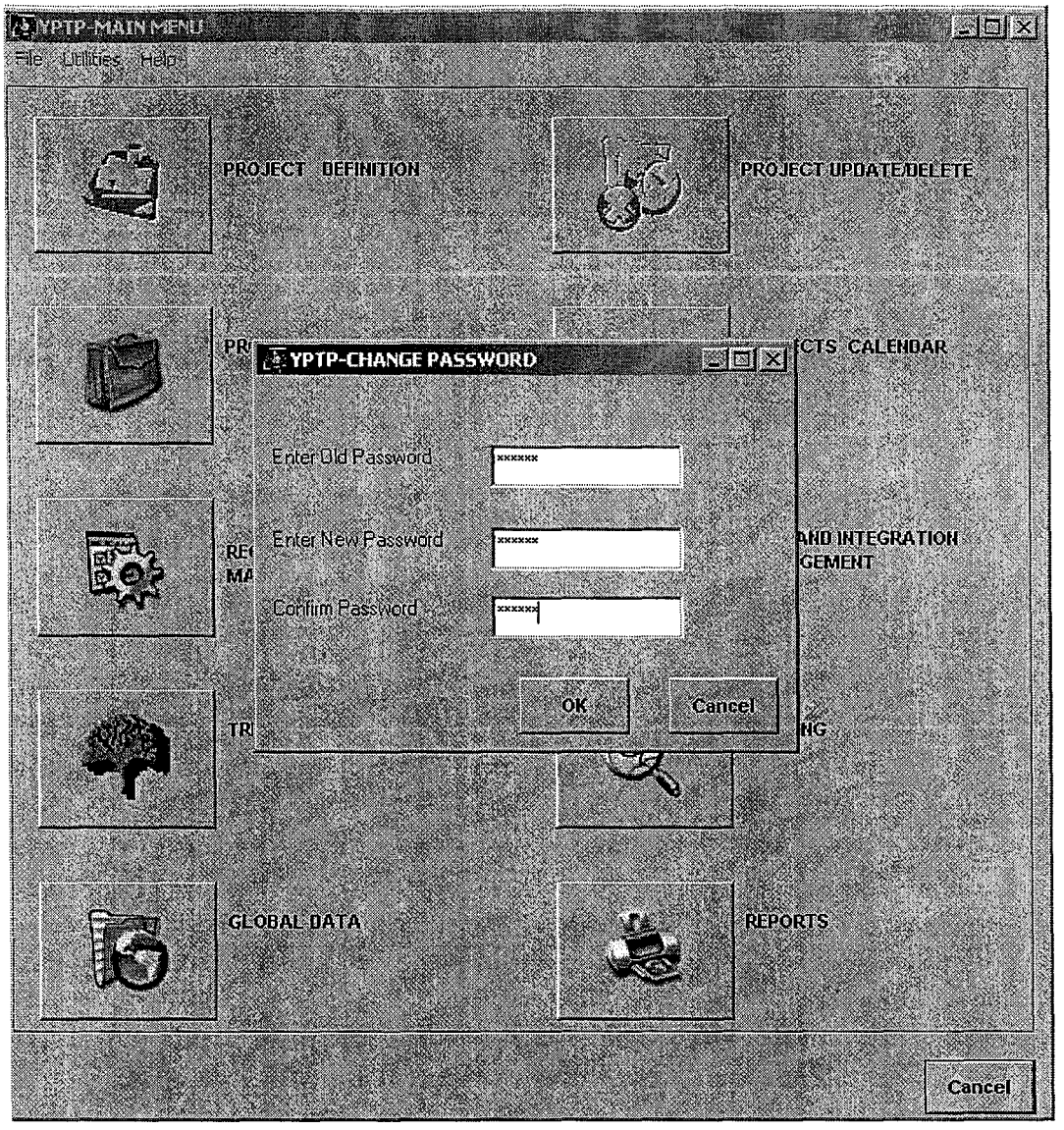
Her kullanıcının ana menüde bulunan tüm butonları kullanarak diğer menü ve pencerelere geçmesi mümkün değildir. Erişim Oracle veritabanında hazırlanan yetkilendirme tabloları ile sınırlandırılmıştır. Bu yetkilendirme tabloları proje teknik yöneticileri, yazılım yada sistem grup liderleri tarafından yapılabilir. YPTP programına giriş sırasında kimliği ve yetkileri veritabanında kaydedilen tablolara göre belirlenen kişinin tanımlanan yetki sınırları içinde ana menüden diğer menülere geçmesi sağlanabilir yada erişim engellenir.

Ana menüdeki "Cancel" butonuna basıldığı zaman şekil 5.24'deki proje seçim penceresine geri dönülür.



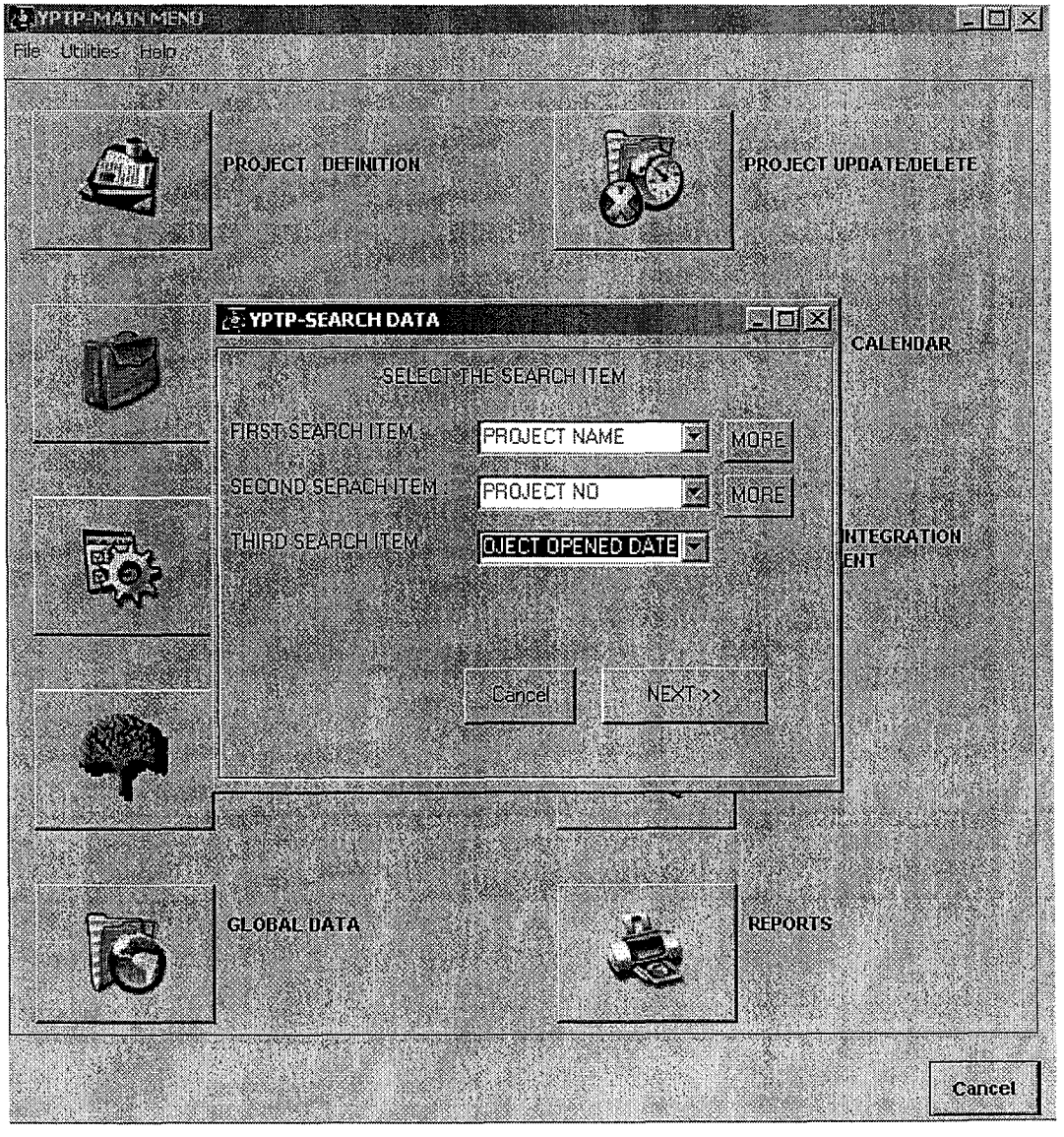
Şekil 5.25 Ana menü

Ana menünün "Utilities" kısmında kullanıcıya yardımcı olması için iki ayrı seçenek konulmuştur. Bunlardan ilki kullanıcı şifresi değiştirilmesi için tasarlanmıştır. "Change Password"e basıldığında ekrana gelecek şekil 5.26'da görülen pencere gelecektir. Burada kullanıcıdan eski kullanıcı şifresini ve yeni kullanıcı şifresini girmesi istenecektir.

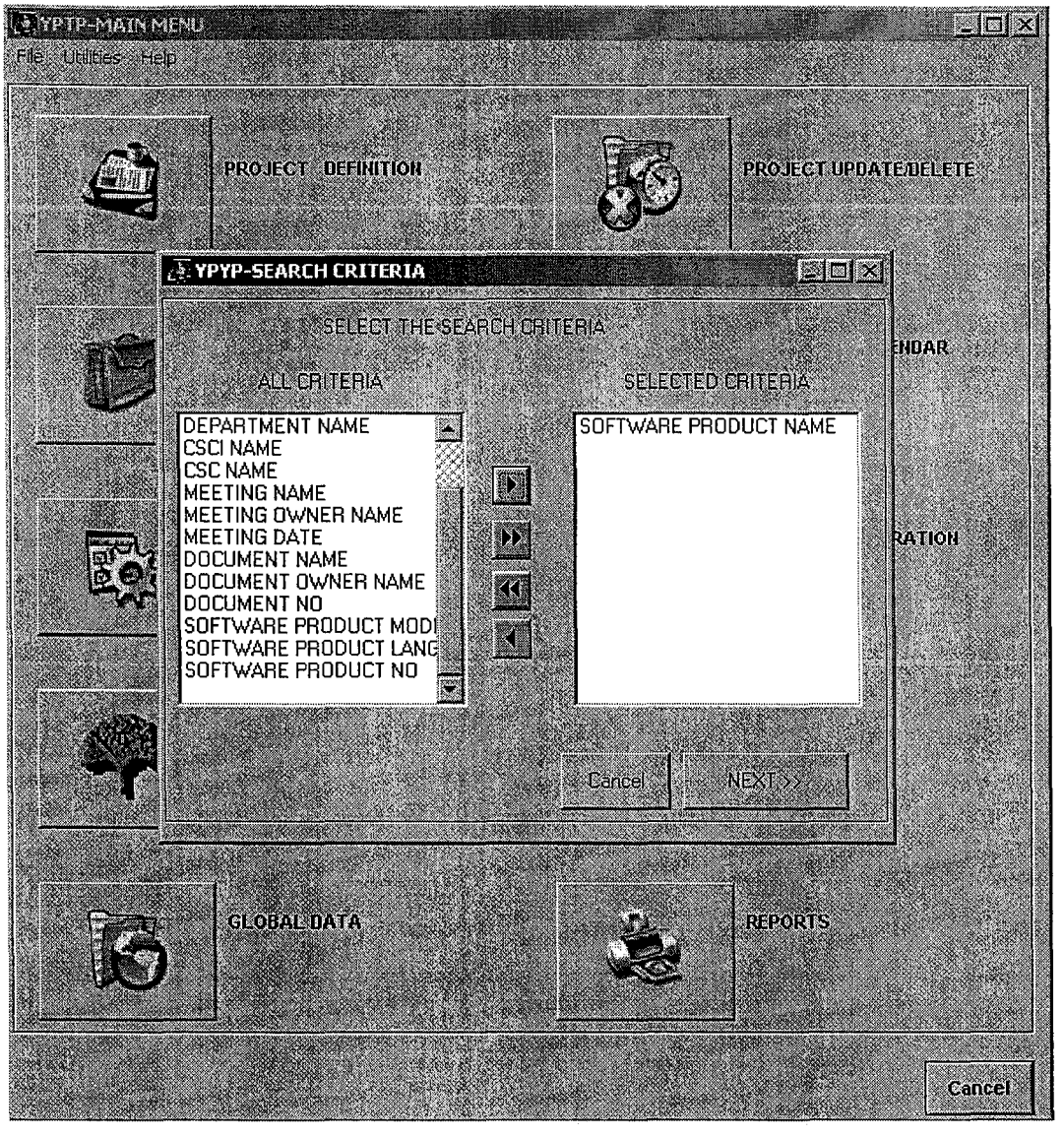


Şekil 5.26 Kullanıcı şifresi değiştirme sayfası

"Utilities" menüsündeki bir diğer seçenek olan "Search Data" seçildiğinde ekrana Şekil 5.27'deki pencere gelecektir. Açılan bu pencere üzerinde veritabanında arama yapılabilecek üç tane veri seçilebilmektedir. Aranması istenen ilk parametre seçildikten sonra "More" butonuna basılırsa program ikinci bir parametrenin aranmasına, ikici satırdaki "More" butonuna basılırsa üçüncü bir parametrenin aranmasına imkan vermektedir. Sayfa üzerinde "Cancel" butonuna basılması arama işlemi sona erdirilmiş olur ve ana menüye geri dönülür. "Next" butonuna basıldığında ise arama işlemi ile ilgili gerekli işlemlerin yapılabilmesi için şekil 5.28'de görülen diğer sayfa ekrana gelir.



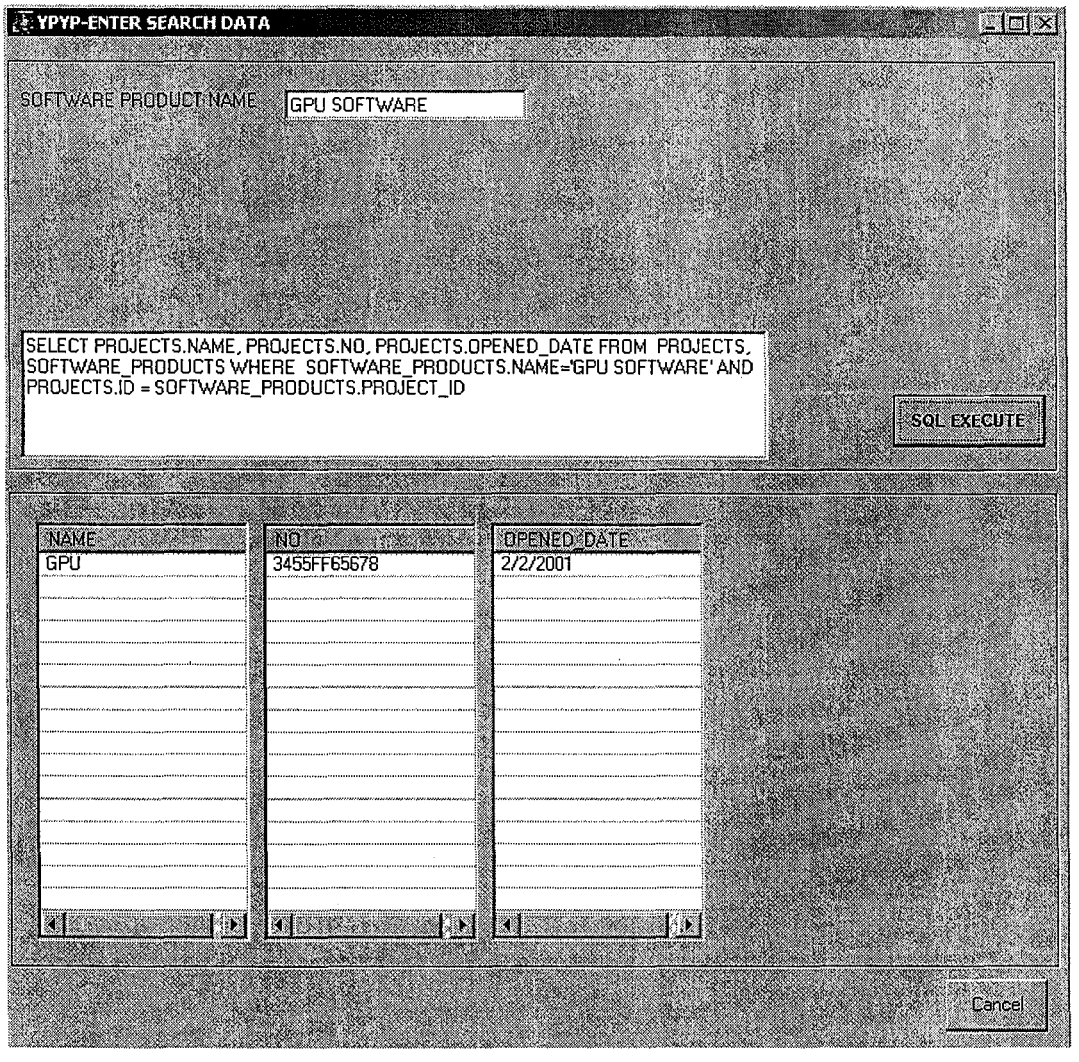
Şekil 5.27 Veri arama sayfası



Şekil 5.28 Arama sayfası

Arama işleminin bu sayfasında açılan yeni ekranın sol tarafındaki listede arama sırasında kullanıcının girebileceği parametre listesi yer almaktadır. Oklar yardımı ile kullanıcı istediği parametreleri seçerek ekranın sağ tarafındaki listeye atabilir. Arama işleminin bir sonraki basamağına "Next" butonuna basılarak ulaşılır.

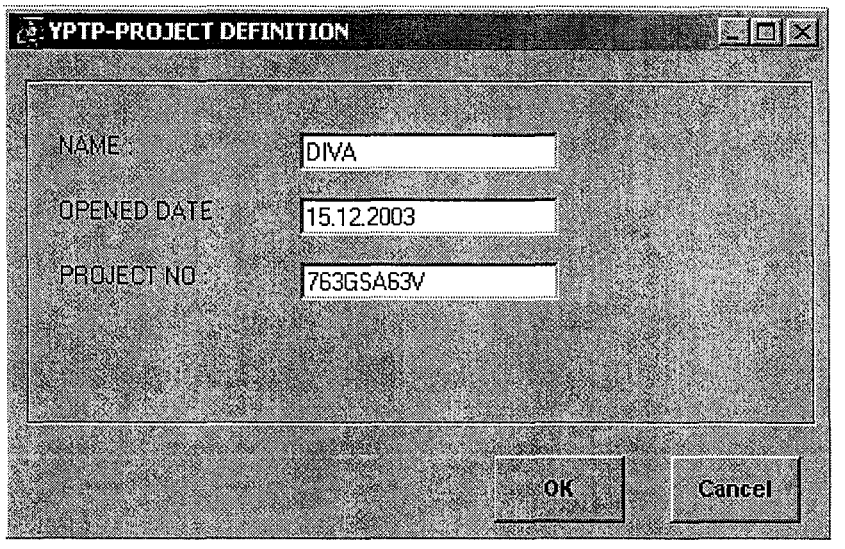
Şekil 5.29'da görülen sayfa, arama işleminin son sayfasıdır. Burada kullanıcının bir önceki basamakta seçtiği parametre, kullanıcı tarafından bir giriş parametresi olarak girilir. "SQL Execute" butonuna basıldığında arama için yapılan tüm tanımlamalar doğrultusunda oluşan SQL ifadesi ekranda görülür ve arama sonucu da ekranın alt kısmındaki tablolarla listelenir



Şekil 5.29 Arama sonuç sayfası

Eğer kullanıcının yetkileri dahilindeyse ana menüdeki "Project Definition" butonuna basıldığında şekil 5.30'daki proje tanımlama sayfası ekrana gelir. Proje tanımlama sadece avyonik şube müdürlerine verilen bir yetkidir. Bu pencerede projenin adı, projenin açılış tarihi ve projeye verilecek tanıtım numarası tanımlanır. Bilgi girişinden sonra "Ok" butonuna basılırsa; proje, veritabanında tanımlanmış olur. "Cancel" butonuna basılması durumunda ise proje

tanımlamaları ile ilgili herhangi bir bilgi girilmiş olsa dahi proje tanımı yapılmadan bu pencereden çıkılarak ana menüye dönülür.

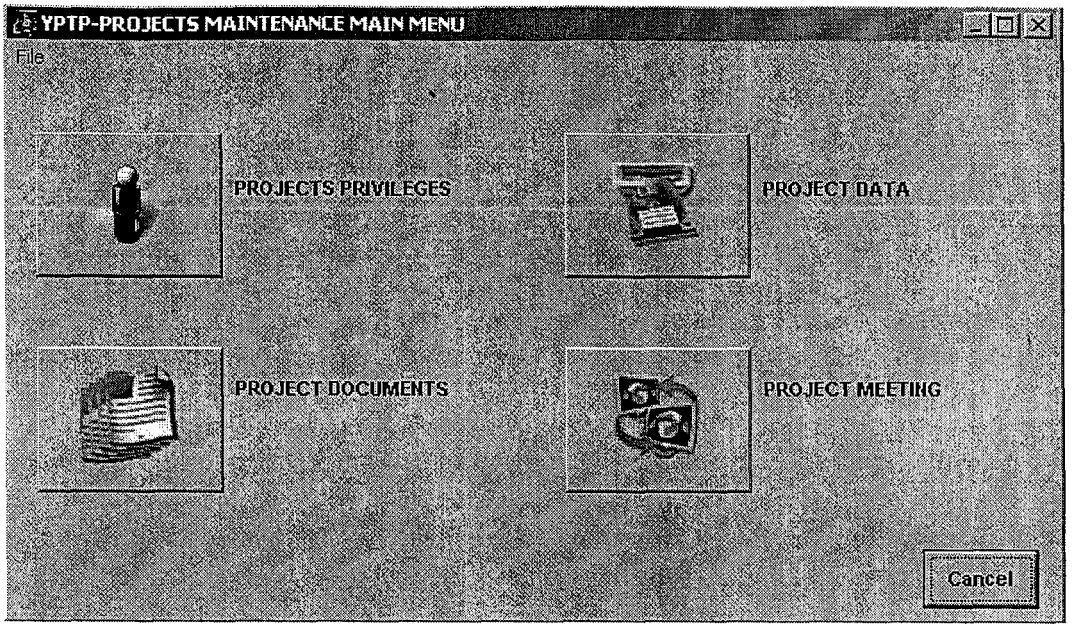


Şekil 5.30 Poje tanımlama sayfası

Ana menü sayfasında proje güncelleme ve silme butonuna basıldığı taktirde Şekil 5.31'de görülen pencere görülür.

Daha önceden proje tanımlama penceresinde tanımlanan tüm projelere bu pencereden erişilerek isim, açılış tarihi ve numara bilgileri görülebilir. Veritabanında kayıtlı projelere ekranda görülen oklar sayesinde teker teker ulaşılabilir. Projeye ait bilgilerden herhangi birisinin güncellenmesi durumunda veritabanına yeni bilginin kayıt edilebilmesi için "Update" butonuna basılmalıdır. "Delete" butonuna basılması ise ekranda görünen projenin silinmek istemesi anlamına gelir. Bu butona basılması durumunda ekranda projenin silinmek istendiğinden emin olunup olunmadığının sorgulanması için bir uyarı mesajı verilir.

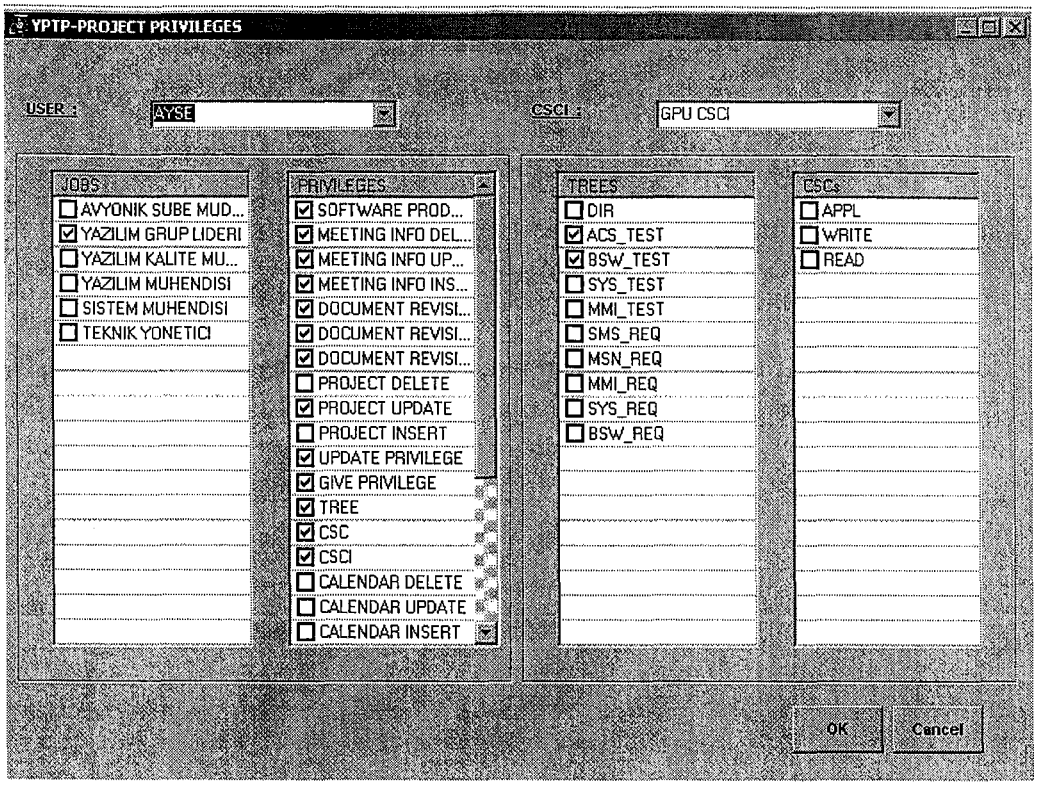
Proje tanımlamaları yapıldıktan sonra bu projelerin bakımlarının gerçekleştirilmesi gerekmektedir. Ana menüdeki "Project Maintenance" butonuna basıldığında ekrana Şekil 5.31'de görülen ve "Project Maintenance Main Menu" olarak adlandırılan başka bir alt menü gelir.



Şekil 5.31 Proje idame ana sayfası

Proje bakım ana sayfası, ekranda iken programdan çıkmak için "File" menüsünden "Exit" seçilmelidir. Proje bakım ana menüsü üzerinde yetkilendirme yapmak, proje verilerini, dokümanlarını ve toplantılarını tanımlamak için butonlar yer almaktadır. Bu pencerede "Cancel" butonuna basılması ana menüye dönülmesini sağlar.

Proje bakım ana menüsü üzerindeki "Project Privilege" butonuna basıldığında Şekil 5.32'deki proje yetkileri penceresi ekrana gelir. Ekranın sol üst tarafında, içinde bulunulan projenin tüm çalışanlarının listesi bulunur. Hangi elemanın yetkilerinde güncelleme yapılacaksa onun ismi listeden seçilir.



Şekil 5.32 Proje yetkilendirme sayfası

Proje yetkilendirme penceresinde seçilen kişinin proje içindeki görevi ise "Jobs" bölümünden seçilebilir. Tanımlı her bir görevin mutlaka sahip olması gereken yetkiler vardır. Bu nedenle kullanıcının atandığı görev tanımlandığı zaman "Privileges" bölümünde bulunan bazı yetkiler otomatik olarak işaretlenir. Veritabanında önceden tanımlanmış bu yetkiler yeterli değil ve seçilen çalışana daha fazla yetki verilmek isteniyorsa bunlar da yetki kolonunda elle işaretlenmelidir. Kullanılan yazılım standardına göre her çalışana her yetkinin verilebilmesi de mümkün değildir. Bir proje görevlisinin asla yapamayacağı işler de bulunmaktadır. Veritabanında tanımlanmış bulunan bu yetkiler seçilmek istendiğinde ise bunun mümkün olmayacağını belirten bir uyarı mesajı verilir.

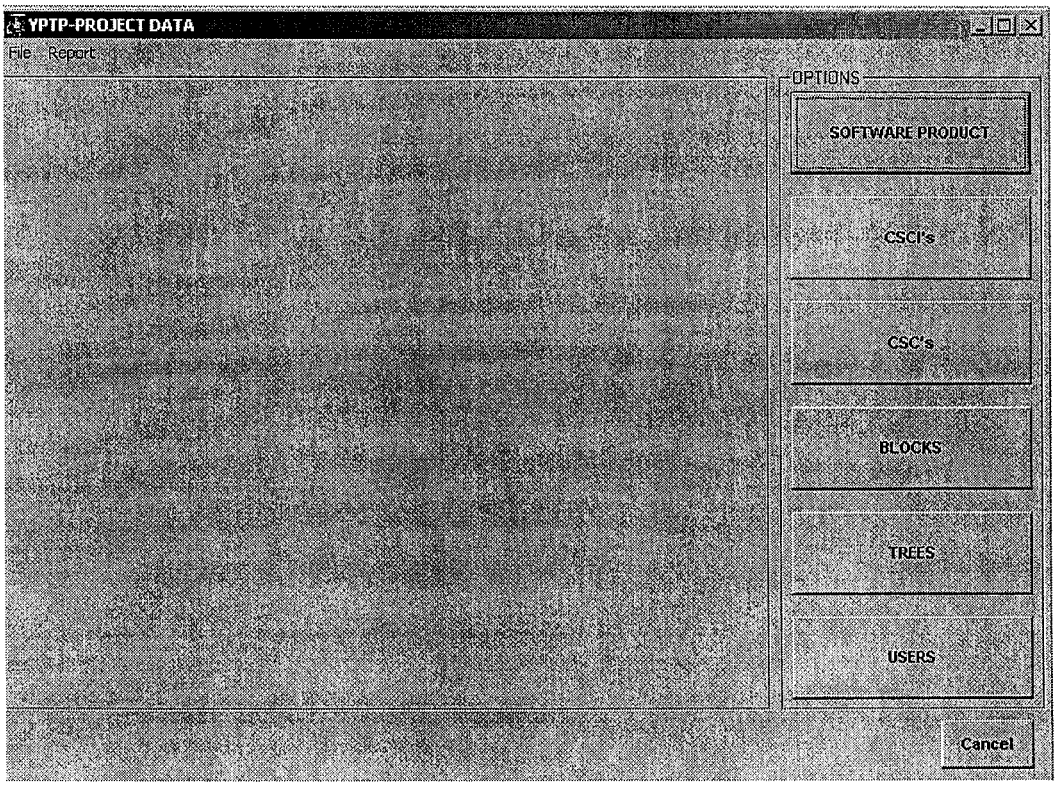
Her yazılım projesi en az bir tane yazılım birimine sahiptir. Her yazılım biriminde ise bir yada daha fazla yazılım bileşeni bulunmaktadır. Proje yetkilendirme sayfasının sağ üst tarafında projeye ait yazılım birimi listesinden bir tanesi seçildiği zaman, seçilen yazılım birimine ait yazılım bileşenleri "CSCs" bölümü altında ekrana gelir. Eğer proje çalışanı bir yazılım mühendisi ise bu

CSC'lerden bir yada bir kaçı seçilerek, yazılım mühendisinin üzerinde çalışacağı yazılım üniteleri kümesi belirlenmiş olur.

"Trees" bölümünün altında ise projeye ait tanımlanmış tüm ağaçlar listelenir. Bu ağaçlardan bir yada bir kaçının seçilmesi proje elemanının üzerinde çalışabileceği ağaçların tanımlanmasını sağlar.

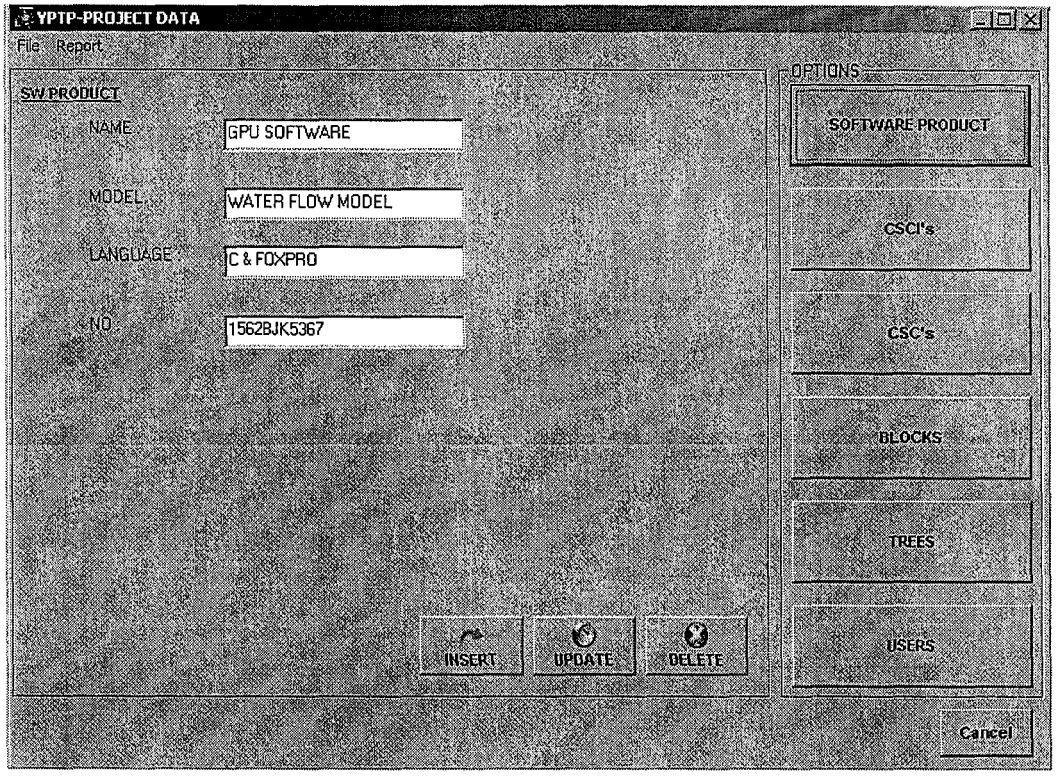
Proje yetkilendirme penceresinde yapılan değişikliklerin veritabanına kaydedilebilmesi için "Ok" butonuna basılması gerekmektedir. Pencere üzerindeki "Cancel" butonuna basılması halinde ise proje bakım ana menüsüne geri dönülür.

Şekil 5.33'de görülen proje bakım ana menüsü üzerindeki "Project Data" butonuna basıldığı zaman, şekil 5.34'deki proje verilerinin tanımlandığı pencere ekrana gelecektir. Bu pencere üzerinde bir kaç seçenek halinde yazılım projesi kapsamında tanımlanabilecek veriler yer almaktadır.



Şekil 5.33 Proje verileri

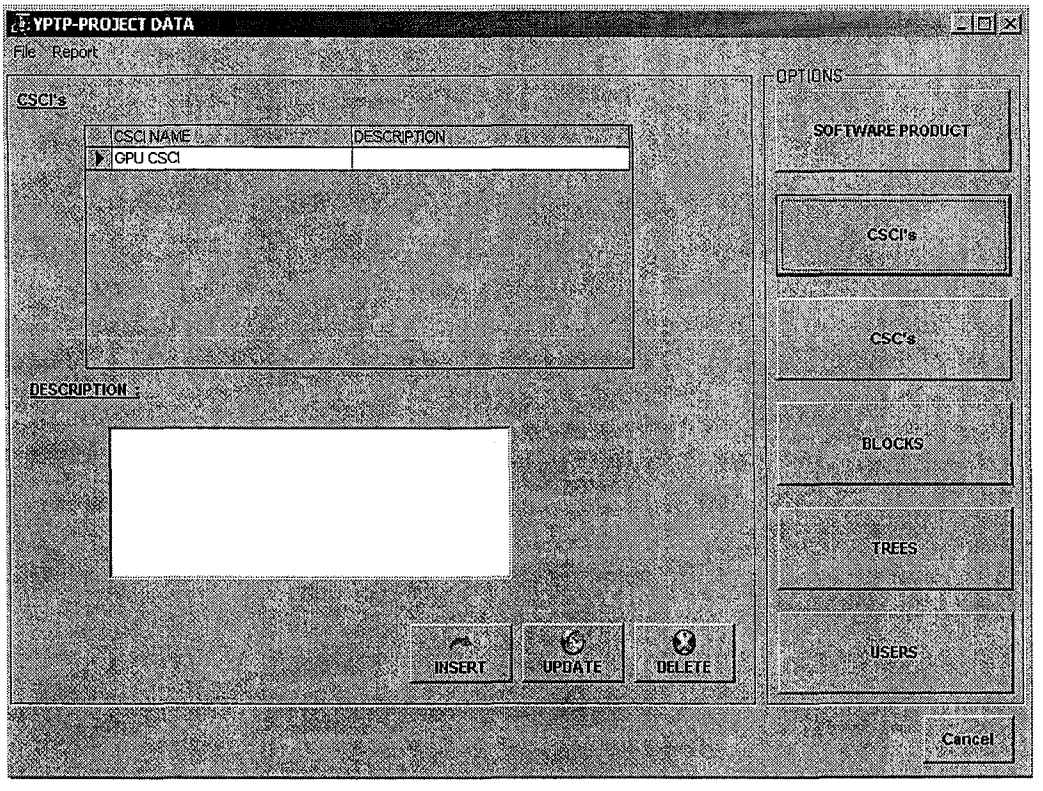
Proje üzerinde tanımlanacak verilerden ilki yazılım ürünüdür. Projeye ait yazılım ürün adı, tanıtım numarası, yazılımı kodlarken kullanılacak dil ve yazılım modeli bu bölümde tanımlanır.



Şekil 5.34 Proje yazılım ürünü sayfası

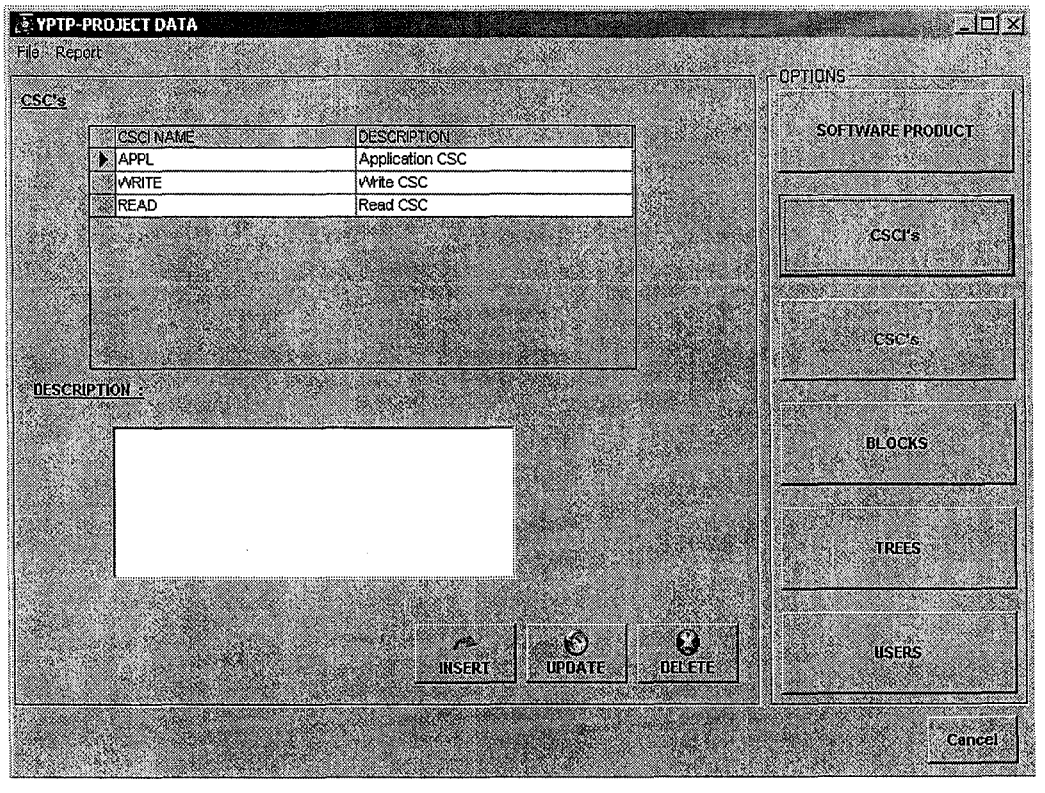
Program kullanıcısının yetkisi olması durumunda alt tarafta bulunan "Insert", "Update" ve "Delete" butonları aktif olacaktır. Yazılım ürünü bilgilerinden herhangi biri değiştirildikten sonra bu değişikliğin veritabanı tarafından da algılanması isteniyorsa "Update" butonuna, yeni bir yazılım ürünü tanımlanacaksa "Insert" butonuna ve mevcut yazılım ürününü veritabanından kaldırılmak isteniyorsa "Delete" butonuna basılır.

"Project Data" penceresi üzerinde CSCI's butonuna basılırsa ekranda yazılım ürünü ile ilgili bilgiler kalkar, bunun yerine yazılım birimi adı ve açıklamasının yazılı olduğu Şekil 5.35'deki grid görülür. Bu pencerede kullanıcının yetkileri doğrultusunda yeni yazılım birimi tanımlama, mevcut yazılım biriminin verilerini güncelleme yada yazılım birimini silme işlemleri gerçekleştirilebilir.



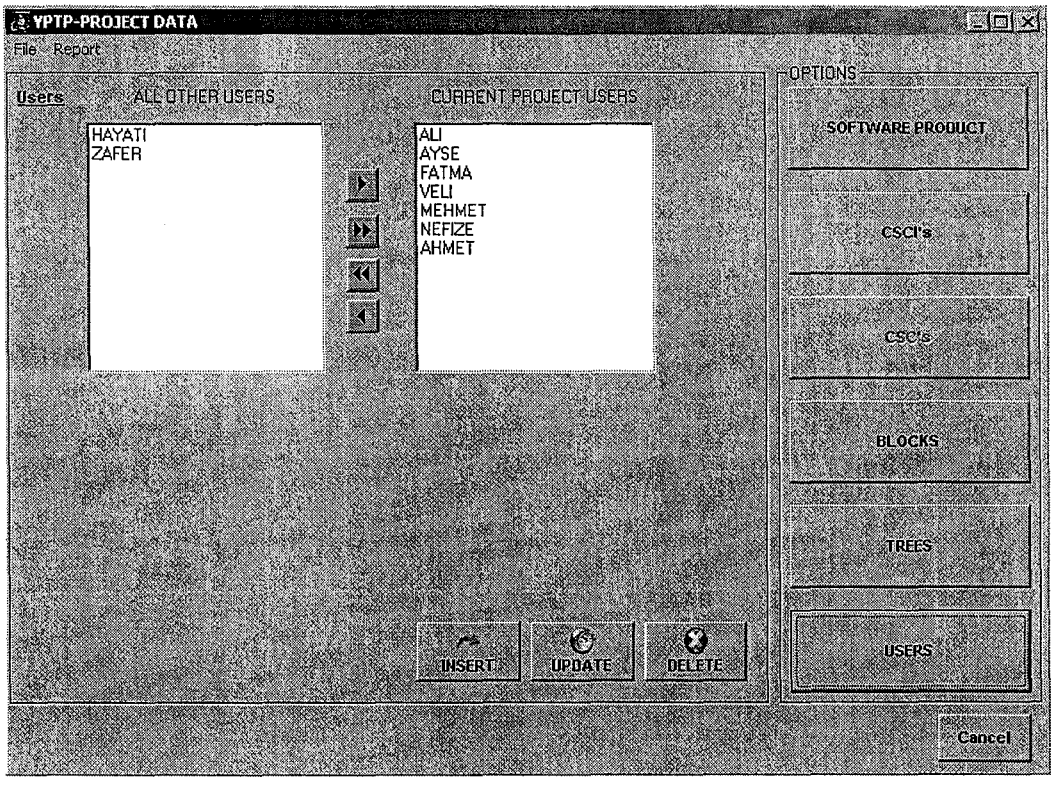
Şekil 5.35 Proje birimleri sayfası

İçinde çalışılmakta olan projenin her bir yazılım birimine ait yazılım bileşenlerinin listelenmesi, yeni bileşenlerin girilmesi, bilgilerin güncellenmesi yada mevcut yazılım bileşenlerinden birinin silinmesi için "Project Data" penceresindeki "CSC's" butonuna basılır. Ekrana Şekil 5.36'daki grid gelir. Bahsedilen işlemlerin yapılabilmesi için kullanıcının bu işlemler üzerinde yetkili olması gerekmektedir.



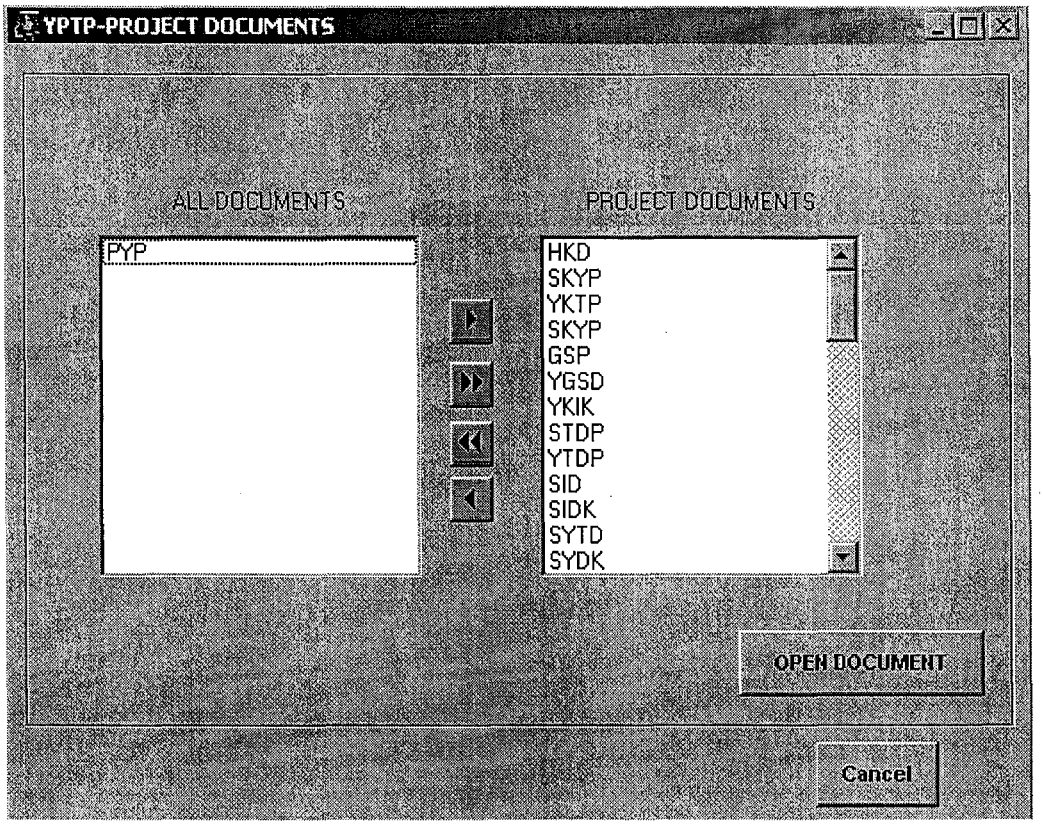
Şekil 5.36 Proje yazılım bileşenleri sayfası

Proje verilerinden bir diğeri ise proje çalışanlarıdır. "Users" butonuna basıldığında Şekil 5.37'deki görüntü ekrana gelir. Sol taraftaki kutunun içinde kurumda içinde çalışan tüm personel listelenmektedir. Pencere üzerinde yer alan oklar yardımıyla bu personelden bazıları sol taraftaki kutuya taşınarak üzerinde projenin çalışanı olarak veritabanına kaydedilmiş olur. Proje teknik yöneticisi gibi personel tanımlama yetkisine sahip olan kullanıcı "Insert", "Update" ve "Delete" butonlarını kullanarak veritabanını güncelleyebilir.



Şekil 5.37 Proje kullanıcıları sayfası

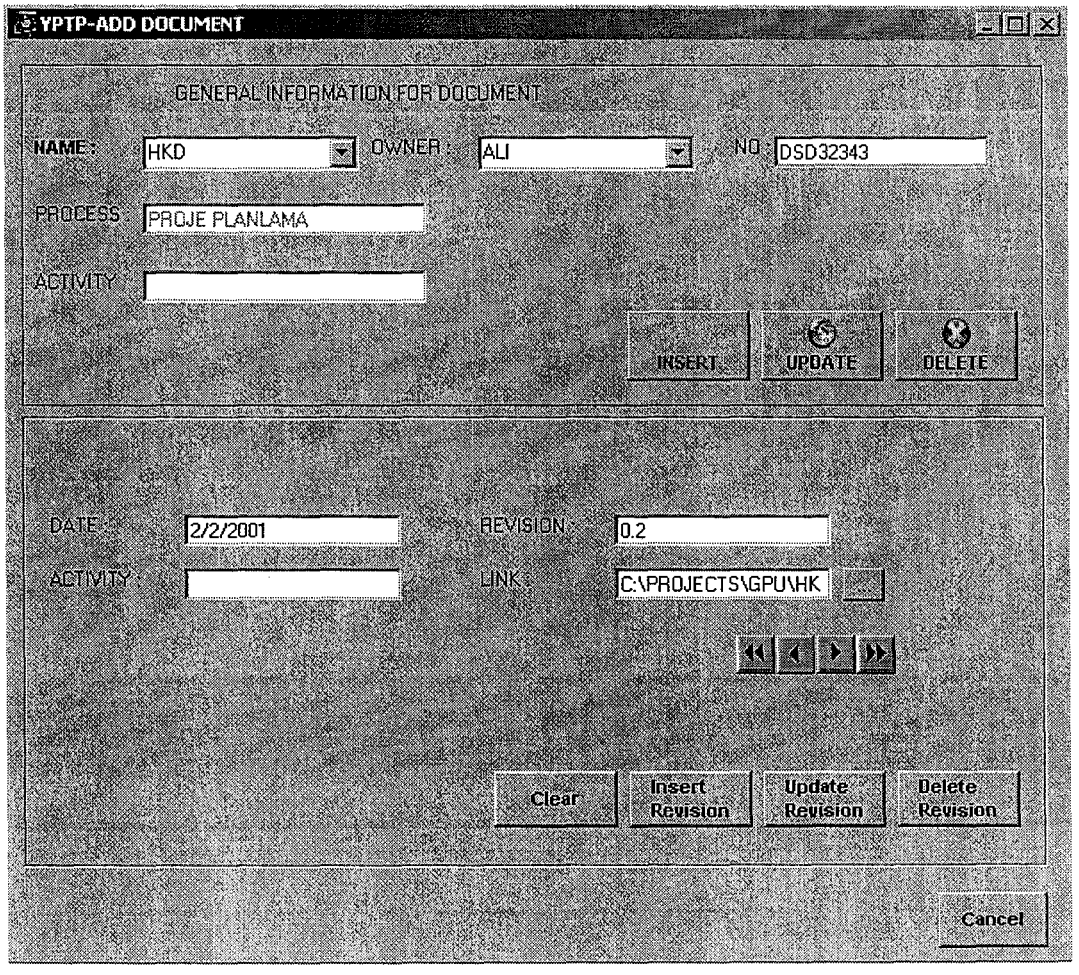
Proje bakım ana menüsü üzerindeki "Project Documents" butonuna basıldığında Şekil 5.38'deki proje dokümantasyon penceresi ekrana gelir. Pencerenin sol tarafındaki kutuda yazılım kalite standardına göre üretilebilecek tüm dokümanların listesi yer almaktadır. Üzerinde çalışılan proje için üretilmesine karar verilen dokümanlar oklar yardımı ile sol kutudan sağ tarafta bulunan kutuya taşınır. Projede üretilecek dokümanların belirlenme işlemini ancak proje içinde buna yetkili kişiler gerçekleştirebilir.



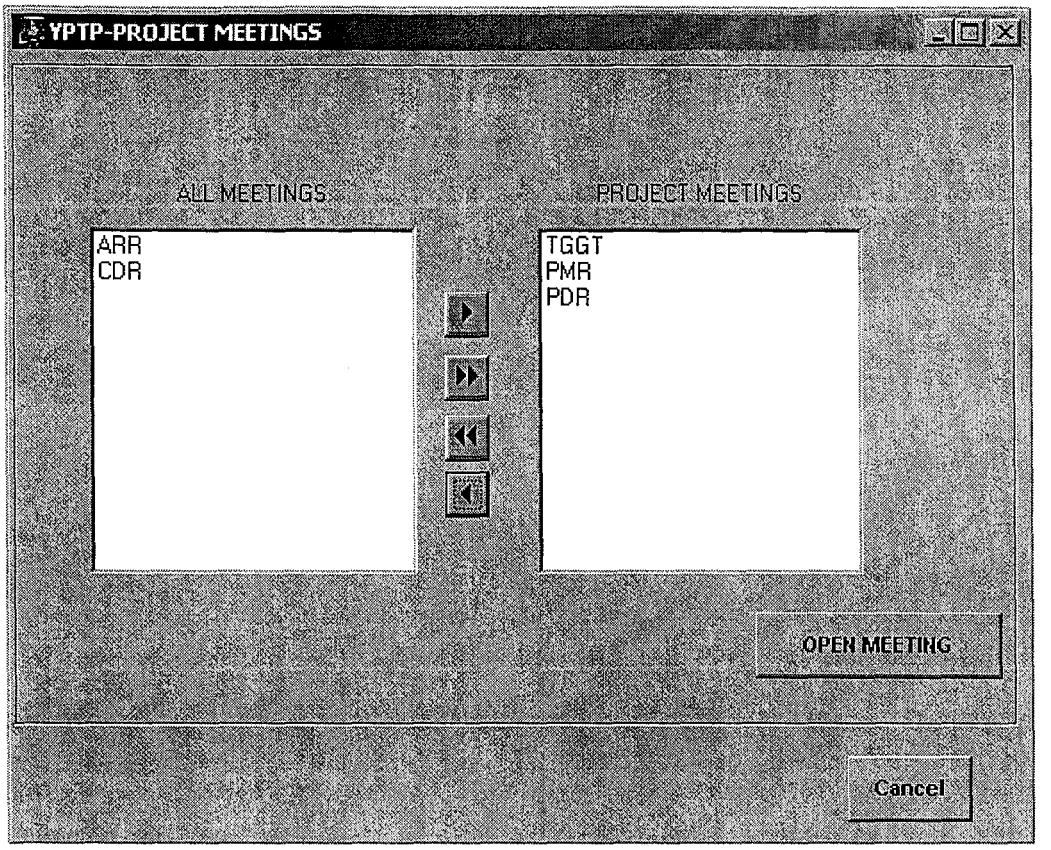
Şekil 5.38 Proje dokümanları sayfası

Projede üretilecek dokümanlardan istenilen biri hakkında daha detaylı bilgi girişi yapılabilir. Bu amaçla proje dokümanları listesindeki dokümanlardan birinin üzerine gelinip fare ile tıklanır ve "Open Document" butonuna basılır. Projeye eklenen doküman hakkındaki bilgileri içeren Şekil 5.39'daki pencere ekrana gelir. Daha önce doküman hakkında bilgi girişinde bulunulmamışsa dokümanı projeye ekleyen kişi; dokümanı üretecek ve dokümandan bundan sonra sorumlu olacak kişiyi tanımlar.

Tanımlanan bu kişi dokümanı üretecek, versiyonlarını çıkartacak, düzeltmeleri yapmaya yetkili kişi olacak ve açılan doküman ekleme penceresindeki ikinci bölümü değiştirebilme yetkisine sahip olacaktır. Dokümanlar değişikliğe uğradıkça yeni revizyon ile tekrar çıkar. Çıkan tüm revizyonların takibini yapabilmek için Şekil 5.39'da görülen oklar kullanılır. Veritabanında saklanmış tüm doküman revizyonlarının üretim tarihi, revizyon numarası ve dokümanın saklandığı dizin bilgileri takip edilebilir. Doküman ekleme penceresi üzerindeki "Cancel" butonuna basıldığında Şekil 5.38'de görülen proje dokümanları sayfasına dönülür.



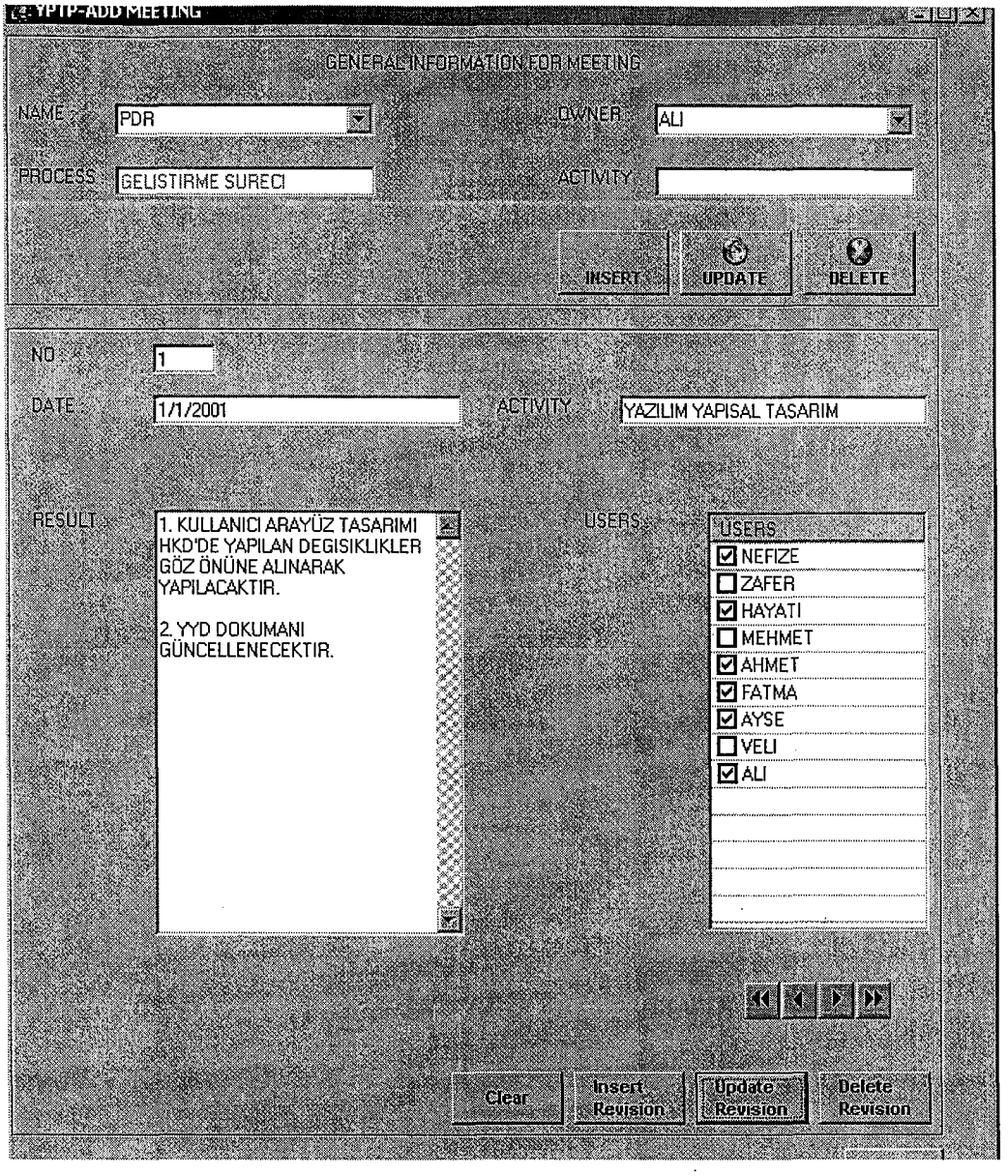
Şekil 5.39 Doküman ekleme sayfası



Şekil 5.40 Proje toplantıları sayfası

Proje bakım ana menüsü üzerindeki "Project Meetings" butonuna basıldığında Şekil 5.40'da görülen ve projede yapılacak toplantıları belirlemeye yarayan pencere ekrana gelecektir. Yapılabilecek tüm toplantıların listelendiği sol taraftaki kutudan oklar sayesinde proje toplantılarının listelendiği sağ taraftaki kutuya alınan toplantılar "Open Meeting" butonuna basılarak açılacak yeni sayfada detaylandırılabilir. Projede toplantı tanımlamak her kullanıcının yetkileri içinde değildir. Genelde proje yöneticisi seviyesindeki kişiler bu işlemi gerçekleştirebilirler.

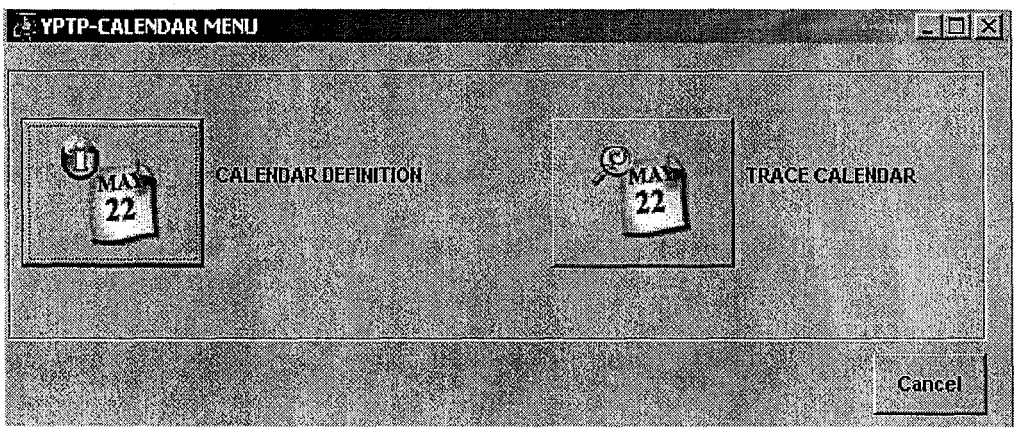
Şekil 5.41'de görülen ve toplantı bilgilerinin yazıldığı pencere iki bölümden oluşmaktadır. Toplantı hakkındaki genel bilgiler olarak isimlendirilen ilk bölüm bu toplantının proje içerisinde yapılmasına karar veren kullanıcı tarafından yapılır. Bu bölümde yer alan toplantı ismi ve toplantının yapılacağı süreç ekrana otomatik olarak gelecektir. Kullanıcı bu aşamada proje içindeki bu toplantıyı kimin düzenleyip, organize edeceğine karar verir.



Şekil 5.41 Toplantı ekleme sayfası

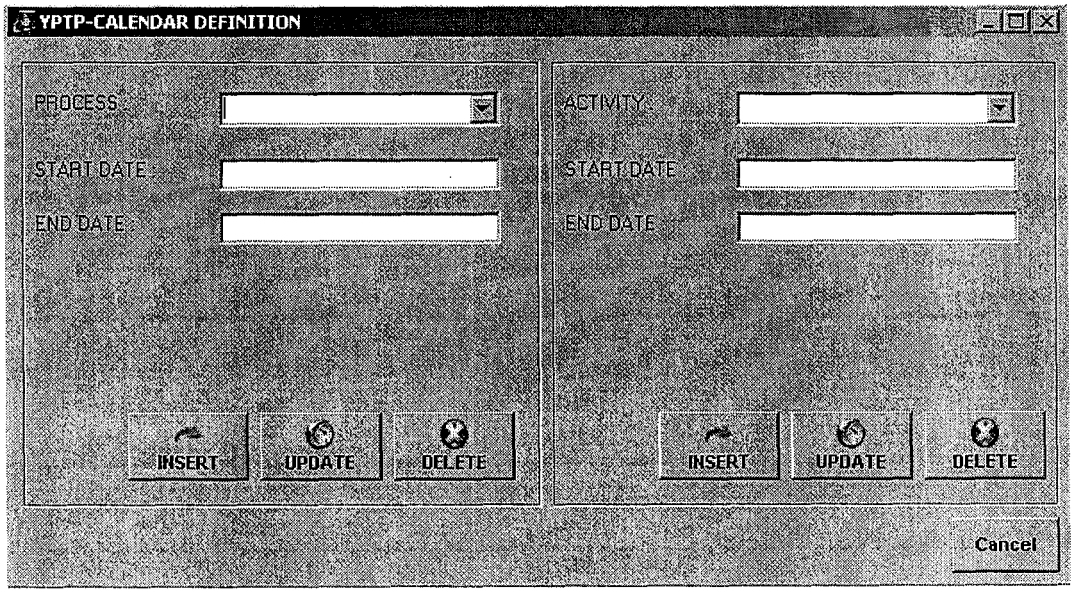
Toplantı bilgilerinden ikincisi ise toplantıyı düzenleyen kişi tarafından doldurulur. Yapılan toplantı sonucunda bir aksaklık çıkmışsa belirlenen aksaklıklar giderildikten sonra toplantı tekrarlanır. Bu nedenle her toplantının tekrarlanma sayısını gösteren numarası toplantıyı düzenleyen kişi tarafından doldurulur. Toplantının düzenlenme tarihi, gerçekleştirildiği aktivite ve toplantı sonucunda alınan kararlar yine toplantıyı düzenleyen kişinin pencere üzerinde yazması gereken bilgilerdir. Kullanıcılar başlığı altında projede çalışan tüm

personelin isim listesi yer almaktadır. Toplantıya katılan personel bu liste üzerinden işaretlenir.



Şekil 5.42 Takvim menüsü

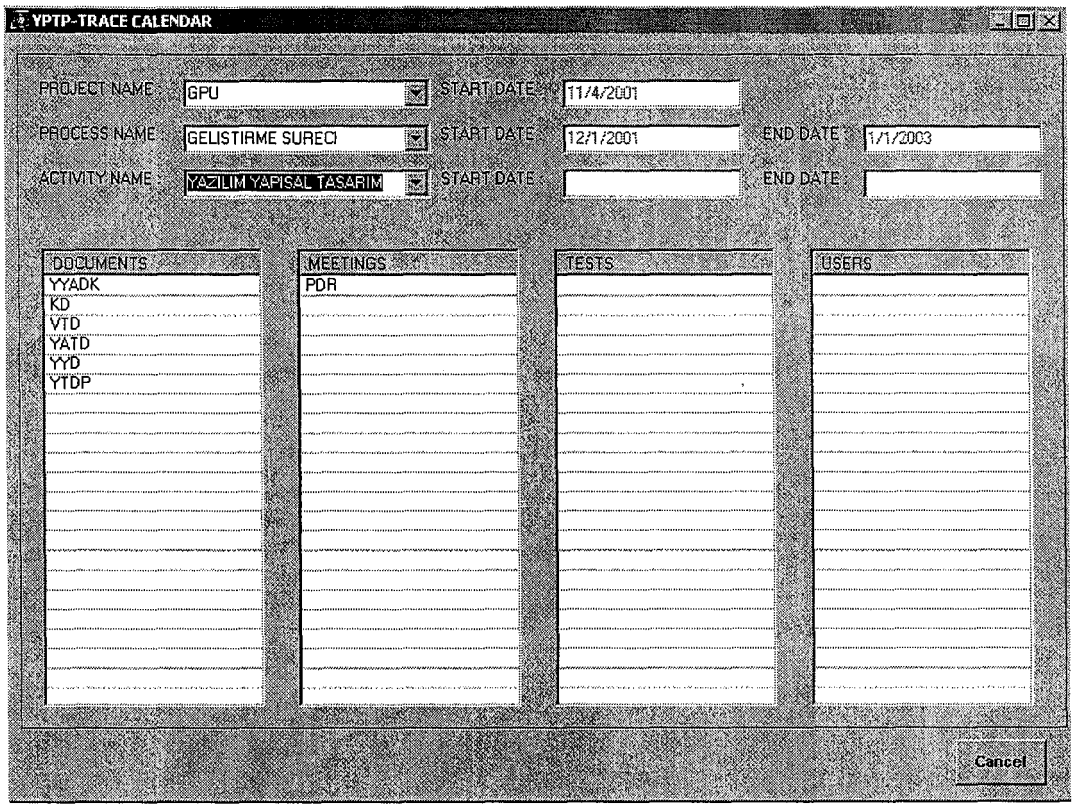
Ana menü üzerindeki "Projects Calendar" butonuna basıldığında Şekil 5.42'deki "Calendar Menu" ekrana gelir.



Şekil 5.43 Proje takvimi tanımlama sayfası

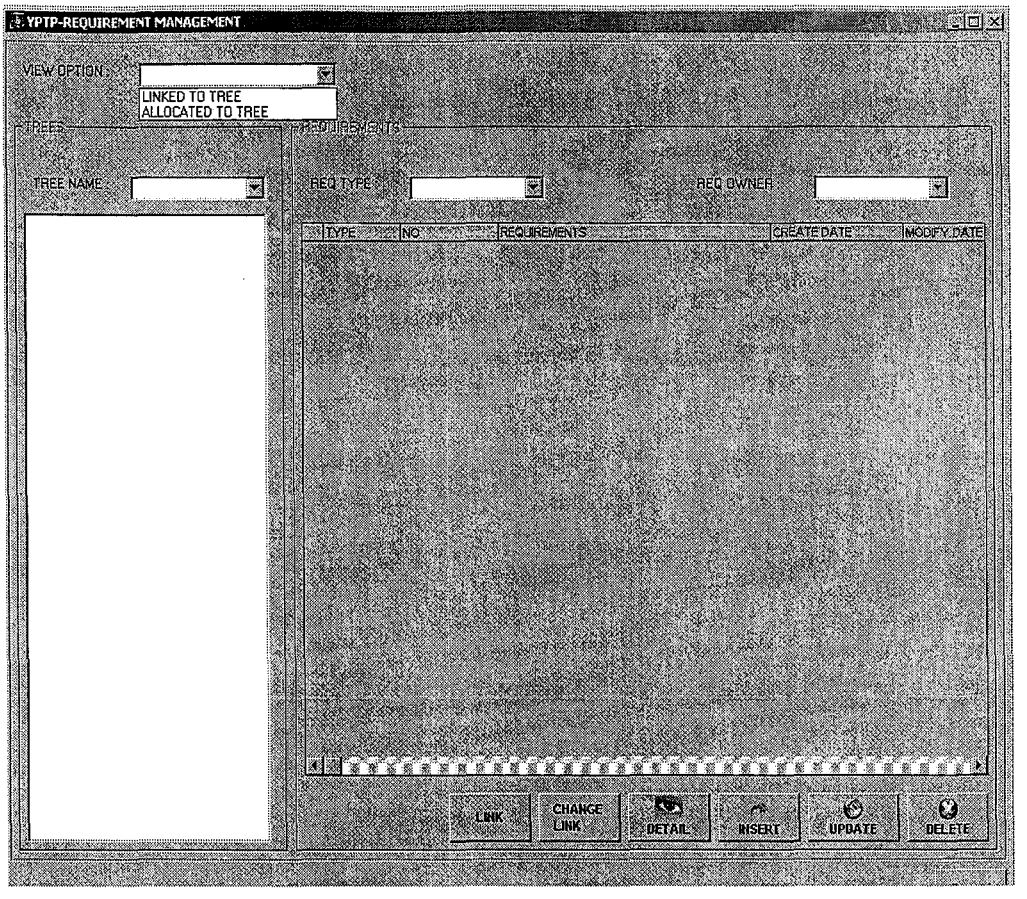
"Calendar Menu" üzerindeki "Calendar Definition" butonuna basıldığında ise proje süresince gerçekleştirilecek süreç ve aktivitelerin başlangıç ve bitiş tarihlerinin tanımlanması için kullanılan ve Şekil 5.43'de görülen "Calendar

Definition" penceresi ekrana gelecektir. Bu sayfada süreç ve aktiviteler için tarih tanımlanabilir, önceden tanımlanmış bilgiler güncellenebilir yada silinebilir.



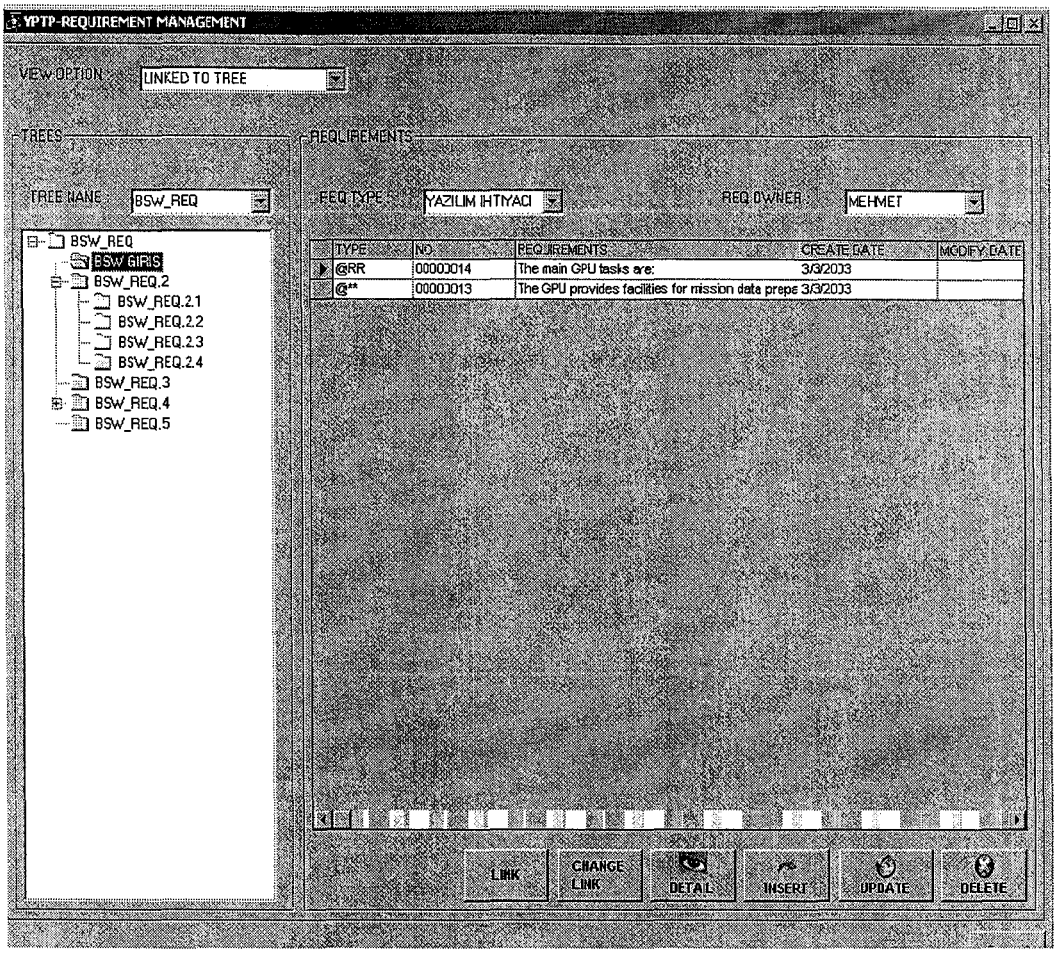
Şekil 5.44 Proje takvimi izleme sayfası

"Calendar Menu" üzerindeki "Trace Calendar" butonuna basıldığında şekil 5.44'de görülen pencere açılır. Proje ismi listesinden proje, süreç ismi listesinden süreç ve faaliyet listesinden faaliyet seçilmesi ile açılan pencere üzerinde projenin açılış tarihi, süreç ve aktivitenin başlangıç ve bitiş tarihleri, bu süreçte oluşturulması gereken dokümanlar, yapılması gereken toplantılar, çalışacak kişiler ve yine bu süreç içinde tanımlanan ağaçlar listelenir. Bu sayfa projeye ait oluşturulan takvim parçalarını izlemek için kullanılır.



Şekil 5.45 İhtiyaç yönetimi ana sayfası

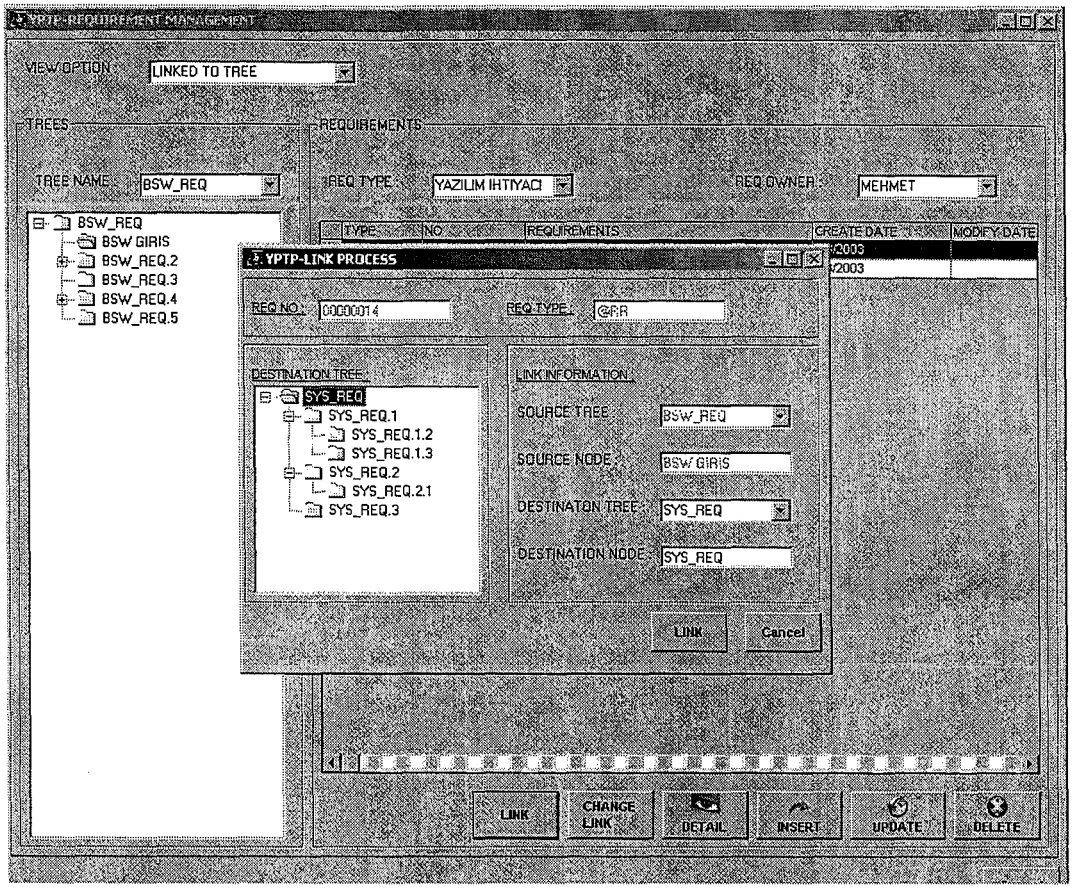
Ana menü üzerindeki "Requirements Management" butonuna basılırsa Şekil 5.45'deki pencere boş olarak açılır. Bu pencerenin amacı proje içinde ihtiyaçları tanımlamak, tanımlanan ihtiyaçları güncellemek yada silmektir. İhtiyaçlar bir ağacın kollarına tanımlanır, bu şekilde tanımlanan ihtiyaçlar sınıflandırılmış olur. Ağaç içinde tanımlanmış olan ihtiyaçların bu pencere üzerinde iki tane gösterim yolu bulunmaktadır; ağaca bağlantı ve ağaca atama.



Şekil 5.46 İhtiyaç yönetimi sayfası

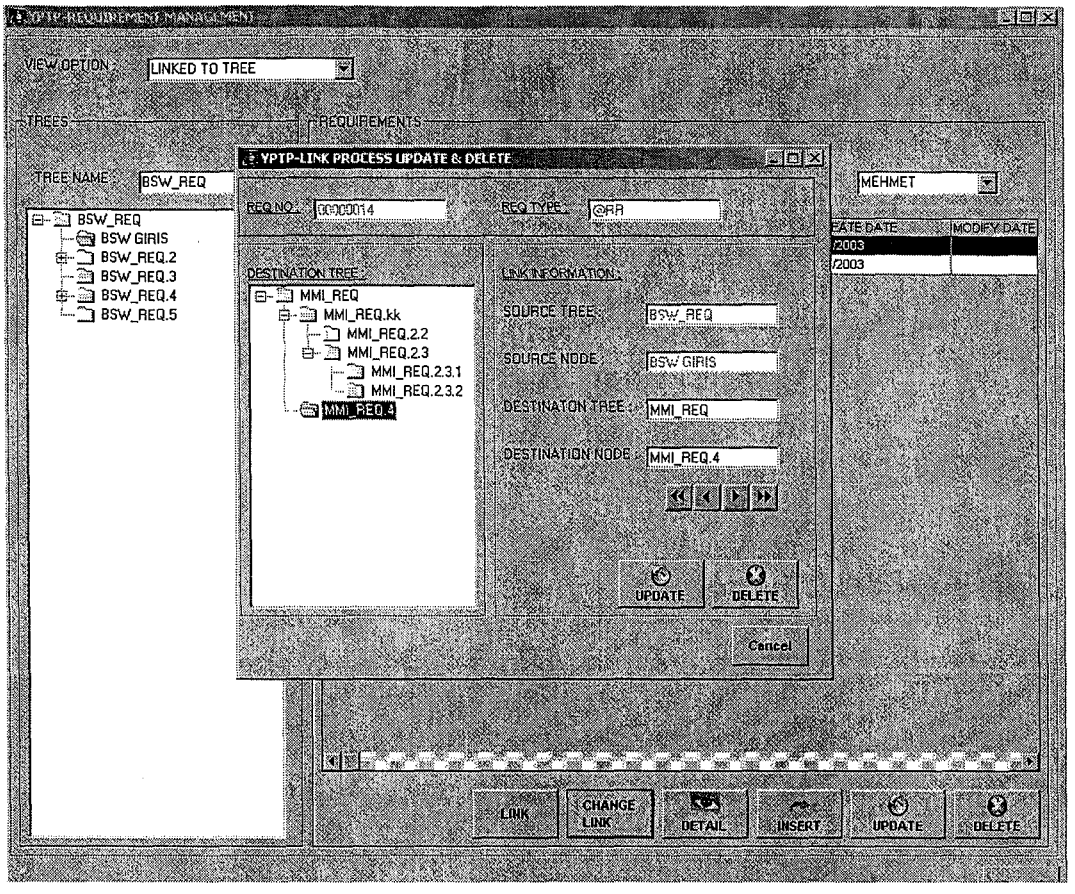
Şekil 5.46, ihtiyaç gösterim şekillerinden ağaca bağlantı seçildiğinde ekranda görünecek penceredir. Fare ile ağaç kollarına tıklandığında pencerenin sağ tarafında o kola ait ihtiyaçlar listelenir. Liste içinde ihtiyacın tipi, ihtiyaç numarası, ihtiyacı tanımlayan kişi, ihtiyaç açıklaması, ihtiyacın tanımlandığı ve güncellendiği tarih bilgileri görülür. Liste üzerinde görünen bu bilgiler alt kısımda bulunan "Insert", "Update" ve "Delete" butonları yardımı ile güncellenebilir, silinebilir yada yeni ihtiyaçlar tanımlanabilir.

Proje içerisinde tanımlanan bir ihtiyaç birden fazla ağacın parçası olabilir. Bu gibi durumlarda başka bir ağaç için de aynı ihtiyacı tanımlamak yerine mevcut ihtiyaç diğer ağaca da bağlanır.



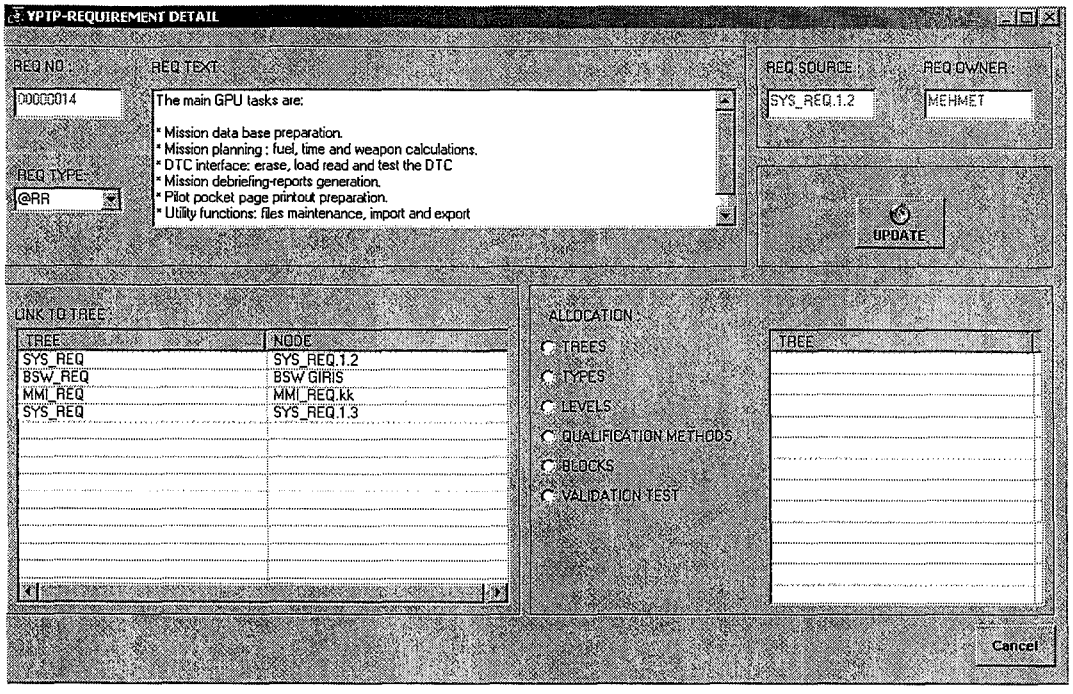
Şekil 5.47 İhtiyaç yönetimi bağlantı sayfası

"Requirements Management" penceresi üzerindeki "Link" butonuna basıldığında ekrana Şekil 5.47'de görülen pencere açılır. Pencerenin üst kısmında bağlantısı yapılacak olan ihtiyaç numarası ve ihtiyacın tipi görüntülenir. Sol kısımda ihtiyacın ilk tanımlandığı ağaç ve kol ismi yer almaktadır. Kullanıcı ihtiyacı bağlamak istediği diğer ağacın ismini yine aynı bölüm üzerindeki hedef ağaç ismi listesinden seçebilir. Hedef ağaç isminin seçilmesi ile pencerenin sol bölümünde seçilen hedef ağaç ve kolları görünür. Kullanıcı fare ile istenilen kola ulaşarak hedef kol adını kolayca belirleyebilir. İşlem tamamlandığında "Link" butonuna basılırsa ihtiyaç seçilen kaynak ağacın koluna bağlanmış olur.



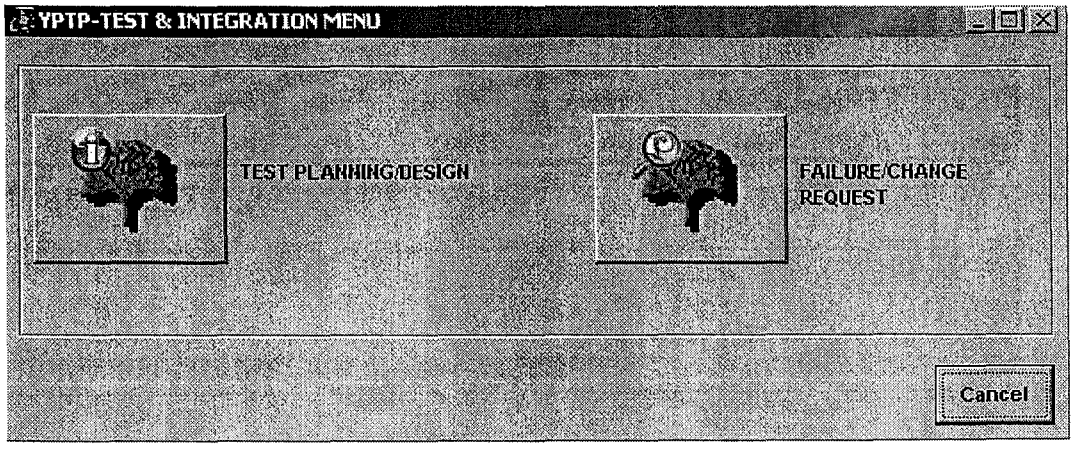
Şekil 5.48 İhtiyaç bağlantıları güncelleme ve silme sayfası

Kaynak ve hedef ağaç kol isimleri tanımlanmış ihtiyaçları güncellemek yada silmek için Şekil 5.48'de görülen "Link Process Update & Delete" sayfası açılır. Bu sayfaya erişim "Requirement Management" penceresi üzerinden "Change Link" butonuna basılması olur.



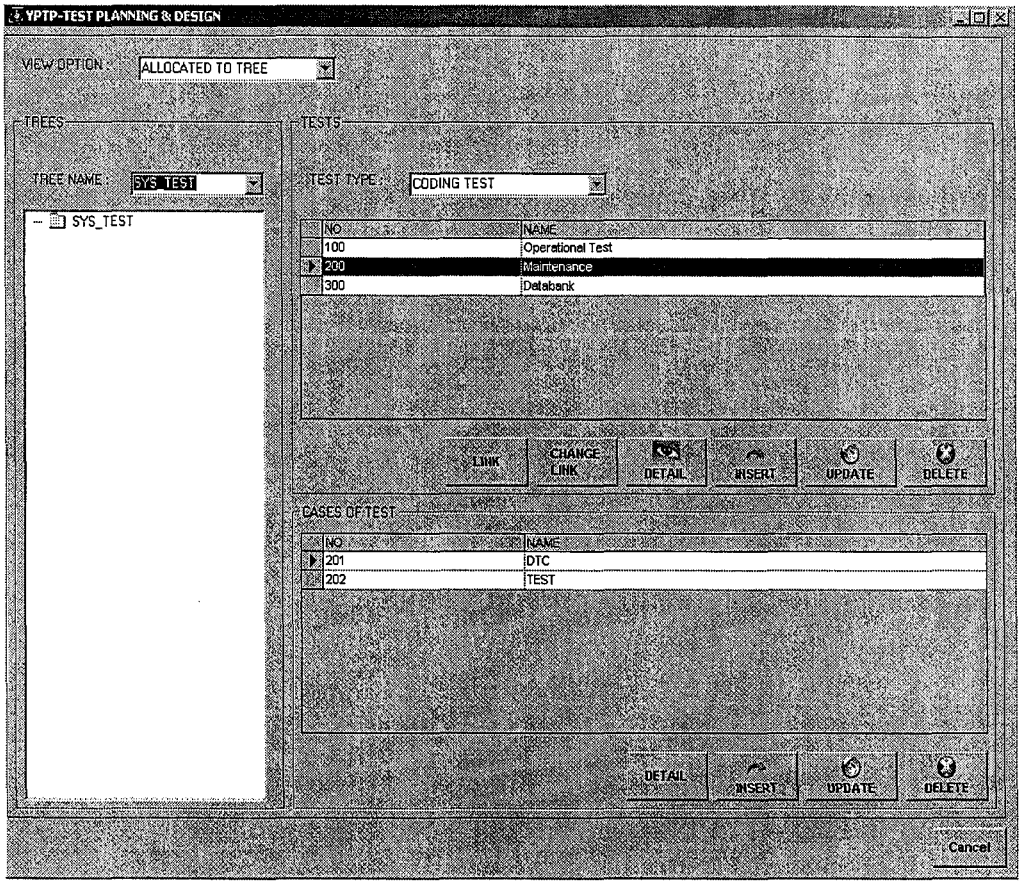
Şekil 5.49 İhtiyaç detayları tanımlama sayfası

Tanımlanan bir ihtiyaç hakkında daha detaylı bilgi görebilmek için Şekil 5.49'da görülen "Requirement Detail" penceresi hazırlanmıştır. Bu pencereye ulaşmak için "Requirements Management" penceresi üzerindeki "Detail" butonuna basılması gerekmektedir. Bu sayfa üzerinde bir ihtiyacın sahip olabileceği tüm bilgiler yer almaktadır.



Şekil 5.50 Test ve entegrasyon menüsü

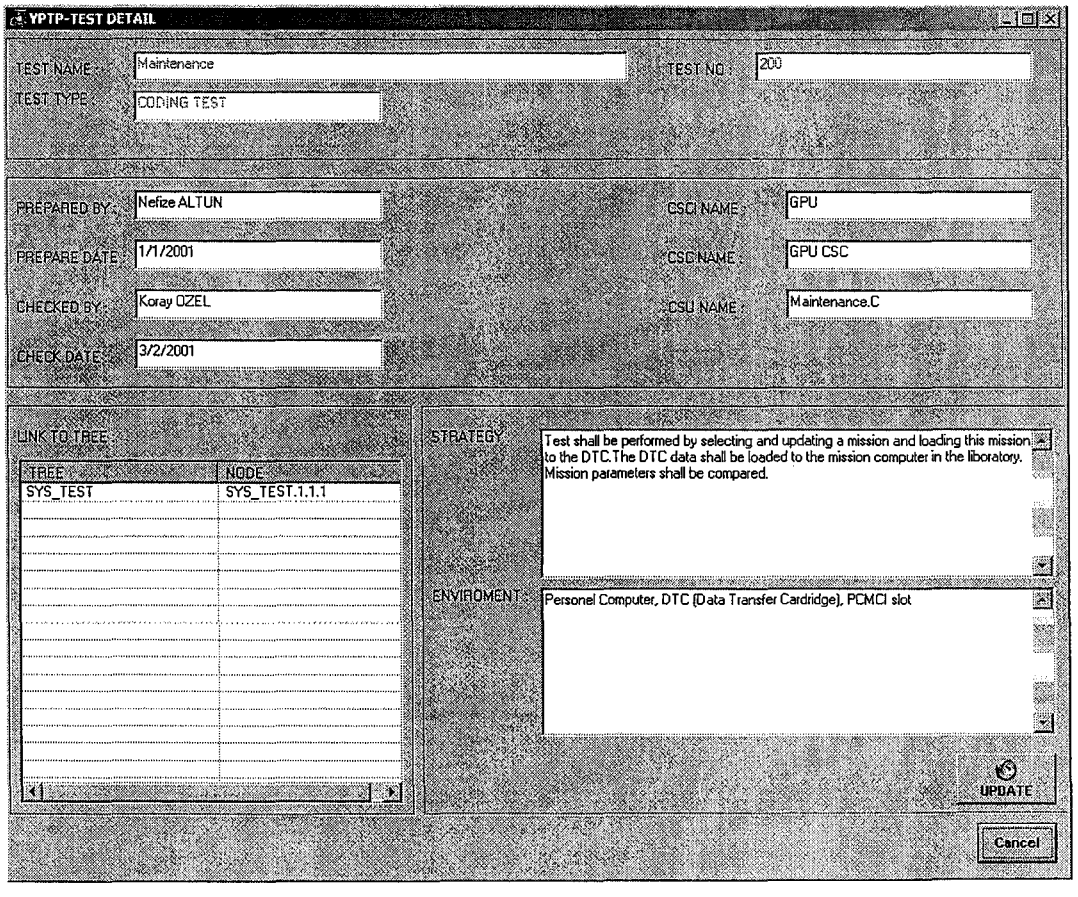
Şekil 5.50'de görülen test ve entegrasyon menüsü ana menü üzerindeki "Test and Integration Management" butonuna basıldığında ekrana gelen bir diğer alt menüdür. Bu menü üzerinde bulunan "Test Planning/Design" butonuna basıldığında ise ekrana Şekil 5.51'de görülen ve yapılan testlerin tanımlanmasını sağlayan pencere gelir.



Şekil 5.51 Test planlama sayfası

Test gösterimlerinin de ihtiyaçlardaki gibi iki yolu vardır. Ağaca bağlantı ve ağaca atama. Ağaca atama gösterim şekli ile seçilen ağaç üzerinde tanımlı tüm testlerin listesi ekranda görülür. Ağaca bağlantı gösterim şeklinde ise testlerin listesini görebilmek için testlerin tanımlı olduğu ağacın koluna gidilmesi gerekmektedir. Testler, test basamaklarından ve test basamakları da birim testlerden oluşurlar. Test planlama ve tasarım sayfasında bu sebeple testler ve içerdikleri test basamakları görülmektedir. Fare ile testlerden biri üzerine gelip o satır seçildiğinde alt kısımdaki listede bu testin içerdiği tüm test basamakları listelenir. Bir testi oluşturan tüm bilgiler listede görüldüğü gibi test numarası ve

ismi değildir. Testler hakkında daha detaylı bilgilerin kayıt altına alınması için seçili herhangi bir test varken testler bölümündeki "Detail" butonuna basılması gerekmektedir. Bu durumda ekranda test ayrıntısı adı verilen ve Şekil 5.52'de görülen bir başka pencere açılacaktır. Bu pencere üzerindeki isim, numara ve test tipi bilgileri bu sayfa üzerine otomatik olarak gelen ve bu sayfadan değiştirilemeyen bilgilerdir. Sayfa üzerinden girilebilecek diğer bilgiler ise testin kimin tarafından hazırlandığı, hazırlanma tarihi, testin kim tarafından yapıldığı, testin yapılma tarihi, testin ait olduğu yazılım birimi, bileşeni ve ünite ismi, uygulanan test stratejisi, test ortamı ve testin bağlandığı ağaç isimleridir. Bilgiler girildikten yada değiştirildikten sonra ekran üzerindeki "Update" butonuna basılması yapılan değişikliğin veritabanına kayıt edilmesi için yeterlidir.

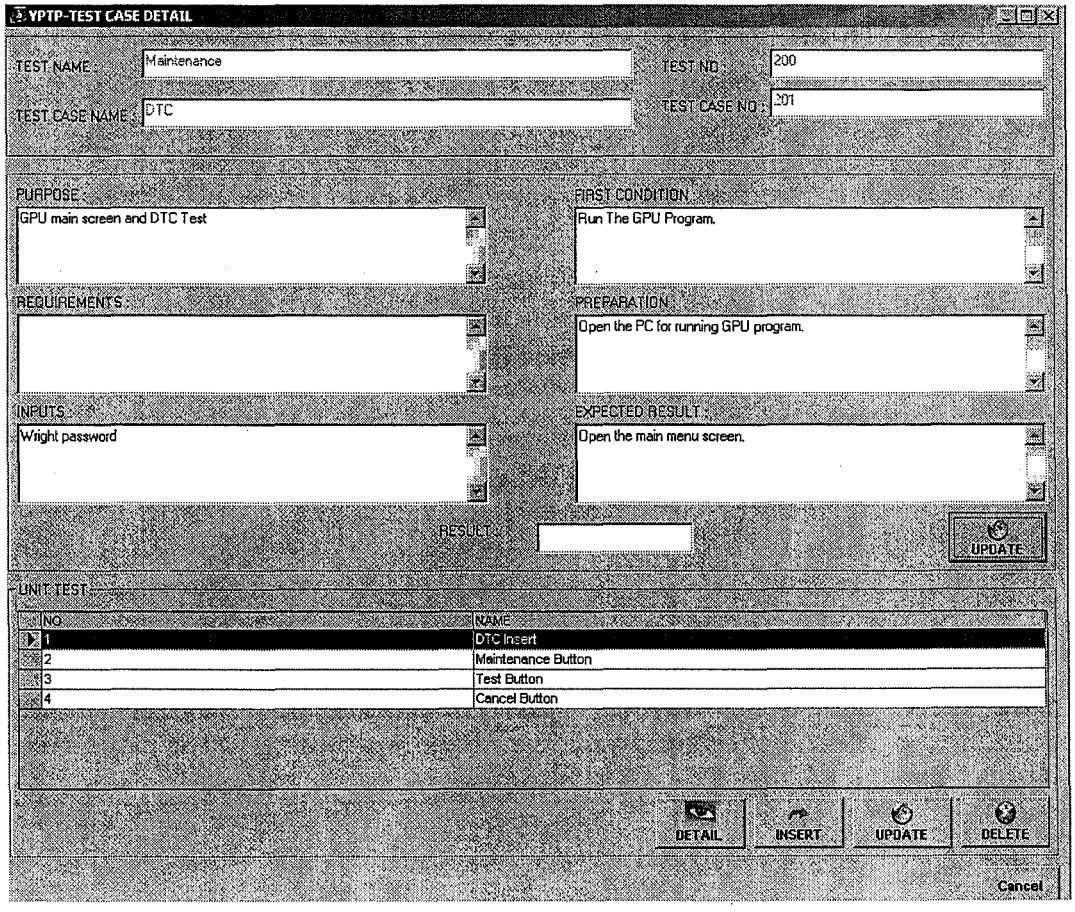


Şekil 5.52 Test detayları tanımlama sayfası

Yine ihtiyaçlarda olduğu gibi testleri de farklı ağaçlara bağlamak ve böylelikle veritabanında tekrar yapılmasından kurtulmak mümkündür. Yöntem burada da aynıdır. Testler bölümündeki "Link" butonuna basılarak açılan sayfada

tanımlanan testi başka bir ağaca bağlamak yada "Change Link" butonuna basılarak açılan sayfada testlerin bağlı olduğu ağaç yada kollarını değiştirmek mümkündür.

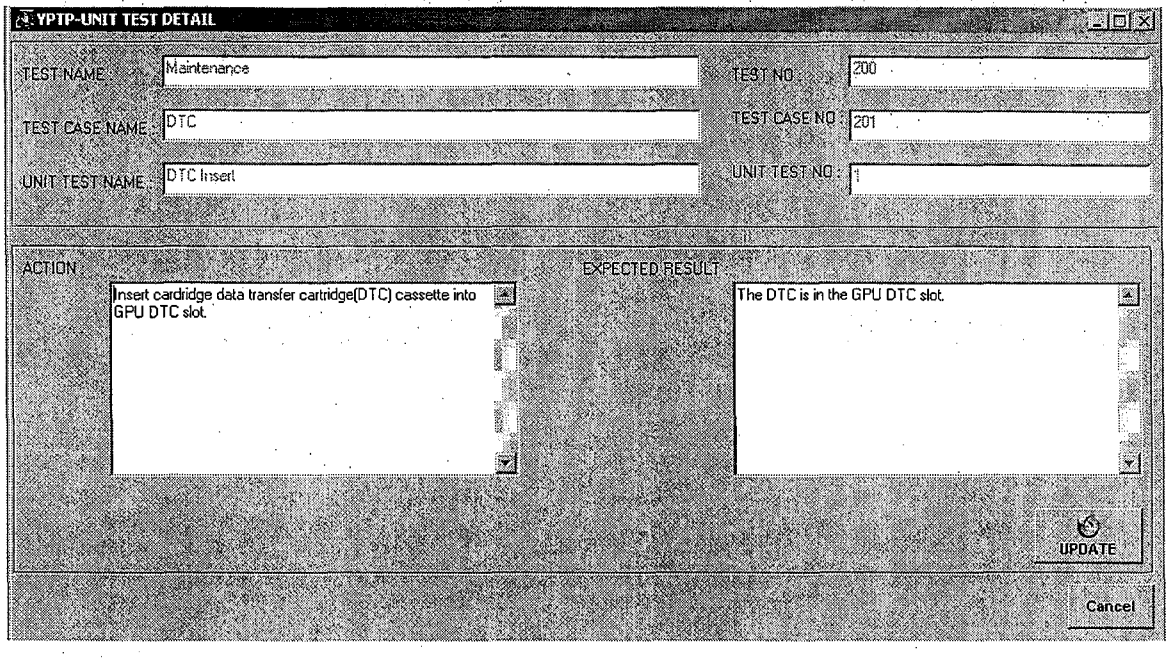
Şekil 5.51 üzerinde test basamakları için ayrılan bölümde listelenen herhangi bir test basamağını fare ile seçip "Detail" butonuna basıldığında Şekil 5.53'de görülen ve test basamakları ayrıntısı adı verilen sayfa ekrana gelir. Bu sayfa üzerindeki test adı ve numarası, test basamak adı ve numarası bilgileri de sayfaya otomatik olarak gelen ve değiştirilemeyecek bilgilerdendir. Sayfanın değiştirilebilecek bilgileri ise test basamağının amacı, karşıladığı ihtiyaç numarası, girdileri, beklenen çıktıları, testin ilk koşulları, test hazırlıkları ve sonuçtur.



Şekil 5.53 Test basamak detayları tanımlama sayfası

Test basamağı ayrıntılarının yer aldığı bu sayfada ayrıca bu test basmağına ait tüm birim testlerin isim ve numaralarını bir liste halinde görmek mümkündür.

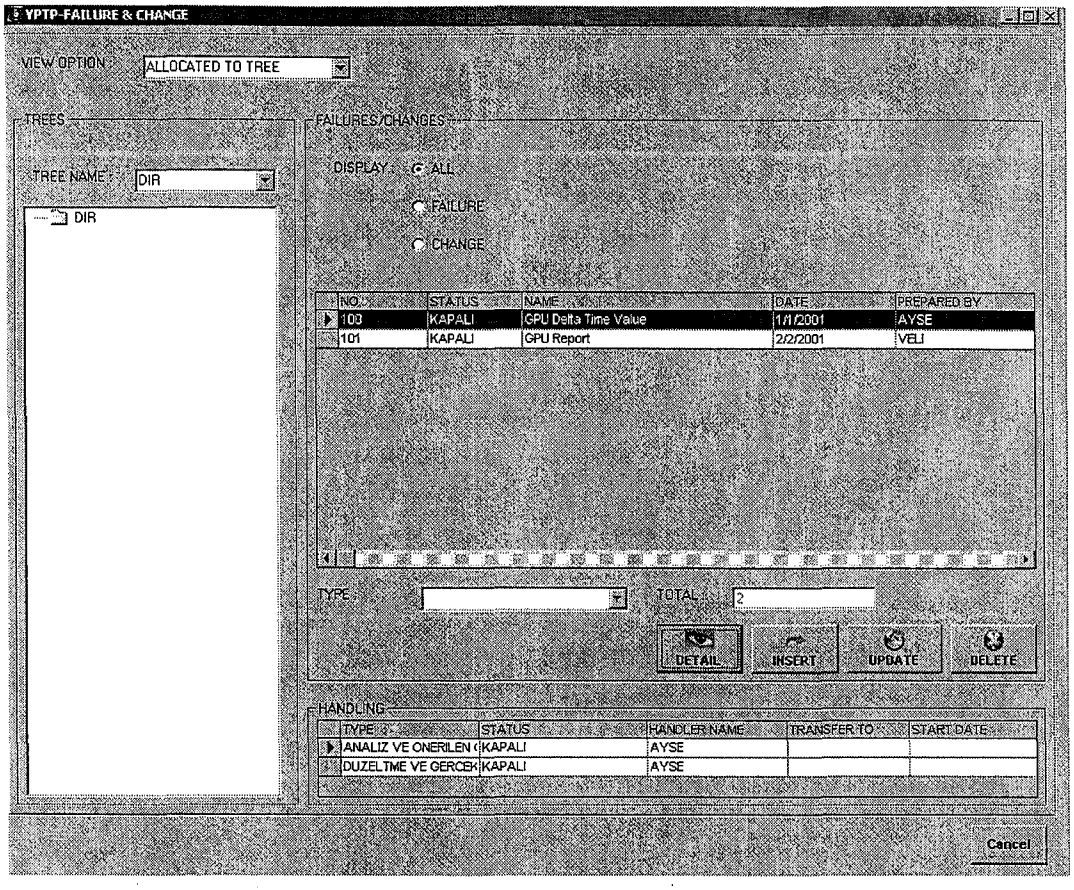
İstenilen birim testin detaylarını tanımlayabilmek yada değiştirebilmek için listedeki herhangi bir birim testi fare ile seçtikten sonra "Detail" butonuna basmak yeterlidir. Bu durumda ekranda Şekil 5.54'deki pencere görülecektir.



Şekil 5.54 Birim test detayları tanımlama sayfası

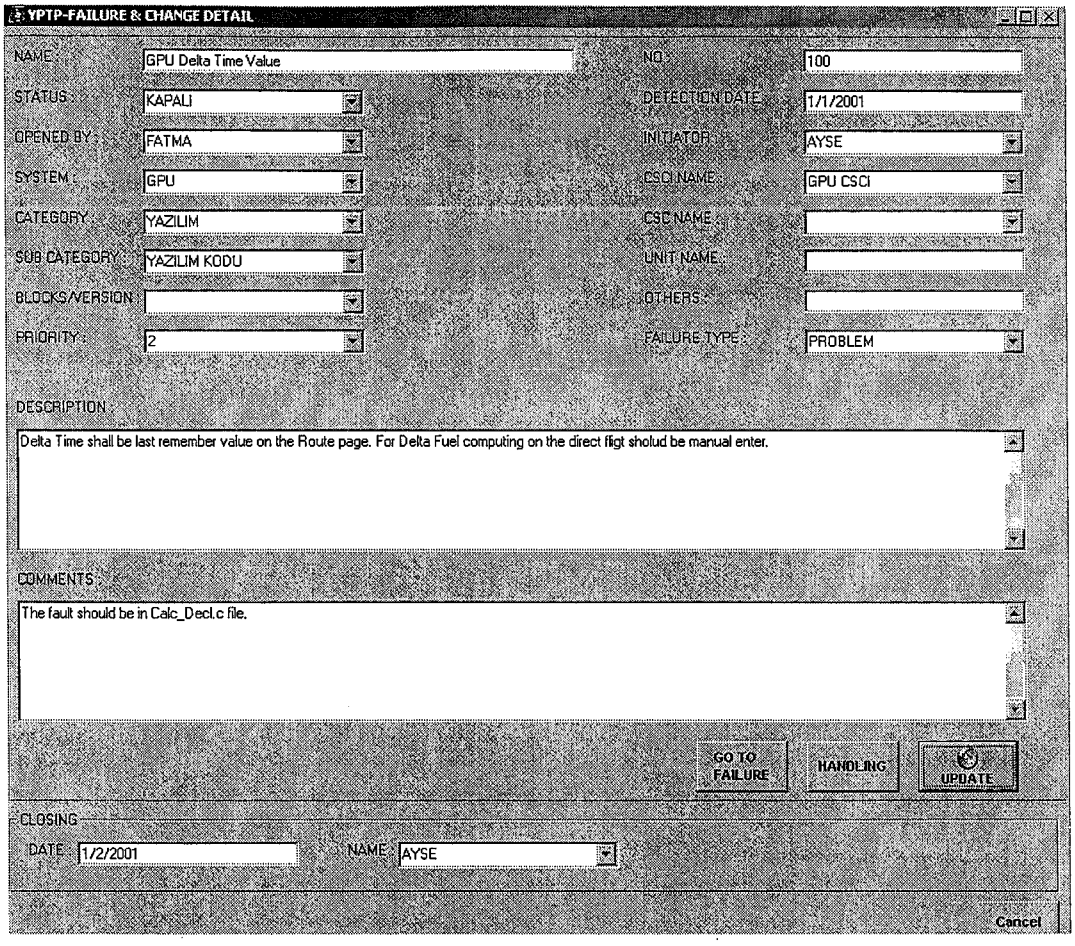
Test ismi ve numarası, test basamak ismi ve numarası, birim test ve numarası bilgileri ekrana otomatik olarak gelirken test faaliyeti ve beklenen sonuç bilgileri kullanıcı tarafından girilcektir. Burada bulunan "Update" butonu yeni tanımlanan yada güncelleştirilen bu bilgilerin veritabanına kayıt edilmesini sağlar.

Şekil 5.50'deki test ve entegrasyon menüsü üzerindeki "Failure/Change Request" butonuna basıldığında şekil 5.55'de görülen ve hata değişiklik sayfası olarak isimlendirilen pencere ekrana gelir. Pencere üzerinde ağaca atama gösterim şekli seçildiğinde pencere üzerinde liste halinde hataların numarası, hatayı bulan kişi, hatayı bulma tarihi, hatanın ismi ve durumu bilgileri ile birlikte tüm hatalar listelenir. Pencerenin hatalar ve değişiklikler bölümünde hata, değişiklik ve hepsi gösterim şekilleri yer almaktadır. Bunlardan hata seçeneğinin seçilmesi durumunda sadece tüm hatalar, değişiklikler seçeneğinin seçilmesi durumunda sadece tüm değişiklikler, hepsi seçeneğinin seçilmesi durumunda ise hataların ve değişikliklerin tümü listelenecektir.



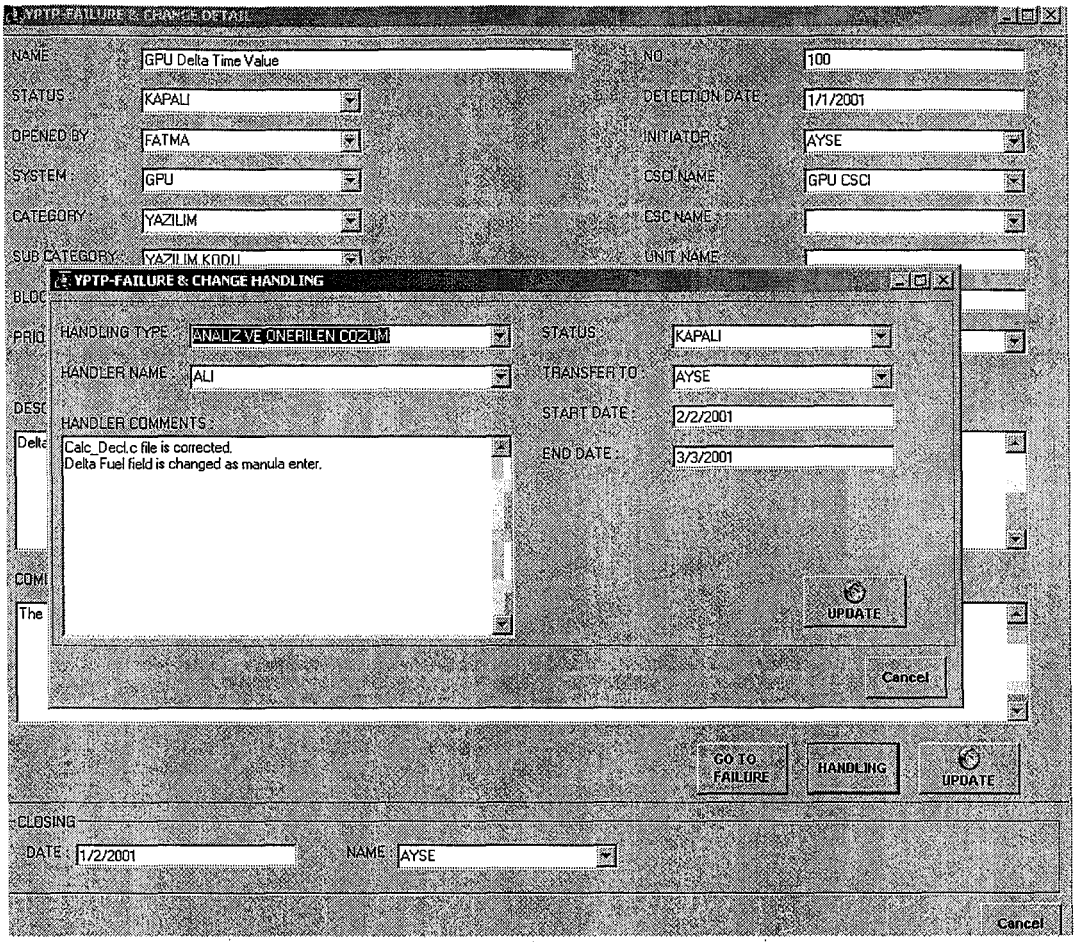
Şekil 5.55 Hata ve değişiklik sayfası

Pencerenin alt kısmında hatayı kontrol eden kişinin adı, hatanın durumu; hata bir başkasına transfer edilmişse transfer edildiği kişi ve tarihi bilgileri görüntülenmektedir. Hata hakkında daha detaylı bilgi tanımlamak yada bilgileri güncellemek istendiğinde fare ile hatanın seçilmesi ve "Detail" butonuna basılması yeterlidir. Bu durumda Şekil 5.56'da görülen pencere ekrana gelir.



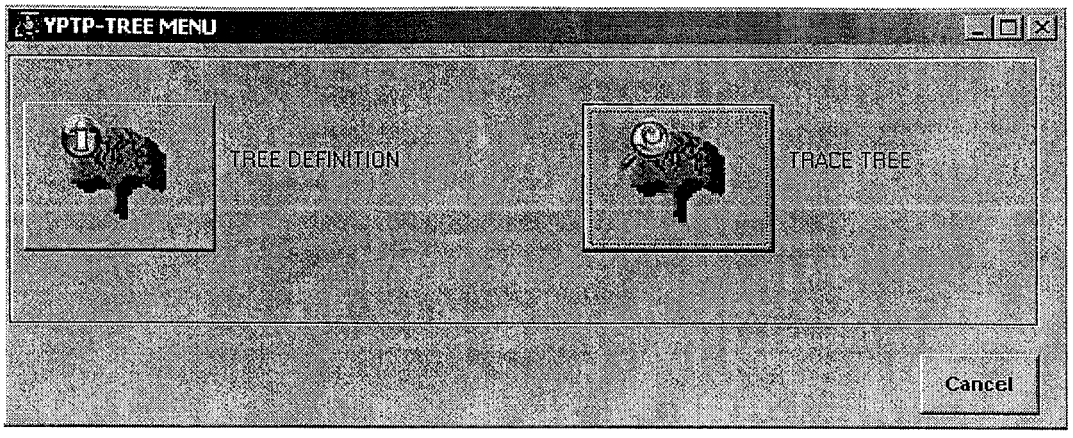
Şekil 5.56 Hata ve değişiklik detayları tanımlama sayfası

Yukarıda görüldüğü gibi bir hata yada değişiklik tanımlaması için pencere üzerinde bir alan ayrılmıştır. Hata ve değişiklik detayları sayfasında tanımlanan "Go To Failure" butonu ekrana gidilmek istenen hata/değişiklik numarasının yazıldığı bir pencere açar. "Handling" butonuna basıldığında ise hata/değişikliği kontrol altında tutan kişi ve yapılan işlemlerin tanımlanması için Şekil 5.57'deki pencere ekrana gelir.



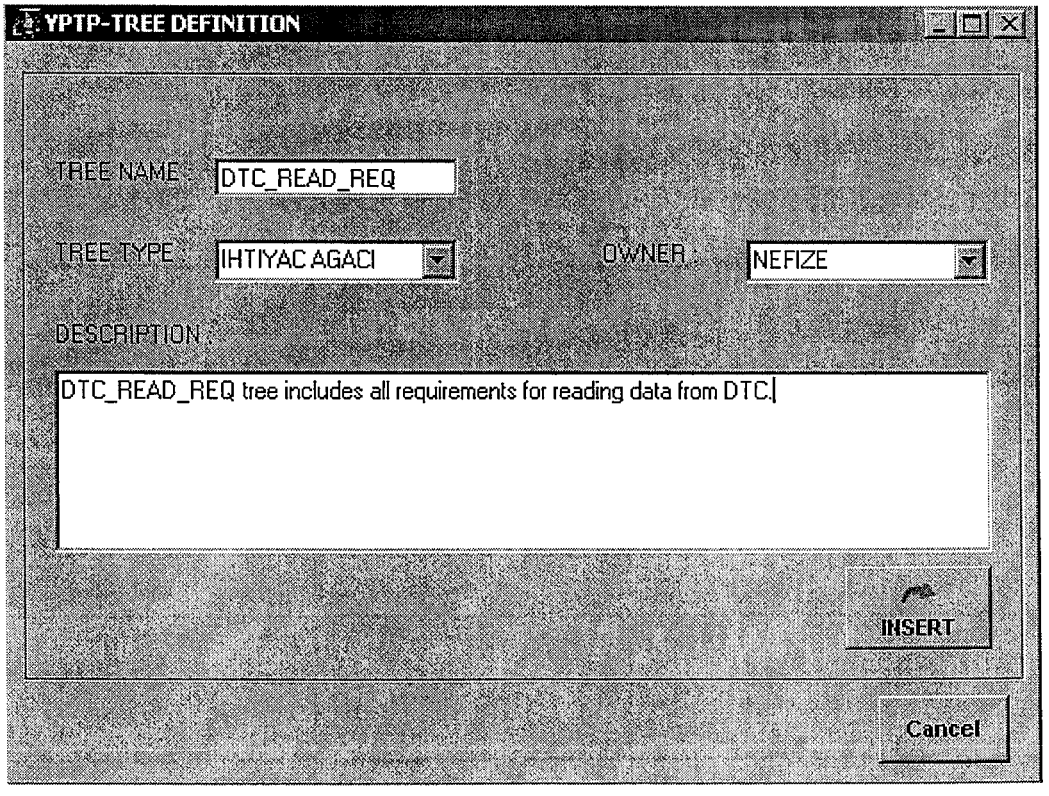
Şekil 5.57 Hata ve değişikliklerin değerlendirilme sayfası

Bu pencere üzerinde hatayı çözen kişinin adı, hatanın durumu, hatanın transfer edildiği kişinin adı, transfer tarihi, hatanın nasıl çözüldüğü ve çözümün bitirildiği tarih bilgilerinin tanımlanması için gerekli alanlar bulunmaktadır.

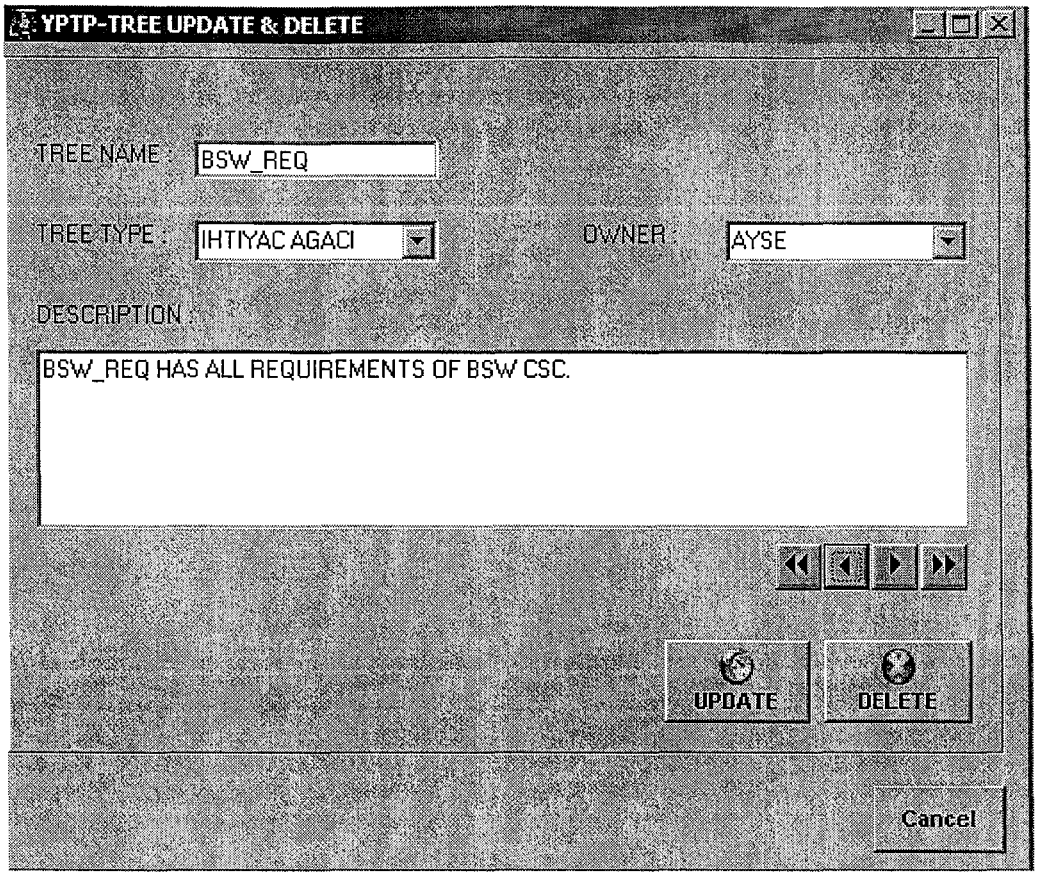


Şekil 5.58 Ağaç menüsü

Ana menü üzerinde "Trees Management" butonuna basıldığında ağaçlarla ilgili bir alt menü olan Şekil 5.58'deki pencere ekrana gelir. Bu pencere üzerindeki "Tree Definition" butonuna basıldığında ekrana şekil 5.59'da görülen ve bir ağaç tanımlamak için gerekli tüm bilgi kutucuklarını içeren pencere gelir.

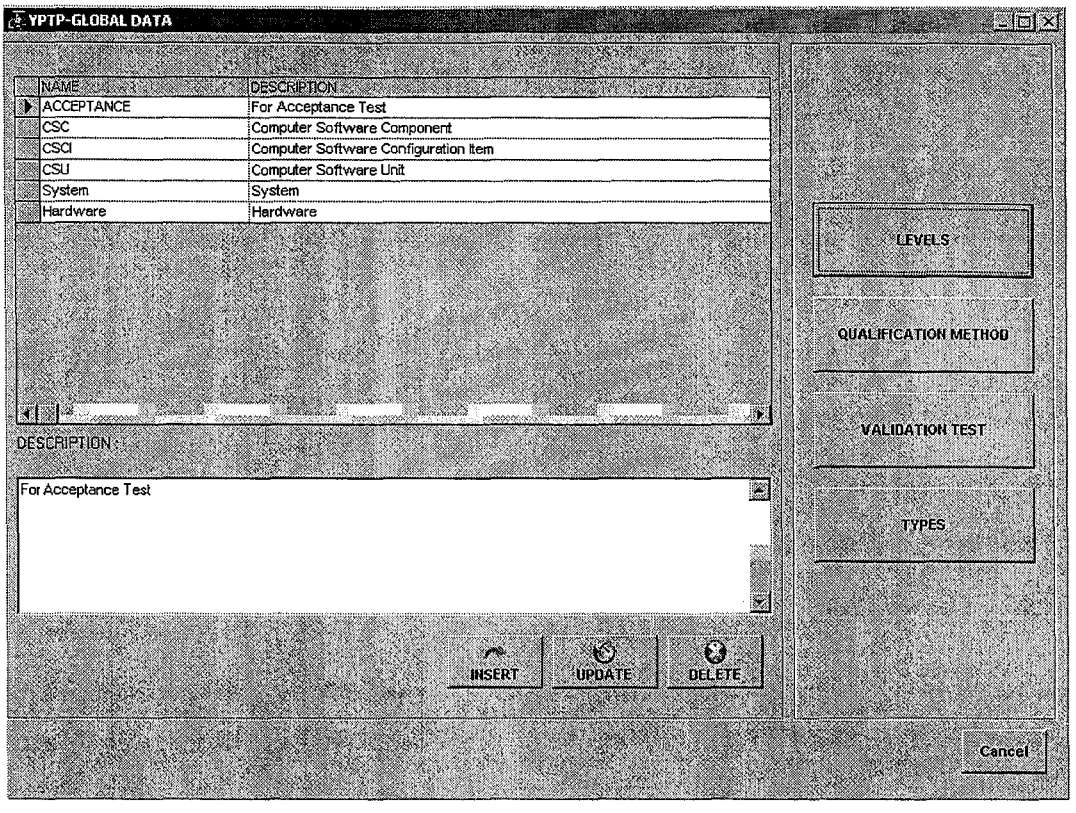


Şekil 5.59 Ağaç tanımlama sayfası



Şekil 5.60 Ağaç güncelleme ve silme sayfası

Ağaç menüsü üzerindeki diğer buton olan "Trace Tree" ye basıldığında ise ekrana Şekil 5.60'daki "Tree Update & Delete" penceresi gelir. Bu pencere üzerinden, tanımlı tüm ağaçlara ulaşılarak mevcut bilgilerde güncelleme yapılabilir yada ağaç silinebilir.



Şekil 5.61 Global veri sayfası

Ana menü üzerinde "Global Data" butonuna basıldığında Şekil 5.61'deki pencere ekranda görünür. Bu pencere ile her biri projenin birer global verisi olarak tanımlanabilen seviyeler, kalite metodları, değerlendirme testleri ve tipler tanımlanabilir, güncellenebilir yada silinebilir.

6. TARTIŞMA VE SONUÇ

Bu tez çalışması IEEE/EIA 12207 yazılım geliştirme standardına uygun olarak çalışan bir kurumun kendisini seçtiği standarda uygun çalışıp çalışmadığı konusunda denetleyebilmesi, seçtiği kalite standardına kontrollü bir şekilde uyabilmesi amacı ile hazırlanmış bir programdır. Bu sayede kurum konfigürasyon yönetimini sağladığı gibi proje takvimlendirme ve idamesini de yapabilecektir.

Bilgisayar ağı ile proje çalışanlarının hepsine ancak yetkileri ölçülerinde hizmet verecek YPTP programı, projelerindeki ihtiyaçları, testleri, tasarım spesifikasyonlarını, bunlar arasındaki izlenebilirlik ve arabirimleri yöneten bir veritabanı programıdır. Böylelikle proje mühendisleri arasında daha iyi bir iletişim sağlanmış olur. YPTP programı yetkilendirilmiş tüm kullanıcılar için ulaşılabilirdir ve kullanıcıların son gelişmelere göre güncelleştirilmiş bilgiye erişimini sağlar. İhtiyaçların, geliştirmenin tüm aşamaları içinde tamamlanma safhalarını izleyerek yönetimini temin eder. Sistem ihtiyaçları; yazılım mühendisliği, donanım mühendisliği ve geliştirme disiplinlerine yönlendirilir. Yazılım ve sistem test yönetimi icra edilir. Bilgi tamamlanması ve yönetimi ile birlikte arabirim yönetimi gerçekleştirilir. Bunları yaparken zaman ve maliyet kaybını aza indirir, daha fazla çalışanla daha az iş yapılmasının önüne geçer.

YPTP programı üç kısımdan oluşturulmuştur; verileri saklayan ve birbirleri ile ilişkilendirilmiş tablolardan oluşan veritabanı, Visual Basic 6.0'da hazırlanan kullanıcı arayüz programı ve arayüz programının veritabanından istediği sorguları, güncellemeleri, ekleme yada çıkarmaları yapmaya yarayan, PL/SQL dilinde yazılmış yardımcı alt program.

Veritabanının Oracle 8i'de hazırlanmış olması gerektiğinde veritabanı tasarımının geliştirilmesine ve büyütülmesine imkan vermektedir. Veritabanındaki tablolar, görüntüler normalize edilerek ilişkilendirilmiş ve tasarımın güncellenmesi ihtimaline göre modüler olarak tasarlanmıştır.

Visual Basic 6.0 kullanımı rahat, öğrenimi ve takibi kolay bir programlama dili olmasından dolayı YPTP programında herhangi bir değişiklik olması durumunda kodlamayı yapacak kişiler farklı olsa bile bu herhangi bir zorluk çıkarmayacaktır.

Program kapsamındaki sorgu ihtiyaçlarının değişmesi durumunda ise sadece PL/SQL ile yazılımış yardımcı alt programın değiştirilmesi yeterli olacaktır. PL/SQL kodunda yapılan bir değişiklik veritabanını yada kullanıcı arayüz programını etkilemeden gerçekleştirilebilir.

YPTP programı geliştirilmeye açık bir programdır. Önemli olan programa eklenmek istenen özelliklerin uygulanan yazılım geliştirme standardını desteklemesidir. Program içerisinde şablonları bir ağaç şeklinde oluşturulacak dokümanların program tarafından otomatik olarak üretilmesi sağlanabilir. Tanımlanan sistem ihtiyaçları, yazılım ihtiyaçları ve bunların doğruluğunu test etmeye yarayan test basamakları arasında izlenilebilirliği sağlamak amacı ile bir iz matrisi oluşturulabilir.

YPTP uygulamasının, IEEE/EIA 12207 yazılım yaşam döngüsü süreçlerine uygun olarak çalışan bir kurumun, yazılım projeleri ile ilgili çalışmalarında standarda uygunluğunun sağlanması için kullanılması planlanmaktadır. Uygulama programının modüler bir yapıda tasarlanması geliştirilmeye açık olarak bırakılan bu programın kurum isteklerine göre tasarlanmaya devam edilebileceği anlamını taşımaktadır.

KAYNAKLAR

1. MIL-STD-498, *Software Development and Documentation.*
2. TRIPP,L. ve DEWEESE,P., Joint Industrial Standards Working Group Members, *12207.0 Software Life Cycle Processes*, USA (1998).
3. TRIPP,L. ve DEWEESE,P., Joint Industrial Standards Working Group Members, *12207.1 Software Life Cycle Processes - Life Cycle Data*, USA (1998).
4. TRIPP,L. ve DEWEESE,P., Joint Industrial Standards Working Group Members, *12207.2 Software Life Cycle Processes - Implementation Consideration*, USA (1998).
5. THAYER, R., *Visual Basic 6.0 Unleashed*, Sams Publishing, USA (1998).
6. TOMLINSON,J., FEDYNICH,J. ve BESAW,J., *Oracle And Visual Basic Developer's Handbook with CDROM*, Sybex Publishing, USA (2000).
7. SNOWDON, N., *Oracle Programming with Visual Basic*, Sybex Publishing, USA (1998).
8. LEVERENZ,L. ve REHFIELD,D., *Oracle 8i 8.1.5 Documantation*, USA (1999).
9. RAPHAELY,D., *Oracle 8i Application Developer's Guide Fundamentals Release 8.1.5 Documantation*, USA (1999).