T.C.
AKDENİZ ÜNİVERSİTESİ
SAĞLIK BİLİMLERİ ENSTİTÜSÜ
Biyoistatistik Anabilim Dalı

HASTANE MERKEZİ LABORATUAR SİSTEMİNDEKİ VERİTABANINDA GEREKLİ İSTATİSTİKİ ANALİZİN YAPILABİLMESİ İÇİN LABORATUAR SİSTEMİNE SPSS PAKET PROGRAMININ ENTEGRE EDİLMESİ İÇİN BİR ARA YÜZ YAZILIMI GELİŞTİRMEK

Mümtaz Serkan ÖZKAYA

Yüksek Lisans Tezi

Tez Danışmanı
Öğr. Gör. Dr. Mehmet YARDIMSEVER

Antalya, 2003

Sağlık Bilimleri enstitüsü Müdürlüğü'ne;

Bu çalışma jürimiz tarafından Biyoistatistik Anabilim Dalı'na bağlı Tıp Bilişimi Programında yüksek lisans tezi olarak kabul edilmiştir. ../../2003

Tez Danışmanı : Öğ. Görev. Dr. Mehmet Yardımsever
Akdeniz Üniversitesi
Tıp Fakültesi
Biyoistatistik Anabilim Dalı

Üye : Prof. Dr. Osman Saka
Akdeniz Üniversitesi
Tıp Fakültesi
Biyoistatistik Anabilim Dalı

Üye : Prof. Dr. Ergün Karaağaoğlu
Hacettepe Üniversitesi
Tıp Fakültesi
Biyoistatistik Anabilim Dalı

Üye : Prof.Dr. Gültekin Süleymanlar
Akdeniz Üniversitesi
Tıp Fakültesi
Nefroloji bilim Dalı

Üye : Prof.Dr. Mehmet Aktekin
Akdeniz Üniversitesi
Tıp Fakültesi
Halk Sağlığı Anabilim Dalı

ONAY:

Bu tez, Enstitü Yönetim Kurulunca belirlenen yukarıdaki jüri üyeleri tarafından uygun görülmüş ve Enstitü Yönetim Kurulu'nun ../../2003 tarih ve ..... sayılı kararıyla kabul edilmiştir.

Prof. Dr. Ramazan Demir
Enstitü Müdürü

ÖZET

Bu çalışmada, Akdeniz Üniversitesi Tıp Fakültesi Hastanesi merkezi laboratuarında bulunan veritabanı üzerinden çekilen belirli özellikteki verilerin, SPSS paket programının anlayabileceği bir veri dosyası haline getirilmesi ve bu modülün geliştirilebilmesi için kullanılan hastane bilişim sistemi elemanları açıklanmıştır.

Oluşturulan bu veri dosyası üzerinde SPSS'in nesne otomasyonu kullanılarak istenen istatistiklerin uygulanması için gereken bir ara yüz yazılımı geliştirilmiştir. SPSS'in nesne otomasyonu bu uygulama içerisindeki açıklanmıştır.

Anahtar Kelimeler: Veri Madenciliği, Veritabanı, Hastane Bilişim Sistemi, SPSS, OLE otomasyonu, Nesne tabanlı programlama, SQL

ABSTRACT

In this study, we tried to explain the methods to transform specific data sets collected over Akdeniz University Medical Faculty Hospital Central Laboratory Database into a type which fits to SPSS package program usage and describe the Hospital Information System components that are used to develope this module.

This interface module was developed to compute the statistics of collected data sets by using SPSS OLE otomation. SPSS OLE otomation components are also described in this study.

Key Words: Data Mining, Database, Hospital Information System, SPSS, OLE Automation, Object Based Programming, SQL.

TEŞEKKÜR

Bu tezin hazırlanmasında yardımlarını esirgemeyen danışmanım sayın Dr. Mehmet Yardımsever'e, Sayın Prof. Dr. Osman Saka'ya, tezim süresince bana moral destek sağlayan mesai arkadaşlarım Evren Tercan, Neşe Zayim, Dr. Hakan Gülkesen, Filiz İşleyen ve Özgür Tosun'a teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

AÜHBS	Akdeniz Üniversitesi Hastane Bilişim Sistemi
CPU	Merkezi İşlem Birimi
DBMS	DataBase Management System
DDE	Direct Data Exchange
DLL	Dynamic Link Library
DML	Data Manipulation Language - Veri İşleme Dili
HBS	Hastane Bilişim Sistemi
HL7	Health Level Seven
IP	Internet Protokol
LBS	Laboratuar Bilişim Sistemi
LIS	Laboratory Information System
NTBS	Nükleer Tıp Bilişim Sistemi
ODBC	Open DataBase Connectivity
OLE	Object Linking and Embedding-Nesne Bağdaştırma ve Katıştırma
PACS	Picture Archiving and Communication System
PBS	Patoloji Bilişim Sistemi
RAD	Rapid Application Development
RAM	Random Access Memory
RBS	Radyoloji Bilişim Sistemi
RI	Relational Integrity
RISC	Reduced Instruction Source Code
RPC	Remote Processing Call
SMP	Symetrical Multiple Processing
SPSS	Statistical Package for Social Sciences
SQL	Structural Query Language
VTYS	VeriTabanı Yönetim Sistemi
XML	Extensible Markup Language

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

GİRİŞ VE AMAÇ

Bilişim sistemleri XX. Yüzyılın son çeyreğinde gerek öncesinde elde edilen birikimin değerlendirilmesi gerekse daha sonrasında elde edilmesi ön görülen bilimsel ve teknolojik gelişmelere ışık tutması açısından çok hızlı bir ilerleme göstermiştir. Bu ilerleme XX. Yüzyıla gelinceye kadar insanlığın elde ettiği bilgi altyapısının son yüzyıl içerisinde logaritmik artışının bir gerekliliğidir.

Bilginin en hızlı arttığı alanlardan birisi de hiç kuşku yok ki sağlık bilimleridir. Sağlık bilimlerindeki bilgi artışının en güzel göstergesi olarak Index Medikus'un, hacmindeki değişim gösterilebilir. Bilginin katlanarak arttığı sağlık bilimlerine bilişim sistemlerinin girişi çok fazla geçmişe dayanmamaktadır. Son yirmi yılda sağlık bilimleri bilişim sistemlerinin sağladığı olanaklardan yararlanmaya başlamasına rağmen etkin kullanımı 10-15 yıldan öteye gitmez. Gelişmiş ülkeler kendi sağlık bilgi sistemlerini oluşturmaya başladıklarında önlerine birçok problem çıkmıştır. Bilişim sistemlerinin sağlık bilgi sistemlerine uygulanmaya başlanmasıyla birlikte, son derece faydalı olacağı düşüncesiyle kurulmuş birçok sağlık bilişim sistemi projesi, yeterli bilginin olmayışı ve tecrübe eksikliği gibi sebeplerden dolayı hayal kırıklığıyla sonuçlanmıştır. Bunların en önemlisi İngiltere'nin yaşadığı tecrübedir. Çok büyük kaynaklar ayrılarak oluşturulan ulusal sağlık sistemleri hala günümüzde birçok sorunu barındırmaktadır. Bu gibi örnekler, gerçekleştirilecek bilişim projelerinin son derece iyi tasarlanmasının gerekliliğini ortaya koymaktadır. Aksi durumda büyük mali yüklerle karşı karşıya kalınabilmektedir.

Türkiye'de bilişim sistemlerinin sağlık bilimlerinde kullanılmaya başlanması son on yıl içerisinde mümkün olmuştur. Bütün dünyada olduğu gibi Türkiye'de de sağlık bilişim sistemlerinin kullanılmaya başlanması son 10-15 yıla dayanmaktadır. Bu süreçte ülkemizde birçok üniversite, devlet hastanesi özel hastaneler veya sağlık kuruluşları kendi olanaklarıyla bünyelerinde sağlık bilişim sistemlerini oluşturmaya çalışmışlarsa da hala ulusal bir sağlık bilgi sistemimizin olmayışı son derece büyük bir eksikliktir. Çünkü bütün sağlık kuruluşları gerek kendi bünyelerinde geliştirenler olsun gerekse sağlık bilişim sistemlerini satın alanlar olsun sağlık sistemindeki standardizasyonun sağlanamayışından dolayı sistemlerinde sürekli değişiklikler yapmak mecburiyetindedirler.

Akdeniz Üniversitesi Tıp Fakültesi 1989 yılından beri kendi çatısı altında oluşturduğu bilişim sistemini her geçen sene daha iyiye götürerek ülkemizdeki sağlık bilişim sistemlerine örnek oluşturabilecek bir yapıya kavuşmuştur. Bunun en önemli sebebi sistemin tamamen hastane bünyesinde geliştirilmesidir. Çünkü her sistem, kendi bünyesinde standartların henüz tam oturmamasından dolayı -bu durum şu an için bütün dünyadaki sağlık bilişim sistemleri için geçerli- çalıştırıldıkları sağlık kuruluşuna bağlı olarak farklılıklar içermektedir. Dolayısıyla sisteme müdahale yeteneğine sahip bilişim

elemanlarının bilişim sistemlerini geliştirmesi son derece önemlidir. Biyoistatistik Ana bilim Dalı altında Türkiye'de ilk olarak açılan önceki adıyla Tıbbi Bilgisayar Uygulamaları yeni adıyla Tıbbi Bilişim yüksek lisans programında eğitim görmüş birçok akademisyen sisteme eğitim gördükleri süre boyunca ellerinden gelen emeği vermişler ve sistemin geliştirilmesi için çaba harcamışlardır.

Bilişim sistemleri son derece büyük yatırımlar gerektiren sistemlerdir. Network altyapısından kişisel bilgisayarların arabirim kartlarına kadar her unsuru mali yükler getiren bu sistemler tasarlanırken çok dikkatli olunmalı ve en yüksek verimi sağlayacak şekilde tasarlanmalıdır. Bununla birlikte sistemin kurulumunda masraftan kaçmak uğruna çeşitli özverilerde bulunulması sistemin geliştirilebilirliğinin önünde engeller taşıdığı için sistemin zaman içinde çökmesi gayet normal bir sonuçtur. Önemli olan sistemin tasarımının en uygun şekilde gerçekleştirilip, ondan elde edilebilecek en yüksek faydanın sağlanmasıdır.

En yüksek verim bir bilişim sisteminden nasıl elde edilebilir? Verimlilik bir sistemden çıkan unsurların giren unsurlara oranı olarak değerlendirildiğine göre Hastane Bilişim Sisteminden (HBS) elde edilebilecek her türlü kullanılabilir, anlamlı çıktı sistemin verimliliğini artırılmasında bir etkendir.

Sadece bu açıdan bakıldığı zaman bile bilişim siteminin elemanları kullanılarak elde edilen faydaların artırılması gerekliliği ortadadır. Bir bilişim sisteminden fayda elde etmenin en iyi yollarından birisi sisteme girilen veya sistemde mevcut, veriler ve bilgilerin kullanılarak yeni bilgilerin oluşturulmasıdır.

Yukarda bahsi geçen verimliliğin artırılmasına yönelik olarak mevcut HBS kullanılarak sahip olduğumuz laboratuar verilerinin akademisyenlerin kullanımına açmak suretiyle yeni bilgilerin oluşturulmasına yönelik yeni bir istatistik modülü geliştirilmiştir. HBS'nde kullanılmakta olan MEDİSTAT modülüne entegre edilebilecek veya bağımsız olarak çalıştırılabilecek bu yeni modülün işlevi Statistical Package for Social Sciences (SPSS) paket programının Object Linking and Embedding (OLE) otomasyonu kullanılmak suretiyle ORACLE VeriTabanı Yönetim Sistemi (VTYS-DataBase Management System-DBMS) tarafından kontrol edilen Laboratuar Bilişim Sistemi'nden (LBS-Laboratory Information System-LIS) Yapısal Sorgulama Dili (Structural Query Language-SQL) ile yapılan sorgulamalar sonucu elde edilen veri dosyalarından akademik veriler elde etmek üzere istatistikler yapabilmesidir. Bu modül sayesinde Laboratuar verileri yetkili kullanıcıların kullanımına açılacak, araştırmacılar hasta verilerine ulaşmak için zaman ve işgücü harcamayacaktır.

Şu an için sadece akademik amaçlı veri toplamaya ve bunlardan istatistik elde etmeye yönelik olarak tasarlanan modül gelecekte farklı sorgulamaların sisteme eklenmesiyle hastane yönetimine karar desteği sağlamaya ilişkin istatistiklerin de elde edilmesi mümkün olacaktır.

Geliştirilen yeni istatistik modülünün mevcut modüle entegre edileceği düşünüldüğünden bundan sonraki bölümlerde yeni modül için Akademistat ismi kullanılacaktır.

GENEL BİLGİLER

2.1. Hastane Bilişim Sistemi Ve Bileşenleri

Bir bilişim sistemi bünyesinde birçok bileşeni bulundurur. HBS bileşenlerinin Akademistat ile ilişkilerini anlamak için bu bileşenlerin HBS içerisindeki işlevlerinin açıklanması gerekir[1]. Akademistat modülüyle etkileşim içerisinde bulunan temel HBS bileşenleri; VTYS, uygulama ara yüz yazılımları, intranet, sunucular, istemciler, SQL, SPSS ve nesne tabanlı programlama dili olarak Clarion 5.5 EE programlama dilidir.

2.2. VeriTabanı Yönetim Sistemi

Bilindiği gibi veri tabanları verilerin toplanması, sorgulanması ve üzerinde analizler yapılıp bağlantılar kurulması amacıyla hazırlanmış programlardır. Bu basit tanımlamanın dışında, Date, Martin ve Ashany sırasıyla veritabanını "bir işletmenin uygulama dizgelerince kullanılan, bellekte saklı ve işler durumdaki veriler bütünüdür", "bir veya birçok uygulamada kullanılmak için gereksiz yinelemelerden arınmış olarak bellekte saklanan ve birbirleriyle ilişkili veriler topluluğu" ve "veritabanı dizgelerinin çalışma alanı verilerin çözümlenmesi, düzenlenmesi, yorumlanması, sınıflandırılması, kaydedilmesi, güncelleştirilmesi, taranması ve bulunmasıdır" şeklinde tanımlamaktadır[2]. Bu tanımlamalardan veritabanlarının özellikleri çıkartılabilir. Ancak daha önce bilgisayarların teknolojik gelişimi ile veritabanlarının yapısal gelişimi arasında paralelliği irdelemek gerekir.

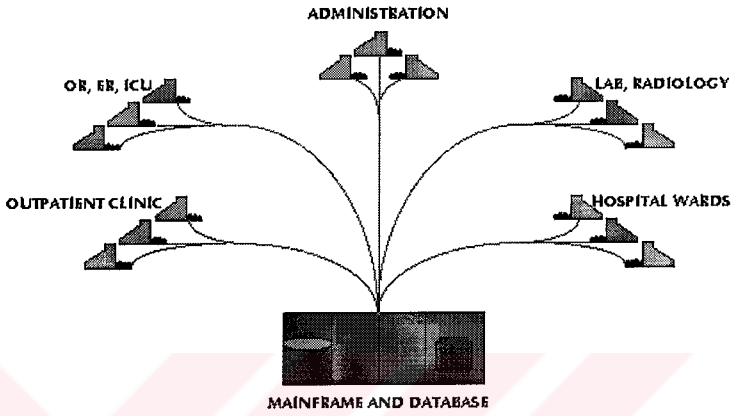
2.2.1 Bilgisayar Teknolojisiyle Veritabanlarının Eşzamanlı Gelişimi

Veritabanlarının ilk kullanılmaya başlandığı dönemlerde veriler ve bu verileri kullanan yazılımlar ana bilgisayar (mainframe) denilen bilgisayarlar aracılığıyla yönetilirdi. Son kullanıcılar ise akılsız (dummy) terminal adı verilen işlem yeteneğinden yoksun terminallerden uygulamalara ulaşırlardı (Şekil 2.1.). Bu tür sistemler en basit anlamıyla "open" ile açılan bir dosyaya bilgilerin satır satır yazılmasından ibarettir. Girilen bilgiler bir dosyaya yazılır sonra okumak için dosyanın en başına dönülüp tekrar okuma yapılır. Bunun bir adım ötesi bunların direkt erişimle binary olarak açılmasıdır. Bu şekilde doğrudan kayıtlara erişilebilmektedir[2],[3].

Reduced Instruction Source Code (RISC) işlemcilerin çıkmasıyla daha önceden kullanılan platformun donanıma bağımlı kalma zorunluluğu ortadan kalkmış açık sistemler yaygınlaşmaya başlamıştır. Bu süreçte dağıtık modüler sistemler ortaya çıkmıştır[4].

Aynı dönemde veritabanlarında kayıtların (record), alanlara (field) bölünmesi yaklaşımı ortaya çıkmıştır[4].

- Bunun ardından kayıtların silinebilmesi ve güncellenebilmesi,
- Kayıtların sıralanarak istenen kayda hızlıca erişim,
- Kayıt boyu büyüdüğünden sıralama yapılacak alanın ayrı bir dosyada tutularak kayıt numarasına referans verilmesi (basit indeks),
- Birden fazla alana göre indeksleme gibi veritabanı olanakları, geliştirilen veritabanı sistemlerine eklenmeye devam etmiştir.

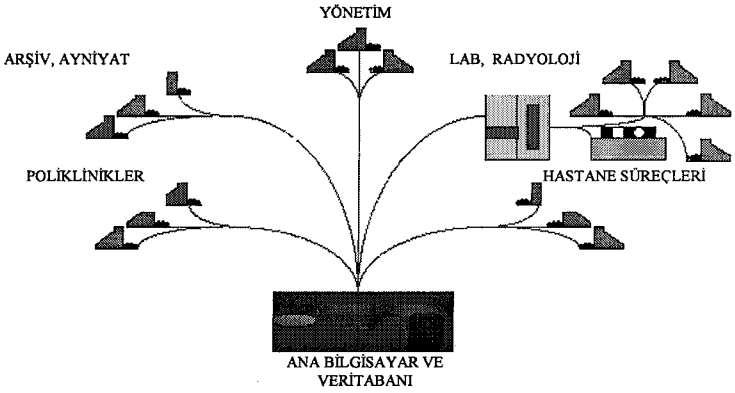


Şekil 2.1. Merkezi veritabanını kullanan aptal terminaller.

Symetrical Multiple Processing (SMP) denilen teknolojik gelişmeyle bilgisayarların birden çok RISC işlemciyi bulundurabilmesi ve bu işlemcilerin paralel işletimle (parallel processing) çalışabilmesi sağlanmıştır (Şekil 2.2.). Bu gelişmeleri takiben veritabanlarındaki gelişmelerde hızla devam etmiştir.

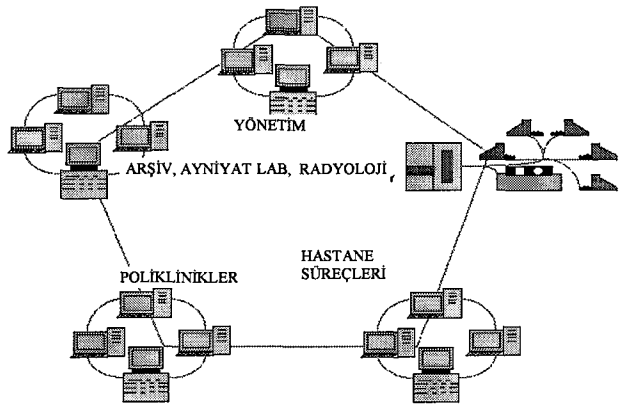
- İndekslerin güçlendirilerek daha hızlı dijital arama (binary search) yapılması.
- Tablo ve kayıt yapılarının her defasında düzeltilmemesi için her türlü kayıt yapısının programla girilip alanların (field) kullanıcı tarafından belirlenmesi,
- Bu tabloların üzerinde indekslerin oluşturulabilmesi,
- Birden fazla indeks desteği sağlanabilmesi.

Ağ teknolojilerinin gelişmesiyle istemci/sunucu (client/server) mimarisinin bu teknolojik gelişmelere eklenmesinin ardından veri, işlem, ve yordam (Remote Processing Call-RPC) paylaşımı gündeme gelmiş (Şekil 2.3.). Yukarıdaki gelişmelerin veritabanlarındaki gelişmelere yansımaları aşağıdaki gibi devam etmiştir.



Şekil 2.2. Paralel işletimle çalışabilen ortak veritabanı kullanan veritabanı

- İş bölümlerinin (transaction) oluşturulabilmesi yani bir çok iş yaptıktan sonra oluşan hatadan dolayı son girilen bilgilerin tamamıyla eski haline getirilebilmesi
- Tabloların birer obje haline getirilerek tablo üzerinde ileriye ve geriye doğru kayıtlar üzerinde hareket yeteneği,
- Birden fazla tablo desteği,
- Kilit (lock) mekanizmaları,
- Birden fazla kullanıcının aynı tabloya ulaşabilmesi,
- İstemci/sunucu mimarisine uygun olarak veri tabanının bir makinede çalışarak diğerlerine hizmet vermesi,
- Birbirinden bağımsız tablolar arasında ilişkilerin kurulabilmesi (referansı verilen kaydın diğer tabloda olması gerekliliği) ve buna dair mekanizmaların gelişimi,
- Daha hızlı verilere erişim için sıkıştırma algoritmaları SQL desteği, yani seç (select), araya gir (insert), güncelle (update), sil (delete) komutlarının birer string olarak programa gönderilerek bunun program tarafından çözümlenebilmesi ve tablolar üzerinden uygun kayıtların çekilebilmesi veya üzerinde değişiklik yapılmasının sağlanması,
- Tablolar arasında ilişki (relation) kurulması yani iki tablo arasında bir birine referans olarak verilen alanlar kullanılarak bir tablodaki kayıtlarla diğer tablodaki kayıtların bir araya getirilebilmesi. Örneğin: Hasta ve test tablosunda hastanın yattığı klinik alanı ile test tablosundaki istem nosu kullanılarak tabloların birleştirilmesi ve hasta kayıtları ile birlikte hastanın test kaydının da getirilebilmesi (inner join).
- Bağımsız ilişkiler kurulması. Örneğin: hasta ve test tablolarında hastanın test istem kaydı olmasa da o hastaya ait kaydın getirilebilmesi (left join, right join, outer join).
- Kayıtlı yordam (stored procedure) veya PL/SQL denen veri tabanının kendi üzerinde çalışan program parçalarına imkan vermesi. Böylece daha hızlı işlem yapabilme yeteneği kazanması[5].



Şekil 2.3. Yordam paylaşımı yapabilen dağıtık veritabanı

2.2.2. Veritabanlarının Sağladığı Avantajlar

Tam işlevli bir VTYS (Full-Function DBMS) bir çok avantaja sahiptir. Bunlar:

- Minimum Veri Tekrarı (Minimum Data Redundancy): Tam fonksiyonlu VTYS'lerinde farklı tablolarda tekrarlı verilerin içerilmesi durumunda bu tablolar arasında mantıksal bütünlük oluşturularak veritabanı yinelemelerden kurtarılır. Veriye erişimi hızlandırmak ve etkinleştirmek üzere yinelemeler yapılsa da veritabanında denetim altında tutulur. Sistemin daha hızlı çalışabilmesi için farklı makinelerde veriler tutulabilir[2],[5].
- Verinin Tutarlılığı (Data Consistency):VT'da yapılan değişiklik tüm veri tabanı için geçerlidir. Veri tutarlılığını sağlamak için :
  - Belli bir andaki işlemlerin tamamı (transaction) son bulmadan veriler üzerinde değişiklik yapılmamasını sağlar.
  - Aynı veri üzerinde okuma (sorgu) gerçekleştirenler yazma (günleme, ekleme, silme) yapanları beklemezler. Sorgu sonucunu veritabanına ulaştıkları haliyle elde ederler.
  - Aynı veri üzerinde yazma (günleme, ekleme, silme) yapanlar okuma (sorgulama) yapanları beklemezler.
  - Aynı veri üzerinde yazma yapanlar aynı veri üzerinde yine yazma yapanların işlemlerinin sonuçlanmasını beklerler[2],[5].
- Verinin Bütünlüğü (Integrity of Data): VT'ındaki veriler sahip oldukları anahtarlarve ilişkilerle mantıksal olarak birbirleriyle bir bağ oluştururlar. Böylelikle veriler arasındaki ilişkiler ve kısıtlar kullanılarak veri bütünlüğü sağlanır. Örneğin hastanın hastaneye girişinden çıkışına kadar hangi servislere uğramış, hangi tetkikler istenmiş, ne sonuçlar elde edilmiş, ne kadar ücret tahakkuk etmiş, bunların hepsi sorgulanabilir[2],[5].

Oracle VTYS ve diğer birçok VTYS'ne ait temel kısıtlar: "NOT NULL (mutlaka bir değer barındırmalı)", "UNIQUE (unique olan bir veri alanı tanımlı

olduğu nesne içinde sadece bir kayda karşılık gelir)", "PRIMARY KEY (tanımlı olduğu nesnenin kayıtları için tekliği tanımlar)", "FOREIGN KEY (veriler arasında ilişkilerin tanımlanmasını sağlayan kısıtlamalardandır)", "CHECK (kadın-erkek gibi iki durumlu seçimlerin veritabanına işlenmesi için kullanılır)"

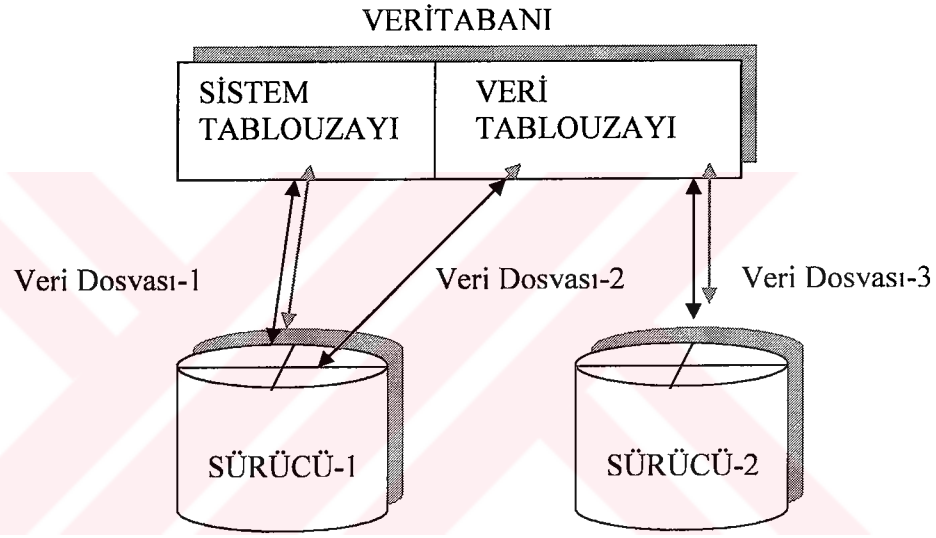
- Veri Paylaşımı (Sharing of Data): Her kullanıcının erişim hakkına bağlı olarak her türlü veri eşanlı olarak erişime açıktır. Her kullanıcı kendisi için gerekli olan verilere ara yüzler (view) aracılığıyla eşzamanlı erişebilir[2],[5].
- Uygulama Geliştirme Kolaylığı (Ease of Application Development): Dördüncü kuşak programlama dilleri son derece kolay program geliştirme imkanı sağlanmaktadır. Bu da düşük maliyetli ve kısa sürede geliştirilebilir programların tasarımı ile verimliliği artırır[2],[5].
- Güvenlik (Security): VTYS'lerinin güvenlik mekanizmalarının sahip olduğu işlemler:
  - Yetkisiz kullanıcıların veritabanına erişiminin engellenmesi
  - VT'ında bulunan nesnelere yetkisi olmayanların erişiminin engellenmesi
  - Disk kullanımının denetimi
  - Sistem kaynaklarının (Merkezi İşlem Birimi-CPU) kullanımının denetimi
  - Kullanıcıların işlemlerinin izlenmesi (log dosyası tutulması)
  - Kullanıcıların yarattığı VT nesnelerinin boyutlarının sınırlandırılması ve denetimi
  - Her VT nesnesi için gerekli erişimlerin tanımlanması
  - Kullanıcılara rollerin atanması suretiyle yapabilecekleri işlemlerin sınırlandırılması
- Çöküntülere Karşı Korunma ve Yedek Alma (Recovery and Backup): Sistemin çökmesi durumunda veritabanı etkilenmişse kaldığı yerden devam etmesi sağlanmalı, kesilen işlemler geri alınamıyorsa bunlar işlenmeden geri alınabilmelidir (rollback). VT'ının veri kaybına uğraması durumunda disk sınırlılığından dolayı alınan yedeklerin VT sistemine tekrar yüklenebilmesi gerekir[2],[5].
- Veri Erişiminde Kolaylık (Ease of Data Access): Kullanıcılar tarafından bir çok farklı veriye ihtiyaç duyulabileceğinden verileri çekebilecek bir yapısal dille Vt'ının uyumlu olması gerekir[2],[5].
- Veri Bağımsızlığı (Data Independence): VT'ından çekilen verilerle bu verilerin tanımlarının birbirinden ayrılmasıdır. Veri bağımsızlığı olan bir VT'ında veri tanımının değişimi veriyi kullanan uygulamanın değişimini gerektirmez[2],[5].

2.2.3. ORACLE Veri Tabanı Yönetim Sistemi

Oracle VTYS işletim sistemi açısından bakıldığında fiziksel (physical) ve mantıksal (logical) olmak üzere iki görüntüsü vardır. İşletim sistemine kayıtlı olan Veri Dosyası (Datafile), Kontrol Dosyası (Control File) ve Kütük Dosyası (Log File) VTYS'inin fiziksel bölümünü oluşturur. Mantıksal bölüm içerisinde ise Tablespace,

Tablo (Table), Görüntü (View), Sıra (Sequence), Eşanlam (Synonym) ve Kullanıcı (User) birimleri bulunur[5]. Fiziksel bölüm işletim sisteminin kontrolünde olmasına rağmen mantıksal bölüme ulaşmak için Oracle'a bağlanıp SQL komutları işletilmelidir. SQL ile sorgulama yapılmadığı durumda Oracle'ın sadece fiziksel bölümüne müdahale edilebilir[6].

Oracle Veritabanında kullanılan her nesnenin bir sahibi (owner) vardır. Bir kullanıcının sahip olduğu nesneler bir veya daha fazla tablespace denilen tablo alanları içinde yer alır. Her nesne yalnızca bir tablespace içerisinde -mantıksal olarak- bulunur. Her tablespace de kendine ait nesnelerini tutmak için işletim sisteminde bir veya daha fazla veri dosyasına (datafile) sahip olabilir[6],[7].



Şekil 2.4. Oracle VT, Tablouzayı ve Veri Dosyası

Özetle; veritabanındaki her bir nesne bir kullanıcıyla ilişkilendirilmiştir. Bu nesneler mantıksal olarak o kullanıcının sahip olduğu tablespacelerin herhangi birinin içerisinde, fiziksel olarak o kullanıcının sahip olduğu herhangi bir veri dosyasında bulunur. Ancak bu veri dosyasının içeriği nesneyi bulmak üzere işletim sistemiyle incelenemez (Şekil 2.4.). Bu nesnenin sahibi ve mantıksal yeri Veri İşleme Dili (Data Manipulation Language-DML) komutlarıyla bulunabilir[5].

2.2.4. ORACLE Veri Tabanı Nesneleri

Oracle'da kullanılan nesnelerin tamamı ya veritabanının yöneticisi ya da uygulama programlarını geliştiren programcılar aracılığıyla yaratılır[2],[5].

2.2.4.1. Indeks

Tablolarda aranan kaydı daha çabuk bulmayı sağlayan nesnelerdir. Veritabanı üzerinde en çok sorgulanan alanlara yönelik olarak tanımlanan indeksler fiziksel olarak da veritabanında yer tutarlar. Aksi belirtilmediği sürece, yani uygulama geliştiriciler tarafından herhangi bir tanımlama yapılmadığı durumda indeks optimizasyonunu VTYS kendisi gerçekleştirir. SQL kullanılarak ipuçlarıyla (hint) optimizasyon sağlanabilir. Tablolar üzerindeki her türlü yazma işleminden indeksler etkilenir[2],[5],[8].

2.2.4.2. Cluster ve Hash Cluster

Clusterlar aynı anda sorgulanan ve birbiriyle sıkı ilişkisi olan birden fazla tabloyu bir araya getiren nesnelerdir. Disk hareketini azaltarak fiziksel hız kaybını en aza indirmeyi sağlayan nesnelerdir. Örneğin laboratuardan yapılan sorgulama işleminde istem no, istem talebinde bulunan doktor, istemi gerçekleştiren birim, istem birimi gibi alanları içeren tabloyla istem sonucunda oluşacak verilerin kaydedileceği tablo alanları birbirisiyle sıkı ilişki içerisindedir. Bu alanları barındıran tablolar aynı cluster nesnesi içerisinde tanımlanırsa istemlere ve hizmetlere daha hızlı erişim sağlanmış olur[2],[5],[8].

Hash clusterlar ise cluster üzerinde tanımlanan indekslerin bir matematiksel formülle (hash işlevi) verilere erişimi sağlayan indeks nesneleridir. Aynı hash anahtarını taşıyan veri grupları bir arada tutulursa erişim hızı artar.

2.2.4.3. Rol (Role)

Oracle veritabanında veritabanı nesneleri kullanıcılara aittir. Bir kullanıcının kendisine ait olmayan bir nesneyi kullanabilmesi için kullanmak istediği nesnenin sahibi tarafından kendisine yetki (grant) verilmelidir[2],[5].

Rol veritabanındaki hakların toplanmış halidir. Geniş veritabanları içlerinde birçok kullanıcı ve hak bulundurdukları için yetkilerin kontrolü güçtür. Yetkilerin kontrolünün kolaylaştırılması için benzer tipteki yetkiler tek tek kullanıcılara verileceğine bir defalığına bir role verilir. Bu rolün sahip olduğu yetkileri kullanıcılara vermek ise bir komutla mümkün olabilmektedir.

Sistem hakları (system privilages) veritabanının kullanımıyla ilgili kullanıcılara sistem tarafından tanınan rolleri içerir. Bu roller sistem kullanıcıları tarafından diğer kullanıcılara hak olarak verilir. Örneğin:

CREATE SESSION: Veritabanına bağlanmahakkını içeren roldür. Bir kullanıcı yaratıldığında sahip olacağı en önemli rollerden birisidir. Bu role sahip olmayan kullanıcı veritabanına bağlanamaz.

CREATE TABLE: Tablo yaratabilmek için gerekli olan yetkiye sahip olan roldür. Tablo yaratabilmek için kullanıcının bu rolün hakkına sahip olmalıdır. Ayrıca bu hakkın sahibinin herhangi bir tablespace içerisinde boş yere sahip olması gerekir.

CREATE VİEW: Görüntü yaratmak için gerekli olan haktır.

CREATE USER: Kullanıcı yaratma hakkıdır.

ALTER TABLESPACE: Tablespacelerin yapısını değiştirme hakkıdır. Örneğin tablespace içerisine yeni bir veri dosyası eklemek gibi.

DROP ANY INDEX: Veritabanındaki herhangi bir indeksi düşürme (silme-DROP) yetkisidir[5].

2.2.4.4. Eşanlam (Synonym)

Veritabanındaki nesnelerin isimlerinin anlaşılır hale getirilmesine olanak tanırlar. Herhangi bir veri içermezler, nesnelere referans gösterirler. Eşanlamlar ile veritabanının anlamlı bir şekilde isimlendirilmesi sağlanır, bu da veritabanının okunabilirliğini artırır. SQL kullanımında eşanlamların kullanımı büyük kolaylıklar sağlar. İki çeşit eşanlam vardır. Bunlar:

Özel Eşanlam (PRIVATE SYNONYM): Kullanıcıların kendi nesnelerine kolay erişimini sağlamaya yönelik olarak oluşturulan eşanlamlardır.

Genel Eşanlam (PUBLIC SYNONYM): Bir nesnenin bütün kullanıcılara hak olarak verilmesini sağlayan eşanlamdır. Bir kullanıcının bir nesnesi için genel eşanlam tanımlanırsa diğer kullanıcılar hak verilmeksizin o nesneyi kullanabilirler[5].

2.2.4.5. Sıra (Sequence)

Veritabanı işlemleri sırasında kullanıcılar birbirini takip eden sıralı numaralara ihtiyaç duyarlar. VTYS, birçok kullanıcıdan aynı anda veritabanındaki verilerden sıralı işlemler yapma isteği geldiğinde her istemde farklı sıralı numaralar üretip çakışmaya meydan vermez. Tablolardaki kayıtlar için sıra numarasının ilerlemesi veritabanı tarafından yapılmak istendiğinde kullanılan nesnelere sıra denmektedir. Sıralar özellikle tekil (unique) olması gereken alanlar için kullanılırlar[5].

2.2.4.6. Kaydedilmiş Fonksiyon ve Prosedür (Stored Function and Procedure)

Yordamlar, paketler, tetikleyiciler gibi program parçacıkları veritabanına kaydedilir ve kullanıcıların kullanımına sunulur. Bunların kullanımı veritabanına yetkisiz erişimi önlemek için de kullanılır. Bu program parçaları Oracle PL/SQL

programlama diliyle geliştirilir. Bu program parçacıkları sürekli kullanılması gereken fonksiyonları veya prosedürleri çalıştırmak amacıyla veritabanına kaydedilir. Kaydedilmiş fonksiyonların ve prosedürlerin yaratılması şöyledir[5]:

CREATE veya REPLACE FUNCTION (PROCEDURE) fonksiyon (yordam)_ismi (parametre, parametre. ...) IS

Tanımlamalar

BEGIN

Komutlar

EXCEPTION

Aykırı durumlar

END;

Buradaki ifadelerde;

- Fonksiyon_ismi tanımlanan fonksiyona verilen ad.
- Yordam_ismi tanımlanan yordama verilen isimdir
- Parametre fonksiyona dışarıdan çağrılırken gönderilecek değerler.
- BEGIN bloğun başlangıcın belirten rezerve isimdir.
- EXCEPTION aykırı durumların başlangıcını belirten rezerve isimdir.
- Aykırı durumlar hata oluşursa neler yapılacağının tanımlandığı bölümdür.
- END bloğun sonunu belirten rezerve isimdir.

Kaydedilmiş fonksiyonların veya prosedürlerin çağrılması:

Fonksiyon_ismi (parametre, parametre, ...);

Yordam_ismi (parametre, parametre, ...);

2.2.4.7. Geri Alma Parçası (Rollback Segment)

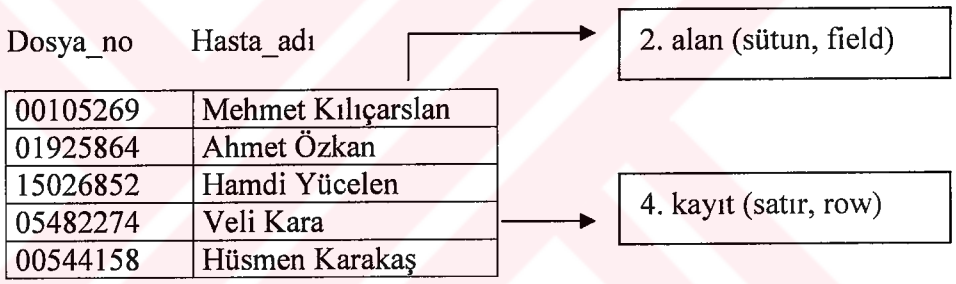
Oracle'da nesnelerin ilk yaratılışlarında her bir nesne için bir yer (segment) ayrılır. Nesne için ayrılan bu yerin yetersiz kalması durumunda bir genişleme (extent) alır. Nesnenin alabileceği genişleme miktarı nesnenin oluşturulmasında tanımlanan ona ait yerin alabileceği en büyük genişleme (max extent) sayısı ile sınırlıdır. Genişleme

sayısı max extent sayısına geldikten sonra bir daha genişleme alamaz. Böylesi durumlarda bu parçayı kullanan nesne yedeklenir ve düşürülür (DROP edilir), yeniden yaratılarak yedekten geri çağrılır[5].

Oracle veritabanı güvenliği için yapılan her Veri İşleme Dili işlemini (select, insert, update, delete, vb.) yedekler. Bu yedeklerin veritabanı üzerinde konuldukları yere geri alma parçası (rollback segment) adı verilir. Kullanıcı COMMIT komutuyla bu işlemleri onaylamış olur ve böylelikle geri alma parçasındaki yedekler silinebilir hale gelmiş olur. Kullanıcı ROLLBACK komutunu verirse yapılan bu işlemleri iptal ederek yapılmış olan değişiklikler geri alma parçasından geri getirmiş olur.

2.2.4.8. Tablo (Table)

Bütün ilişkisel veritabanlarında tablolar nesneleri (entity) temsil eder. Bu tablolar tüm kullanıcıların yetkilerine göre ilişkisel veritabanı modeline uygun verileri tutar. Tablolardaki veriler tablonun satır ve sütunlarıyla tanımlanırlar. Satırlar tabloya ait kayıtları sütunlar ise kayıtların özelliklerini tutarlar[2],[5].



Şekil 2.5. Bir veritabanı tablosunun basit yapısı

2.2.4.9. Tablespace

Veritabanı kullanıcılarının sahip olduğu nesnelerin veritabanında mantıksal olarak tutulduğu yere tablespace adı verilir. Veritabanı nesneleri fiziksel olarak veri dosyalarında (data file) tutulurlar. Tablespaceler veritabanının etkinliğini artıran önemli bir veritabanı nesnesidir[2],[5].

2.2.4.10. Kullanıcı (User)

Veritabanındaki her nesne bir kullanıcıya aittir. Kullanıcılar nesneleri yaratma, kullanma ve silme yetkisine sahiptirler. Oracle veritabanında tanımlanmış olan üç tane kullanıcı vardır[6],[8].

Bunlardan SYS ve SYSTEM sistem kullanıcıları, SCOTT ise deneme tablolarının sahibi olan kullanıcıdır. SYS kullanıcısı aynı zamanda veri sözlüğünün (data dictionary) sahibidir. Veritabanının en önemli kullanıcısı SYS kullanıcısıdır. İkinci önemli kullanıcı olan SYSTEM, SYS'nin sahibi olduğu veri sözlüğünün hepsini kullanma hakkına sahiptir. Her iki kullanıcı da tablespace, user, role vb. gibi önemli veritabanı nesnelerini yaratma hakkına sahiptir. SQL aracılığıyla kullanıcılar SCOTT kullanıcısının nesnelerini kullanabilirler[6],[8].

2.2.4.11. Görüntü (View)

Bir veya birden çok tablonun verilerinin belirlendiği şekilde görülebilmesi için kullanılan nesnelerdir. Görüntü, tablodaki kayıtları referans alarak o anlık gösterimi kullanıcıya sağlar. Görüntüyü oluşturan daha önceden tanımlanmış olan bir SQL cümleciğidir[5],[6].

2.3. Uygulama Ara Yüz Yazılımları

Nesne Tabanlı programlama bilgisayar yazılımı geliştirmede yeni ve güçlü bir model sunmaktadır. Bu yaklaşım yeni programların geliştirilmesinde ve gerekliyse yazılımın bakım, tekrar kullanım, program değişiklikleri yapılması gibi konularda hız kazancı sağlamaktadır[9].

Bununla birlikte, nesne tabanlı programlama programcıların bazı temel düşünme ve tasarım yapılarında değişiklikleri zorunlu kılmıştır. Bu tasarım yaklaşımı değişikliği bu teknolojiye uygun şekilde geçişi sağlamaktadır. Clarion programlama dili nesne tabanlı programlamaya basit bir şekilde geçişi sağlayabilmektedir[9].

2.3.1. Clarion Programlama Dili

Clarion programlama dili program geliştirmede, geliştirilen programların okunmasında sağladığı kolaylık ve gereksiz yazım kurallarından arındırılmasıyla son derece kullanışlı bir program geliştirme aracıdır. Clarion programlama dili programlama araçlarıyla genişletilmiş bir veritabanı geliştirme sistemi olması özelliğinde dolayı veritabanı geliştirme araçlarıyla genişletilmiş olan Visual Basic, Delphi, Power Builder gibi benzeri program geliştirme araçlarından ayrılır[2],[3].

Topspeed firması tarafından geliştirilen Clarion programlama dili ile DOS ortamında alt seviyeli programlar geliştirilebildiği gibi Windows ortamında (3.x, 95, 98, NT, 2000) ileri düzeyde veritabanı uygulamaları geliştirmeye olanak tanımaktadır. 1996 yılında piyasaya sürülen Clarion 2.0 sürümünden sonraki sürümler Windows ortamında OLE (Object Linking and Embedding), DLL (Dynamic Link Library), DDE (Direct Data Exchange), RAD (Rapid Application Development), ODBC (Open DataBase Connectivity), SQL, ActiveX ve Client/Server (İstemci/Sunucu) mimarisi desteği vermektedir[2].

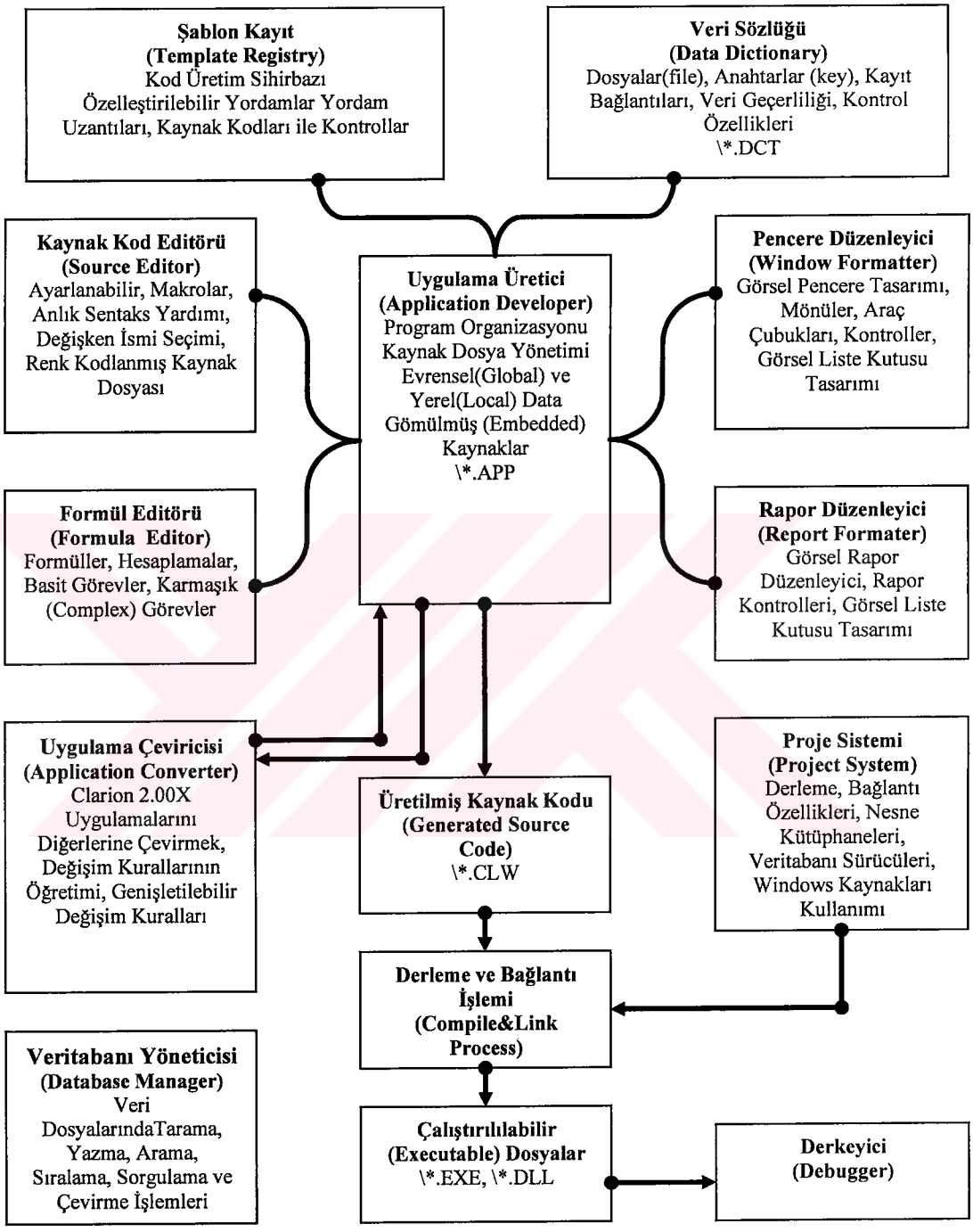
2.3.2. Clarion Programlama Dili Program Geliştirme Temelleri

Clarion uygulama geliştirme aracı programcıların yeni uygulama programları yazmaları için geliştirilen bir programdır. Uygulama geliştirme aracı windows programlarının en iyi şekilde çalışması için gerekli olan kaynak kodu, çalıştırılabilir program dosyaları, kaynak dosyaları vb. gibi bütün dosyaları ve program yazma, derleme, bağlantılar yapma vb. gibi her türlü işlemi kontrol eder ve yönetir[9]. Clarion'un uygulama geliştirme aracının genel işleyiş şablonu Şekil 2.6'da gösterilmiştir.

Program geliştirme aracı oldukça düzenlemeye olanak tanır. Program geliştirme aracı çalışılan amaca göre düzenlenebilmektedir. Böylelikle en yüksek oranda verimlilik sağlanabilmektedir. Uygulama (application) ve proje (project) terimlerinin her ikisi de Clarion Uygulama Geliştirme Aracı tarafından kullanılan iki önemli terimdir. Zaman zaman ikisi de aynı amaca yönelik olarak kullanılsa da Proje Sisteminde uygulama geliştirme aracının derleme ve bağlantı yaratma aşamasında ikisinin birbirinden ayrılması gerekir[9].

Bir clarion projesi, kaynak kodlardan oluşmuş bir grup program dosyası ve çalıştırılabilir bir dosya oluşturmak için gerekli olan komutları içerir. Proje, şablonlarla oluşturulmuş programlardan çok manuel olarak geliştirilmiş programlardır. Uygulamalarda ise program daha çok şablonlarla oluşturulmuştur. Ayrıca derleme ve bağlantıların oluşturulması uygulama dosyasında (.APP) kayıtlıdır[9].

Veri sözlüğü geliştirilen uygulamanın dosyalarını, alanlarını, anahtarlarını, ve dosya ilişkilerinin tanımlandığı merkezi birimdir. Veri sözlüğü, dataların disk üzerinde nereye ve nasıl kaydedildiği, aynı zamanda verinin son kullanıcının raporlarında ve ekranında nasıl görüntüleneceği gibi kaydedilmiş bilgileri içerir. Sözlük dosyaları (.DCT); dosya isimlerini, dosya tanımlamalarını, dosya anahtarı ve indeks tanımlamalarını, veritabanı bağlantı bilgisini, ilişkisel bütünlüğü (RI=relational integrity), alan veri tiplerini, alan tanımlamalarını, alan balon yardımlarını, alan durum çubuğu mesajlarını, alan geçerlilik kurallarını, alan bilgi girişi görünümleri ve biçimlendirmesini, her alan için tanımlı ekran ve rapor denetimlerini, ve daha birçok programlama nesnesini içerir[9].



Şekil 2.6. Clarion uygulama geliştirme aracının genel işleyiş şablonu

Bütün bu bilgilerin merkezi bir yerde kaydedilmesinin sağladığı fayda, uygulamanın geliştirilmesi ve kurulmasında meydana gelecek çok büyük zaman kaybını ortadan kaldırmaktır. Buna ek olarak bu sözlük yapısı, uygulamanın birbirine bağımlı görünüm oluşturmasını sağlanmış ve son kullanıcının en az bilgiye gereksinim ile programları kullanmasına olanak sağlar. Veri sözlüğünün bu yapısı sayesinde alanlar,dosya bağlantıları, geçerlilik kuralları vs. sadece bir kez tanımlanır ve alanlar uygulama içerisinde referans gösterilerek program geliştirilir. Böylelikle her program geliştirileceğinde aynı alanların tekrar dan tanımlanmasına gerek kalmaz[9].

Bir veri sözlüğü oluşturduktan sonra yeni bir uygulama geliştirmenin birinci adımı bir uygulama dosyası (.APP) yaratmaktır. Uygulama dosyası yordam özelliklerini, veri tanımlamalarını ve programcının tanımladığı başka özellikleri tutar. Uygulama dosyaları kaynak kodu yaratmak için her şeyi bulundurur ve bu kodlardan uygulama programı yaratır[9].

Pencere nesnelerinin -pencerelerin ve kontrollerin- görsel tasarımı için Clarion programlama dilinde pencere biçimlendiricisi (window formatter) kullanılır. Pencere içerisinde tanımlananların Clarion kaynak kodlarını pencere biçimlendiricisi kendisi yaratır. Uygulama yaratıcısı (application generator), oluşturulan kaynak kodları uygulamanın içerisinde uygun yere yerleştirir[9].

Geliştirilen programların tamamında komutları göstermek, girdileri kabul etmek, kullanıcılara verileri veya başka bilgileri sağlamak üzere pencereler kullanılır. Ekrana bir pencere yerleştirmek için;

- pencereyi kontrol edecek olan prosedür seçilip yaratılır,
- uygulama ağacı diyalogundan prosedür ismi üzerinde sağ tuşa basılarak açılan açılır (popup) mönüden pencere seçilir (eğer tanımlanmış pencere yoksa yeni yapı (new structure) diyalogundan pencere türü seçilir),
- pencere boyut ve özellikler açısından düzenlenir,
- istenirse mönü editör ile pencereye mönü yerleştirilir,
- pencereye kontroller -veri tabanı alanlarına veri girmek için giriş kutuları, işlemlerin onaylanması veya reddedilmesi için komut düğmeleri (butonları), metin, karakter dizileri (string) veya kullanıcılara programın kullanımıyla ilgili bilgiler veren ifadeler (prompt) ile pencerenin kullanımını kolaylaştıracak diğer kontroller- yerleştirilir,
- kontrol özellikleri ayarlanır,
- tekrardan yordam özellikleri (procedure properties) diyaloguna dönülür.

Clarion programlama dilinde kullanılan pencereler içerisinde etkileşimli interactive) ve etkileşimsiz (non-interactive) olmak üzere iki çeşit kontrol vardır[9].

Etkileşimli kontroller kullanıcı tarafından üzerine tıklanan veya içerisine bir şeyler yazılan kontrollerdir. Hareket kontrolü (action control) için, düğmeler (button); kullanıcı seçim kontrolleri (user choice control) için, kontrol (check), seçim (option), radio, combo, spin ve list; giriş kontrolü (entry control) için, giriş (entry), ve metin (text); karışık kontroller (hybrid control) için, sayfalar (sheet), sekmeler (tab), ve bölgeler (region) kullanılmaktadır[9].

Etkileşimsiz kontroller ise kullanıcının etkileşimli kontrolleri anlamasını kolaylaştıracak görsel ibarelerdir. Durağan (statik) kontroller -pano (panel), hatırlatma ifadeleri (prompt), grup (group), kutu (box), elips (ellipse), çizgi (line) ve görüntü (image)- herhangi bir hareketi başlatmazlar, ancak kullanıcının diğer kontrolleri anlamasını kolaylaştırırlar[9].

Clarion programlama dili etkileşimli ve etkileşimsiz kontrollerin dışında OLE kontrolleri ve VBX kontrolleri gibi harici kontrollerin (custom control) kullanımına olanak vermektedir[9].

OLE kontrolleri uygulamanın penceresine OLE nesneleri veya .OCX kontrolleri yerleştirir[9].

OLE sunucusunun (server) kullanımı sonucunda programa sunucunun bütün imkanları küçük bir programla aktarılmış olur. Ancak programın son kullanıcılarının OLE sunucusunun kullanım hakkına (lisans) ve donanımlarının sunucunun, gereksinim duyduğu asgari özelliklere sahip olması gerekmektedir. Clarion uygulamaları 486DX işlemcide 4Mb RAM (Random Access Memory) ile etkin bir biçimde çalışabilmesine rağmen eğer uygulama Excel, PowerPoint, SPSS ve/veya Word gibi uygulamaları çağırıyorsa son kullanıcıların bu programların kullanım hakkına sahip olması ve yine bu programları çalıştıracak konfigürasyonda (Pentium, 16 Mb'dan büyük RAM'e sahip) bilgisayarı olması gerekir[9].

OCX kullanımı ise genellikle program içerisine yerleştirilen bir çok kaynak kodu veya program kodu gerektirir. Bununla birlikte çoğu OCX 'ler ek kullanım hakkı ücreti alınmaksızın dağıtılmaktadır ve çoğu OCX'ler OLE sunucusu oluşturmak için gerekenden daha az kaynak istemektedir[9].

Uygulama geliştirme (application generator) ara yüzü OLE kontroller için kısmi işlevsel nitelikli otomatik kaynak kodu oluşturma desteğine sahip değildir. OLE kontrol işlevlerinin kullanılabilmesi için o OLE kontrol nesnesinin kaynak kodların program içerisine yerleştirilmesi (embedding) veya OLE kontrol şablonlarının kullanılması gerekir. Aksi durumda programda sadece OLE nesnesinin imajı görünür[9].

Bir OLE sunucusuna erişmek için OLE kontroller kullanılırken en azından sunucuyu çalışır hale getirmek (activation) ve durdurmak (deactivation) için programa bazı kodların yerleştirilmesi gerekir (OCX'ler otomatik olarak çalıştırılabilirler). Bu

özellik, kullanıcının OLE sunucusuna ait bütün işlevlere erişebilmesini sağlamaktadır. Sunucunun tam olarak nasıl davranacağı ve hangi sunucu işlevlerinin kullanıcılar tarafından kullanılması sağlanacağı gibi özellikler ek kodlarla programa yerleştirilir. Detaylı OLE sunucu kullanımını tanımlayan OLE sunucu dokümanlarından yararlanılmalıdır[9].

Sunucu bir düğmeye tıklama, bir mönü seçimi, bir listeden seçim yapma, fare tıklaması, vb. kontrollerle çalıştırılabilir (activation). Sunucuya çalıştırmak için "PROP:DoVerb" terim ifadesi kullanılır. Örneğin sunucuyu tanımlanmış durumuyla (default mod) açmak için;

```
?OleControl{PROP:DoVerb} = 0
```

Yeni pencere açma durumuyla (open mod) açmak içinse;

```
?Ole{PROP:DoVerb} = -2    !Ole sunucu kendi ayrık penceresinde açılır
```

komutları kullanılır. Açılmış bir sunucuyu kapatmak içinse "PROP:Deactivated" terim ifadesi kullanılır. Mönü seçeneklerine OLE sunucunun durdurulması (deactivation) ile ilgili bir mönü seçeneği yerleştirilerek bununla ilişkili sunucuyu kapatma kodları kontrol olayları tanımlama (control event handling) kod alanının kabul (accepted) kısmında tanımlanarak sunucunun kapatılması sağlanır. Sunucuyu kapatmak için aşağıdaki gibi bir tanımlama yapılır:

```
?Ole{PROP:Deactivate}              !Ole sunucusunu kapatır
```

Bir OCX tanımlaması için genellikle daha çok kodlama gerekir. Gerekli kodlar OCX'e özel tanımlıdır ve genellikle OCX'in yaptığı işlemleri tanımlamaya yöneliktir[9].

OLE Kontrol Mönüleri: Program pencerelerinde olduğu gibi OLE kontrolleri de mönüler bulundurabilir. OLE kontrolleri mönüsünü tanımlamak için pencere biçimlendiricisinde OLE kontrolü seçilir daha sonra mönüden yeni mönü seçilir. Bundan sonra pencere için mönülerin tanımlandığı gibi mönü tanımlamaları yapılır[9].

OLE kontrolleri mönüsü pencere mönüsüyle birleştirilir. OLE sunucusu çalışma zamanında (runtime) etkinleştirilmişse OLE sunucu mönüleriyle pencere menüleri birleştirilir.

Kayıt Dosyası ile Bağlanmış (Linked) veya Gömülmüş (Embedded) Dökümanın Birleştirilmesi

Kayıt dosyasının tanımlanması (OPEN('Filename\!objectName)) ile bağlanmış veya gömülmüş dokümanın tanımlanması (LINK('Filename)) birbirine benzer şekillerde

yapılır. Her iki durumda da OLE nesnesi harici bir dosyaya bağlantı yapar. Ancak bunlar arasında bazı önemli farklılıklar vardır:

- OLE sunucusu bağlanmış veya gömülmüş doküman dosyalarını uygulamadan bağımsız bir şekilde kullanıp değişiklikler yapabilmesine rağmen kayıt (storage) dosyasında bunu gerçekleştiremez.
- Bir kayıt dosyası çoklu nesneler içerebilir. Örneğin; iki çalışma sayfası veya bir çalışma sayfası ile bir Word dokümanı.
- OLE taşıyıcısının (container) özellikleri kayıt dosyası içine kaydedilip buradan değiştirilir, fakat bağlanmış veya gömülmüş dokümanlarda böyle bir durum yoktur.

2.3.2.1. Kontrol Tipi Olarak OLE

OLE radio düğmesi seçildiğinde, nesne tip listesi (object type list) kayıtlı OLE nesnelerini seçer. OLE yapısı yaratma (CREATE) veya açma (OPEN) özelliklerine göre alınır (Şekil 2.7.).

2.3.2.2. Nesne Tipi (Object Type)

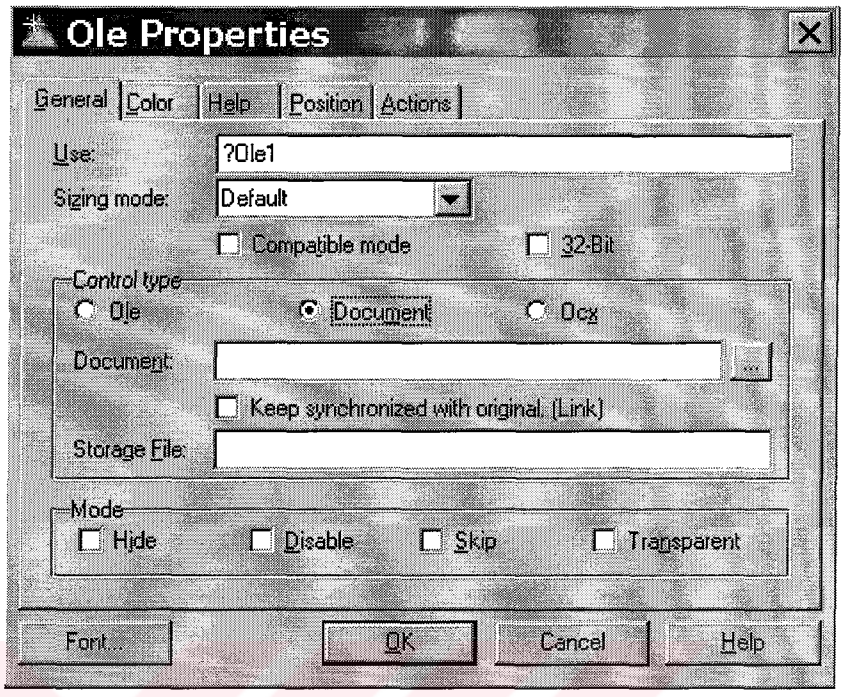
Yaratmak veya açmak üzere kayıtlı Excel çalışma sayfası, Word dokümanı, PowerPoint Slaytı, SPSS RTF Nesnesi vb. gibi OLE nesnelerinden biri seçilir. Bir OLE sunucusu seçildiğinde pencere düzenleyicisi otomatik olarak bu sunucuyu yükler. Bu pencere tasarım işlemleri sırasında zaman kaybına sebep olabilir. Bunun için pencerenin tasarımı ve çiziminin önce, OLE kontrolünün tanımlanmasının sonra yapılması veya sunucunun, tasarım aşaması yerine özellik sözdizimlerinin (syntax) çalışma zamanında (runtime), belirlenmesi gerekir[9].

Kayıt Dosyası (Storage File) : OLE bileşik kayıt dosyasının (.OLR) adını ve içerisinde onu açacak olan nesneyi belirler. Dosya ismi ile nesne ismi birbirinden ters bölüm çizgisi (backslash) ve ünlem işaretiyle (exclamation point) ayrılır:

Filename\!Objectname.

Eğer bu alan boş bırakılırsa oluşturulan uygulama belirlenen tipte yeni bir OLE nesnesi (Excel çalışma sayfası, Word dokümanı, PowerPoint Slaytı, SPSS RTF Nesnesi vb.) yaratır.

OLE sunucusu (SPSS, Excel, PowerPoint, Word, vb.) bileşik kayıt dosyasına (compound storage file) erişebilir ve üzerinde değişiklikler yapabilir ama bunu sadece geliştirilen uygulama üzerinden gerçekleştirebilir[9].



Şekil 2.7. OLE Özelliklerinin tanımlandığı pencere

2.3.2.3. Kontrol Tipi Olarak "Document"

OLE özellikleri penceresinde denetim çeşidi olarak "document" seçimliği seçildiğinde nesne tipleri listesi değişir ve "keep synchronized with original" işaret kutusu (check box) görünür. OLE yapısı belge ve bağlantı özelliklerini çağırır[9].

Belge (Document): Belge dosyasının tam yerinin yazılması veya elips (...) düğmesiyle seçilmesi gerekir. Belge özel OLE sunucusuyla birleşmiş bir dosyadır. Uygulama programı sunucunun çalışma anında bu dosyayı çağırabilmektedir. (örn. Testler.sav dosyası SPSS ile birleşmiştir.)

Eğer dosya isminin tanımlandığı yerde dosya yoksa uygulama mevcut dizine bakar. Belge dosyasının son kullanıcının bilgisayarında programda tanımlanmış olduğu yerde bulunması gerekir[9].

"Keep synchronized with original (Link)" seçim kutusu işaretlendiğinde OLE yapısına bağlantı (link) özelliği eklenmiş olur. Bunun anlamı asıl metinde yapılan her türlü değişikliğin doğrudan uygulama programını da etkilemesidir. İşaretin kaldırılması ise belge özelliğinin OLE yapısına eklenmesini sağlar ki, bu da asıl metindeki değişikliklerin uygulama programını etkilememesi anlamına gelir[9].

2.3.2.4. Kontrol Tipi Olarak "OCX"

OLE özellikleri penceresinde denetim çeşidi olarak "OCX" seçimliği seçildiğinde nesne tipleri listesi kayıtlı OCX nesnelerini de içerir. OLE tanımlaması oluşturma (create) ve açma (open) özelliklerini etkinleştirir. Nesne tipi açılır mönüsünden (combo box) yaratmak veya açmak üzere OCX nesneleri seçilebilir[9].

2.3.2.5. OCX'lerle OLE Kontrolü

Kullanılmadan önce OCX'lerin Windows işletim sistemine kaydedilmesi gerekir. OCX'lerin kayıt edilmesi için OCX yardım belgeleri kullanılabilir. Bazı OCX kurulum programları programın kurulum aşamasında OCX nesnelerini otomatik olarak kaydeder. Bununla birlikte çoğu programın OCX'leri kayıt için lisans sözleşmesine ihtiyaç duyar[9].

Genellikle geliştirilen programın OCX'lerle etkileşimi diğer program unsurlarının OCX'lerin işlevlerine yönelik olarak özellik atama ve geri çağırma işlevlerine (callback function) yöneliktir. Bu yüzden kullanılacak olan OCX'in özelliklerinin (property), kullanım yerlerinin (event) ve işlemlerinin (operation) iyi bilinmesi gerekir. Bu bilgilere ulaşmak için OCX kullanım belgeleri kullanılır[9].

Clarion çalışma zamanı kütüphanesi (runtime library) ihtiyaç duyduğunda geri çağırma işlevine (callback function) ait OCX'le ilgili bilgiyi uygulama içerisine aktarır. Çalışma zamanı kütüphanesinin OCX işlevlerini çağırabilmesi için geri çağırma işlevlerinin kaydedilmesi gerekir[9].

2.3.3. SPSS OLE Nesnelerinin OLE Otomasyonuna Uygulanması

Nesne tabanlı programlama program nesnelerinin üzerinde tasarım, düzenleme ve değişiklik yapmaya olanak verir. Program nesnesi mesaj alıp verme yeteneğine sahip kara bir kutu gibi düşünülebilir. Bu kutu komut kodlarını (programlama dilinin komut dizileri) ve verileri (komutların üzerinde işlem yapmak için kullandığı veriler) aynı anda içerisinde barındırır. Geleneksel programlama yaklaşımında komut kodları ve veriler birbirinden ayrıktır. C, PASCAL, BASIC gibi geleneksel programlama dillerinde fonksiyonlar ve yapılar (structure) birleşik değildir. Örneğin C programlama dilinde komut kodu üniteleri fonksiyon olarak isimlendirilirken veriler yapı olarak değerlendirilir. Bir C fonksiyonu birden fazla yapıyı kullanabileceği gibi birden fazla fonksiyon aynı yapıyı kullanabilir [9],[10].

Nesne tabanlı programlama dillerinde komut ve veri nesne adı verilen bir programlama bileşeni içerisinde birleştirilir. Bu programcı için bazı büyük avantajlar sağlar. Ama daha önce nesne için kara kutu tanımlamasının neden kullanıldığını açıklamakta fayda var. Nesne tabanlı programlamanın temel kuralı nesne kullanıcısının bu kara kutu içerisinde ne olduğunu bilmeye ihtiyacının olmamasıdır.

Nesne, hakkındaki her şeyi tanımlayan classlar ile tanımlanır. Nesneler classların bağımsız durumları olarak tanımlanabilirler.[10]

Ayrıca nesne tabanlı programlama kavramı içerisinde "metot" teriminden de bahsedilir. Metot, basitçe tanımlamak gerekirse; kara kutuya yani nesneye gönderilen mesajın uyguladığı işlemdir. Nesneye gönderilen mesajın nesne içerisinde çalışmasını tanımlayan bir komuttur[10].

Ayrıca mesajların bir parçası olarak kullanılan argümanlar mevcuttur. Örneğin fetch (alıp getirme) mesajı neyi alıp getireceği bildiren bir argüman içerebilir veya roll-over (döngü) mesajı hangi hızda döneceğini bildiren bir argümanı içerebileceği gibi ikinci bir argümanla kaç kez döneceği tanımlanabilir[10].

Çizelge 2.1. Yüksek seviyeli OLE otomasyon nesneleri ve kullanıcı arabirimi karşılıkları

Object	User Interface
Application (ISPSSApp)	SPSS for Windows application
Options (ISPSSOptions)	Setting in the options dialog box (Edit menu)
SPSS Info (ISPSSInfo)	None
Documents (ISPSSDocument)	All windows open in SPSS
Data Document (ISPSSDataDoc)	Data Editor window
Syntax Document (ISPSSSyntax Doc)	Syntax window
Draft Document (ISPSSDraftDoc)	Draft Viewer window
Output Document (ISPSSOutputDoc)	Viewer window
Pivot Table (PivotTable)	Produced by many items on the Analyze menu
Igraph (ISPSS|Graph)	Produced by items on the graphs Interactive submenu
Text (ISPSSRtf)	Produced by some items on the Analyze menu
Chart (ISPSSChart)	Produced by items on the Graphs menu
Map	Produced by items on the Graph Map submenu

OLE otomasyonu nesnelerin başka uygulamalar, geliştirme araçları, makro dilleri içerisinde kullanılmasını sağlayan standart arabirim setlerinin kullanımına imkan verir. OLE daha genel kullanımı olan Component Object Model (COM)'un bir parçası olmakla birlikte onun kullanım özelliklerinden yararlanır. Programlama terminolojisinde nesne bir birim (unit) gibi davranabilen komut ve dataların bir birleşimidir. Örneğin bir kontrol, bir doküman içerisindeki bir madde, bir doküman veya bir uygulama programı. OLE nesnesi de böyle bir bileşen ya da COM nesnesidir.[10]

Bütün SPSS'e ait uygulamalarda kullanılan nesneler OLE taşıyıcısı (cotainer) olarak isimlendirilirler. Nesne tabanlı programlama dilleri genellikle editörlerinde OLE taşıyıcısı bulundururlar. Nesne tabanlı programlama yaklaşımıyla geliştirilen program SPSS'i çalıştırdıktan sonra ihtiyaç duyduğu nesnelere erişir. Nesnelerin kullanımına olanak tanıyan program OLE otomasyon sunucusu (server), nesneleri kullanan program -Akademistat- OLE otomasyon istemcisi (client) olarak bilinir.[10]

Çizelge 2.1'de SPSS'in OLE otomasyon istemcilerinin kullanımına olanak tanıdığı nesne classları olarak adlandırılan nesne tiplerini göstermektedir. Her nesnenin o nesneyle ne yapabileceğinizi tanımlayan özellik (PROPERTY) ve metot (METOD) olarak adlandırılan kendine ait nitelik (attribute) ve komutları (command) vardır.[10]

Çizelge 2.2. Akademistat modülünde kullanılması öngörülen istatistikler

Tanımlayıcı İstatistikler	Frequencies Explore Crosstabs
Ortalamaların Karşılaştırılması	Independent Samples T-Test Paired Samples T-Test ANOVA
Korelasyon	Bivariate
Regresyon	Linear Binary Logistic
Non-parametrik	Chi Square 2 Independent Samples K Independent Samples 2 Related Samples K Related Samples

SPSS ile OLE otomasyonu nasıl kullanıldı? Her şeyden önce programın SPSS ile nasıl etkileşeceğine karar verildi. Daha sonra uygulama programı yazıldı. Akademistat'ın SPSS aracılığıyla yapabileceği istatistikler Çizelge 2.2'de gösterilmiştir[10],[11],[12].

SPSS'in kendi ara yüzüyle çalışılırken kullanılan birçok özellik OLE otomasyonu tarafından kullanılır. Akademistat'ın içereceği OLE otomasyon özellikleri:

- SPSS veri dosyası açmak kaydetmek ve veri dosyası bilgisine ulaşmak,
- Çizelge 2.1'de belirtilen pivot table, grafik ve diğer istatistik çıktıları elde etmek üzere SPSS istatistik ve grafik prosedürlerini çalıştırmak,
- SPSS istatistikleri için ayarlar yapmak.

2.4. Yerel Ağ (Intranet)

Akdeniz Üniversitesi Hastanesi fiziksel olarak dağınık bir yapıdadır. Zaman içinde yeni binaların yapılması ile sistem bu yapıya uygun bir biçimde geliştirilmiştir. Akdeniz Üniversitesi Hastanesi dört ana binadan oluşmaktadır. Bu binalar arasında, paylaştırılmış üç adet veri tabanı bulunmaktadır. Merkezi hastane binasına en büyük boyutlu veri tabanı sunucusu yerleştirilmiştir. Bu sunucuda hasta verileri, önemli kodlar, yönetim verileri gibi en önemli bilgiler depolanmıştır. Radyoloji binasında Görüntü Arşivleme ve İletişim Sistemi (Picture Archiving and Communication System-PACS) bulunmaktadır ve bu veri tabanı sunucusu görüntü dosyalarını ve Health Level Seven (HL7) iletişim protokolünü depolamaktadır. AÜHBS PACS sistemi HL7 kod işleyicisine sahiptir ve bu sayede sistemdeki veri tabanları arasında güvenli veri transferini sağlamaktadır. Son veri tabanı sunucusu ise laboratuar verilerini depolamakta ve merkezi hastane binasındaki laboratuarda bulunmaktadır. Bu veri tabanı içerisinde resmi istem kodları ve laboratuar test sonuçları tutulmaktadır. Veri tabanı sisteminin tümü ilişkisel olarak organize edilmiştir ve nesneye dayalıdır[1],[11],[13].

2.4.1. Yerel Ağda Kullanılan Sunucu ve İstemci Özellikleri

Akdeniz Üniversitesi Hastanesi üniversite kampüsü içindedir. İçinde değişik binalar bulunmaktadır. Üniversite ağının omurgası binaları fiber optik kablolar ile birbirine bağlamaktadır. Binaların içinde UTP CAT-5 kabloları kullanılmıştır. Üniversite Bilgi İşlem Merkezi iki adet C-Sınıfı Internet Protokol (IP) grubu kullanmaktadır. Switch'ler 10/100 MB hıza sahiptir[1],[11],[13].

AÜHBS dahilinde kullanılan veri tabanı ve uygulama sunucularının sayıları sırasıyla dört ve sekizdir. Her bir veri tabanı sunucusu dört adet işlemciye sahipken uygulama sunucuları için bu sayı ikidir. Veri tabanı ve program sunucularının bellek kapasiteleri 1 GHz ve 256 MHz'dir. Veri tabanı ve uygulama sunucularının hard disk kapasiteleri ise sırasıyla 40 ve 4x8 GB boyutundadır[1],[11],[13].

AÜHBS'nin geliştirilme safhasında çeşitli işletim sistemleri kullanılmıştır. İlk olarak Data General AOS-VS kullanılmıştır. Daha sonra internet erişimi için sisteme yeni bir Unix sunucusu eklenmiştir. En son olarak da veri tabanı güvenliği ve programlama için NT sunucusu kullanımına başlanılmıştır. Ağ yönetimi ve hizmetleri için Windows 2000 Server Professional kullanılmaktadır[1],[2],[3],[11].

Kişisel bilgisayarlarda ise Windows 95 ve 98 işletim sistemleri bulunmaktadır. Yeni programların Windows XP ortamında geliştirilmesinden dolayı kademeli olarak kullanılan işletim sistemi ve tabii ki sistem konfigürasyonu da değişmektedir[1],[11].

Bir hastane bilişim sistemi için verilerin güvenli depolanması en önemli bileşendir ve sorgulama sistemi ya da veri yönetimi bulunmayan veri tabanı sistemlerinin düzgün bir şekilde çalışması olası değildir. Bu kriterlerin ışığında mümkün olan en iyi veri tabanı sisteminin kullanılması zorunludur. Hastane Bilişim Sistemi dahilinde halen kullanılmakta olan veri tabanı yönetim sistemi bileşenleri aşağıda gösterilmiştir[1],[11].

Sunucular

- ORACLE 7.3.3 for Windows NT
- ORACLE Enterprise Manager 1.3.5
- 1 Ana, toplam 6 veri tabanı

İstemciler

- Personnel ORACLE 7.3.3 for Windows
- SQL*NET Client 2.3.3
- ORACLE Protocol Adapters 2.3.3.

Merkezimizdeki tüm bu uygulama programları visual programlama dilleri kullanılarak geliştirilmiştir. Veri tabanı programlaması ve sorgulanması için SQL seçilmiştir. Tercih edilen programlama dilleri ise aşağıda sıralanmıştır[1],[11]:

Sunucular

- ORACLE (SQL*Plus ve PL/SQL)
- Clarion for Windows (C5EE)

İstemciler

- Clarion for Windows (C5EE)

AÜHBS içerisinde yüklenmiş olarak bulunan Radyoloji Bilişim Sistemi (RBS), Laboratuar Bilişim Sistemi (LBS), Patoloji Bilişim Sistemi (PBS) ve Nükleer Tıp Bilişim Sistemi (NTBS) gibi farklı bilişim sistemleri vardır. Tüm sistem birbirine sorgular vasıtası ile bağlanmıştır. Sisteme girme izni olan tüm kullanıcılar birbirlerine bu sorgular ile bağlanmaktadırlar[1],[11].

2.5. SQL (Structural Query Language)

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

SQL'de kullanılan create, alter, drop komutları ile veritabanı nesneleri üzerinde yapılan işlemlere veri tanımlama (data definition) adı verilir. Dolayısıyla bu komutların kullanımıyla oluşan dile veri tanımlama dili adı verilir. Veritabanı nesnelerinin oluşturulması (create), değiştirilmesi (alter) ve silinmesi (drop) işlemlerini hakların ve rollerin verilip alınmasını kontrol etmek, tablo, indeks ve clusterların analiz edilmesini sağlamak için kullanılır. Bu dille yaratılan nesneler veritabanında veri sözlüğünde tutulur. Bu komutlar doğrudan veritabanını etkilediği için yaptıkları değişiklikler doğrudan veri sözlüğüne yansıtılır[5],[14].

Sekiz tane temel DDL komutu mevcuttur.

- CREATE Nesne yaratmak için kullanılır.
- ALTER Nenenin yapısını değiştirmek için kullanılır.
- DROP Nesneyi silmek için kullanılır.
- GRANT Hak vermek için kullanılır.
- REVOKE Verilen hakkı geri almak için kullanılır.
- ANALIZE Tablo, indeksler hakkında istatistik hazırlamada kullanılır.
- AUDIT Veritabanı nesnelerinin kontrol işlemleri için kullanılır.
- COMMENT Veritabanı nesneleri için yorum yazmada kullanılır.

2.5.2. Veri Sözlüğü (DATA DICTIONARY)

Veri sözlüğü içinde veritabanı hakkında sadece okunabilir nitelikteki bilgilerin (veritabanındaki kullanıcılar, kullanıcıların hakları, veritabanı nesneleri, tabloların kısıtlamaları hakkında bilgi, vb.) bulunduğu tablo ve görüntüler kümesidir. Veri sözlüğü içerisinde veritabanına ait bilgiler bulunur. Veri sözlüğü veri tabanının kuruluşunda - sonradan ekleme yapılabilecek şekilde- standart olarak yaratılır. Veritabanında ki değişiklikler doğrudan veri sözlüğünü de etkiler. Kullanıcının bir tablodaki herhangi bir yapıda yaptığı değişiklik veri sözlüğüne de etki eder. Böylelikle veri sözlüğü veritabanına ait güncel ve sağlıklı bilgi alınmasına olanak sağlar[5],[14].

Her kullanıcı sahip olduğu haklar çerçevesinde veri sözlüğünden yararlanabilir. SQL cümleleri ile veri sözlüğünden veriler alınabilir.Veritabanının bütünlüğünün sağlanabilmesi için hiçbir kullanıcının veri sözlüğünün yapısında ve içeriğinde değişiklik yapma yetkisi yoktur. İlişkisel Veritabanı Yönetim Sistemlerinde veritabanında olan değişiklikler anında veri sözlüğüne yansıtılır[5],[14].

2.5.3. Veri İşleme Dili (Data Manipulation Language-DML)

DDL ile oluşturulmuş olan tablo ve görüntülerin içlerindeki kayıtlar üzerindeki değişiklikler (ekleme, silme, güncelleme bakma) veri işleme dili ile yapılır.

Altı tane temel DML komutu mevcuttur[5],[14].

2.5.3.1. Seçme (Select)

Bir veya daha çok tablo ve görüntünün içindeki verileri listeler. SELECT komutu ile kayıtlar ve sütunlar listelenebilir. İç içe alt sorgulu (subquery) olarak da kullanılabilir[14]. Komutun yapısı şöyledir[5]:

```
SELECT [ALL| DISTINCT]{* | [<nesne ismi>].<sütun ismi>, ...}
FROM {<nesne ismi> [@<bağlantı ismi>][<diğer isim>],...}
[WHERE <şart bölümü> [alt sorgu]]
[[START WITH <şart bölümü>]CONNECT BY <şart bölümü>]
[GROUP BY {<sütun ismi>,...}]
[HAVING<şart bölümü>]
[ORDER BY {<sütunun sırası>|<diğer isim>| <sütun ismi>,...}[ASC| DESC]]
```

2.5.3.2. Ekleme (Insert)

Tablo ve görüntülere kayıt girmek için kullanılan komuttur[14]. Yapısı[5]:

```
INSERT INTO {<nesne ismi>@<bağlantı ismi>}[(<sütun ismi,...>)]
VALUES (<değer>,...)|<altsorgu>
```

2.5.3.3. Günleme (Update)

Tablo ve görüntülerdeki kayıtların değiştirilmesini sağlar. Tablodaki kayıtlarda değişiklik yapabilmek için tablonun kullanıcıya ait olması veya başkasının tablosuysa kullanıcının UPDATE ve UPDATE ANY TABLE haklarından en az birine sahip olması gerekir[14]. Yapısı[5]:

```
UPDATE {<nesne ismi>@<bağlantı ismi>}[<diğer isim,...>]
SET (<sütun ismi>,...) = {<sütun ismi>,...| <altsorgu>}
WHERE <şart bölümü>
```

2.5.3.4. Silme (Delete)

Tablo veya görüntülerdeki kayıtları değiştirmek için kullanılır. Kullanıcıların bu komutu kullanabilmeleri için tablonun kendilerine ait veya DELETE ya da DELETE ANY TABLE haklarından en az birinin verilmiş olması gerekir[14]. Kullanımı[5]:

DELETE FROM {<nesne ismi>@<bağlantı ismi>}[<diğer isim,...>)]
WHERE <şart bölümü>

2.5.3.5. Plan Açıklama (Explaın Plan)

Veritabanının performansı için kullanılan bir komuttur. Hangi SQL'lere hangi işlemlerin uygulandığını gösteren ve o işlemleri tabloya alan komuttur[5],[14].

EXPLAIN PLAN [SET STATEMENT_ID = 'yazı']
INTO {<nesne ismi>[@<bağlantı ismi>]}

2.5.3.6. Tablo Kilitleme (Lock Table)

Bir veya daha fazla tabloyu belirtilen şekilde kilitleyen komuttur. Kilitlenen tablo belirli bir süre için erişime kapatılmış demektir. Çeşitli uygulamalardan veya SQL*Plus'tan çalıştırılan SQL sorguları sonucunda veritabanı bunlar için uygun kilitlemeleri oluşturur. Tablo kilitleme komutu otomatik kilitlemeyi devreden çıkarttığı için, yanlış kullanım sonucunda tablo tamamen erişilemez hale gelebilir. Bunun için bu komutun kullanımında dikkatli olunmalıdır[14]. Kullanımı[5]:

LOCK TABLE {<nesne ismi> [@<bağlantı ismi>]}
IN <kilitleme çeşidi>MODE [NOWAIT]

2.6. SPSS

SPSS akademik ve bilimsel araştırmalarda sıklıkla kullanılan istatistiksel analiz imkanı sağlayan bir paket programdır. Akademistat modülünün geliştirilmesi safhasında SPSS programının seçilmesi için birçok değişik sebep bulunmaktaydı. Bunlardan bazıları; SPSS'in Windows tabanlı olması, hafızada istatistiklerin hızla yapılabilmesi için bir tür sunucu bulundurması ve nesneye dayalı uygulamaların geliştirilmesine imkan tanımasıdır. SPSS programının bu kadar popüler olmasını sağlayan bir diğer avantajı da program ile gerçekleştirilen her prosedürün nesne dosyaları ile geliştirilebilmesidir [10],[11].

Gerçek dünya nesneleri gibi OLE otomasyon nesneleri de öğelere ve kullanım şekillerine sahiptirler. Programlama terminolojisinde bu öğelere özellik, kullanım şekillerine de metot denmektedir. Her bir nesne sınıfı o nesnelerle neler yapabileceğimize karar vermemizi sağlayan özel metot ve özelliklere sahiptir.

Özellikler nesnelerin renk veya hücre derinliği gibi bazı niteliklerini belirler yada dönüştürür. Metotlar ise nesneler üzerinde bazı işlemleri gerçekleştirir. Örneğin tablo seç metodu bir tablonun içindeki tüm elemanları seçmede kullanılır yada yazı fontu özelliğini kullanarak seçili yazı bölgesinin fontunu belirler (veya mevcut durum öğrenilmek isteniyor ise fontu görüntüler) [1].

Çizelge 2.3. Gerçek hayat-SPSS nesneleri karşılaştırması örneği

Nesne	Özellik-property	Metot-method
Kalem (Gerçek Hayat)	Sertlik Renk	Yazma Silme
Pivot Tablo (SPSS)	Yazı Fontu VeriHücresi Derinliği Başlık Teksti	Tablo Seç Seçimi Temizle Alt Bilgileri Sakla

Çoğu SPSS nesnesi, nesnenin niteliklerini sorgulamamızı sağlayacak özelliklere ve nesne üzerinde değişiklik yapabilmemizi sağlayacak metotlara sahiptir. SPSS metotları ile bazı yüksek-seviyeli OLE otomasyon nesnelerinin özellikleri Çizelge 2.4'de verilmiştir[1].

*Özellikler(*Properties*):* Özellikler nesnelerin niteliklerini belirler yada dönüştürür. Bazı özellikler yukarıda da anlatıldığı gibi diğer nesnelere dönüşümü sağlarken diğer bazı özellikler renk yada derinlik gibi niteliklerin dönüşümünde kullanılır[11].

Metotlar(Methods): Bir tablodaki tüm elemanların seçilmesi ya da seçili bir şeyin bırakılması gibi nesneler üzerinde bazı işlemlerin gerçekleştirilebilmesini sağlarlar. Özellikler gibi metotlar da başka nesnelere dönüşümü sağlayabilirler[11].

Aşağıdaki değişken isimleri programa dahil edilmiş olan örnek metinlerde kullanılmıştır ve tüm metinler için önerilmektedir. Pivot tabloların dışındaki tüm nesne sınıflarının isimlerinin ISPSS ile başladığına dikkat edilmelidir[10],[11].

Bir nesneyi almak demek o nesneye bir referans yaratarak özelliklerini ve metotlarını yapılacak şeyler için kullanılır hale getirmek demektir. Elde edilen her bir nesne referansı bir değişkende depolanır. Bir nesneyi alabilmek için öncelikle uygun sınıftan bir nesne değişkeni tanımlamak (declaration) ve sonra bu değişkeni belirli nesneye ayarlamak gerekir (Çizelge 2.5.). Örneğin Clarion programlama dilinde dizayn edilmiş çıktı dokümanı alabilmek için; [2],[11]

```
OutputDoc=?SPSS{'GetDesignatedOutputDoc'}
```

Çizelge 2.4. Yüksek seviyeli OLE otomasyon nesneleri için bazı özellik ve metotlar

Nesne	Özellik (Property)	Metot
ISPSSApp	Document Options SPSSInfo	ExecuteCommands GetDesignatedOutputDoc NewDataDoc OpenDataDoc Quit
ISPSSOptions	DisplayCommands OutputBeep WarningVisible	None
ISPSSInfo	NumVariables VarType	GetSelectedVariables
ISPSSDocument	DataDocCount	GetDataDoc
ISPSSDataDoc	Modified PromptToSave Visible	Copy GetNumberOf Cases GetVariables SelectCells SaveAs
ISPSSSyntaxDoc	Designated PromptToSave Text	Close PrintDoc Run SaveAs
SPSSOutputDoc	PrintOptions SplitterPosition Visible	ClearSelection ExportCharts InsertTitle SelectAllMaps
ISPSSDraftDoc	Height Width WindowState	Close GetDocumentPath PrintRange
ISPSSItems	Count	GetItem
PivotTable	BackgroundColour TableLook TextStyle	CreateChart HideFootnotes SelectCaption ShowAll
ISPSSGraph	CoordinateSystem Elements Title	DeleteTitle GetElement Redraw
ISPSSRTF	None	RtfText
ISPSSChart	None	ExportChart
Map	None	None

Nesne hiyerarşisinde daha üst sırada bulunan nesnelerin özellik ve metotları, alttaki nesneler tarafından alınmak için kullanılırlar. Yukarıdaki ikinci önerme, dizayn edilmiş çıktı dokümanını üst-seviye nesnesi olan uygulama nesnesine ilişkin bir metot görevindeki GetDesignatedOutputDoc vasıtasıyla alır. Benzer şekilde bir pivot tablosu nesnesi alabilmek için önce pivot tablosuna sahip, çıktı dosyası çağırılmalı, ardından o çıktı dosyasındaki tüm parçalar alınmalı ve bu şekilde devam edilmelidir[10],[11],[12].

SPSS paket programı istatistiksel sonuçlar üretmek için otomasyon nesneleri yaratabilmektedir. İstenen istatistiksel sonuçları elde edebilmek için OLE otomasyonu kullanılabilmektedir. Bu özellikler SPSS programını kendi uygulamamızda kullanmamız açısından bize imkanlar sağlamakta ve bu yolla araştırmacılar SPSS programını bilgisayarlarına yüklemeksizin istatistik hesaplar yapabilmektedirler[10],[11].

Çizelge 2.5. Nesne tipi veya sınıflarının değişken isimleri

Nesne	Tip veya Sınıf	Değişken İsmi
SPSS application	ISPSSApp	objSPSSApp
SPSS options	ISPSSOptions	objSPSSOptions
SPSS file info	ISPSSInfo	objSPSSInfo
Documents	ISPSSDocuments	objDocuments
Data document	ISPSSDataDoc	objDataDoc
Syntax document	ISPSSSyntaxDoc	objSyntaxDoc
Viewer document	ISPSSOutputDoc	objOutputDoc
Print options	ISPSSPrintOptions	objPrintOptions
Output items collect	ISPSSItems	objOutputItems
Output item	ISPSSItem	objOutputItem
Chart	ISPSSChart	objSPSSChart
Text	ISPSSRtf	objSPSSText
Pivot table	PivotTable	objPivotTable
Footnotes	ISPSSFootnotes	objFootnotes
Data cells	ISPSSDataCells	objDataCells
Layer labels	ISPSSLayerLabels	objLayerLabels
Column labels	ISPSSLabels	objColumnLabels
Row labels	ISPSSLabels	objRowLabels
Pivot manager	ISPSSPivotMgr	objPivotMgr
Dimension	ISPSSDimension	objDimension

2.7. Akademistat İstatistik Modülü

Akademistat, AÜHBS üzerinden araştırma yapmak isteyen araştırmacılara ihtiyaç duydukları verileri sağlamak amacıyla geliştirilmiştir.

Uygulamaya belli zaman aralığı tanımlaması, hastalara uygulanan testlerin tanımlamaları ve hasta dosya numaraları ile kullanıcı tanımlamaları gibi kriterler yerleştirilmiştir[18],[19],[20],[21].

Bu kriterlere göre sorgulama gerçekleştirildikten sonra AÜHBS veritabanı üzerinden bir çıkış dosyası oluşturulur. SPSS ile sorgulanan verilere istatistikleri uygulamak için SPSS veri dosyasına (SAV) ihtiyaç vardır. Bunun için oluşturulan veri dosyasının uzantısı SAV olarak meydana getirilir. Ancak bu oldukça fazla program kodu yazmayı gerektirdiğinden veri dosyasının formatı (.TXT) olarak kaydedilmektedir.

Üretilen TXT dosyasına istatistikleri uygulamak için uygulama programı ara yüzünden çeşitli istatistiklerin seçilmesi öngörülmüştür. İstatistik çeşidi seçildikten sonra program ilgili prosedürü çağırır. Prosedürler SPSS oturumunu başlatır. SPSS programı, Akademistat içerisinde tanımlanmış olan otomasyon nesne tiplerini OLE otomasyonuyla çalıştırır[10],[11].

SPSS, AÜHBS'nden istenen sorgulama veri dosyasına uygulanan testlerin çıktısını meydana getirir. Çıktı dosya oluştuktan sonra kullanıcı bu dosyaya müdahale edebilir.

Akademistat modülü doktor ve diğer araştırmacıların akademik çalışmalarında ve günlük klinik çalışmalarında onlara yardımcı olması için tasarlanmış ve oluşturulmuştur. Bu modül daha önce AÜHBS'nde mevcut olmayan bir eksikliği yerine getirmektedir[22],[23],[24],[25].

Akademistat modülünün yapısı ve kullanımı akademik ve klinik açıdan kullanım için elverişli bir şekilde geliştirilmiştir. Bu modülün diğer tıp fakülteleri ve hastaneler için bir örnek olmasını umarız. Uygulamanın sadece araştırmacıların ihtiyaç duydukları verileri elde etmek için bir kaynak olarak değil, aynı zamanda hastane bilişim sistemi üzerinden hasta verilerinin paylaşımı için rutin sağlık hizmetlerinde de yardımcı olacağı düşünülmektedir. Böylece hekimlerin klinik karar verme süreçlerinde yardımcı olması beklenmektedir[17],[18],[19],[20].

Günümüzde bilimin hemen hemen bütün dalları kendi bünyelerine hızlı elektronik gelişmeleri adapte etmektedir. Bilişim sistemleri ve elektronik veritabanları gün geçtikçe daha fazla önem kazanmaktadır. Sağlık bakım kalitesini artırmak için doktorlar kendileri için gerekli verilere ve elektronik kaynaklara erişebilmelidirler. Bunun için Akademistat modülü hastane bilişim sistemi için bir gerekliliktir diyebiliriz.[9], [11], [12], [13]

Beklenen odur ki Akademistat modülü birçok doktor ve araştırmacıya günlük klinik uygulamalarında ve akademik çalışmalarında fayda sağlayabilir. Ayrıca umarız ki bu çalışma dünya üzerindeki hastane bilişim sistemlerine sahip olan hastaneler ve tıp fakülteleri için de faydalı olur.[9], [11], [12], [13]

GEREÇ VE YÖNTEM

Fiziksel olarak dağınık bir yapıya sahip olan Akdeniz Üniversitesi Hastanesini meydana getiren dört temel bina mevcuttur. Bu binaların tamamı HBS'ne bağlanmış durumdadır. HBS'ne entegre biçimde çalışan radyoloji ve laboratuar bilişim sistemleri de sistemi destekleyen önemli alt sistemlerdir.

Laboratuar bilişim sistemi içerisinde sisteme doğrudan bilgi aktaran ölçüm cihazları olduğu gibi sonuçların veri giriş uzmanları tarafından girilmesini gerektiren cihazlar da mevcuttur. Sisteme direkt bağlı olan cihazlar ölçüm materyallerinin cihaza yüklenmesinin ardından veritabanına ilişkisel olarak veri aktarımı sağlamaktadır. Otomatik veri aktarımına imkan tanımayan ölçüm cihazlarında ise elde edilen veriler gerekli ara yüzler aracılığıyla veri giriş uzmanları tarafından veritabanına aktarılmaktadır.

Her üç bilişim sisteminde oldukça fazla veri üretilmektedir. HBS içerisinde idari açıdan kullanılan ve genellikle finanssal değerlendirmelere olanak tanıyan bir istatistik modülü kullanılmaktadır. Bu modülün işlevi günde, haftada, ayda kaç hasta hangi polikliniklere veya kliniklere geliyor; kaç tane, hangi tür ameliyat yapılıyor; kaç ameliyat başarıyla sonuçlanmış vs. gibi temel tanımlayıcı istatistiklerin yapılmasıdır.

Bu modüle entegre edilecek olan yeni istatistik modülü daha çok akademik amaçla kullanılması amaçlanmış bir uygulama programı niteliği taşımaktadır. Yeni modülü oluşturan beş temel unsur bulunmaktadır. Bunlar; Windows işletim sistemi, Oracle veritabanı yönetim sistemi, SQL, SPSS paket programı ve Clarion 5.5 EE (Enterprize Edition) programlama dili.

Laboratuar sistemine dahil olan hastalardan elde edilen materyallere bağlı olarak çeşitli biyokimyasal, mikrobiyolojik, genetik vb. veriler HBS'ni oluşturan veritabanına otomatik olarak veya kullanıcı ara yüzü aracılığıyla aktarılmaktadır. Bilimsel makale yayınlamak isteyen bilim insanları genellikle araştırdıkları konuya ilişkin verileri veritabanı üzerinden çeşitli yöntemlerle tarayarak veya klinik ve poliklinik hastalarının laboratuar verilerini takip ederek elde edebilmektedir. Akademistat modülü belirli kriterlere göre veritabanı üzerindeki tablolardaki verileri tarayarak elde edilen verileri belli bir formatta yeni bir veri dosyasına dönüştürmektedir. Elde edilen veri dosyası SPSS'e nesne (object) dosyalarının da kullanımıyla SPSS veri dosyası (.SAV) olarak tanıtılabilmekle birlikte Akademistat istatistik modülünde ASCII dosya (.TXT) haline getirilmektedir. Yine SPSS'in söz dizim (systax) dosyaları ile tanımlanacak olan istatistikler SPSS server tarafından işletilerek bir çıktı (output -.SPO-) dosyası elde edilmektedir. Çıktı dosya üzerinde istenirse yine SPSS söz dizim dosyalarıyla yazıcı çıktısı alma, grafiklerin düzenlenmesi gibi çeşitli işlemler yapılabilmektedir.

Modülün geliştirilmesinde HBS'nde kullanılan bütün modülleri geliştirmek için kullanılan programlama dili olan Clarion programlama dili kullanılmıştır.

Akademistat istatistik modülünün HBS içerisinde birçok kullanım alanı vardır. Modül içerisine zaman içerisinde farklı verilerin seçimini sağlayacak olan sorgulamalar eklenebileceği gibi kullanılabilecek farklı istatistikler de yerleştirilebilecektir. Bu özelliklerinden dolayı Akademistat modülü gelişmeye açık bir uygulama programı özelliği taşımaktadır.

BULGULAR

4.1. Sistemin Genel Değerlendirilmesi

Akademistat modülü bilişim sistemlerinin temel unsurlarından olan VTYS olarak kullanılan Oracle 7.3.3, veritabanı sorgulama dili olarak kullanılan SQL, program geliştirme aracı olarak kullanılan Clarion EE 5.5, yerel ağ donanımı ve SPSS paket programıyla etkileşim içerisindedir. Modülün her bir unsurla etkileşimi sonucunda elde edilen bulgular ayrı başlıklar altında incelenmiştir.

4.1.1. Veritabanı ve VTYS Açısından Bulgular

Oracle VTYS ilişkisel veritabanlarının bulundurması gereken minimum veri tekrarı, verinin tutarlılığı, verinin bütünlüğü, veri paylaşımı, uygulama geliştirme kolaylığı, güvenlik, çöküntülere karşı korunma, yedek alma, veri erişiminde kolaylık ve veri bağımsızlığı olanaklarının tamamına sahip olmasından dolayı özellikle yoğun hastane veri trafiği arasında veritabanın sorgulanmasında sorgu sonucunun hızlı dönmesini sağlamaktadır. Ayrıca modülün geliştirilmesi aşamasında veritabanı tablolarının bir bütün halinde incelenebilmesini sağlayan Oracle Navigator program ara yüzü ile laboratuar veritabanının bir bütünlük içerisinde tanımlanmasına olanak verir. Laboratuar veritabanından Akademistat modülünün listelerine SPSS programına aktarılacak olan verileri oluşturmak amacıyla sorgulamaya tâbi tutulan tablolar, bu tabloları oluşturan alanlar ve veri tipleri aşağıda tablolar ve bu tablolar içerisinde programa veri aktarmak için kullanılan alanlar şunlardır:

TESTS tablosunda bulunan alan isimleri:

Çizelge 4.1. TESTS tablosunu oluşturan alanların isim ve veri tipleri.

CODETYPE(CHAR)	SPECNUM(NUMBER)	LIMITTYPE(NUMBER)
TESTNUM(CHAR)	LISUNIT(VARCHAR2)	NHIGHLIMIT(NUMBER)
TESTNAME(VARCHAR2)	SIUNIT(VARCHAR2)	NLOWLIMIT(NUMBER)
SHORTNAME(VARCHAR2)	SIFACTOR(NUMBER)	WHIGHLIMIT(NUMBER)
MODELNUM(CHAR)	VERSIONCNT(NUMBER)	WLOWLIMIT(NUMBER)
INSNUM(CHAR)	MEASURECNT(NUMBER)	SAMPVOLUME(NUMBER)
LABNUM(NUMBER)	VALUETYPE(NUMBER)	DATESHIFT(NUMBER)
TUBENUM(NUMBER)	ACTIONTYPE(NUMBER)	NOTES(VARCHAR2)

Bu tablodaki, CODETYPE ve TESTNAME alanları laboratuarda uygulanmakta olan testlerin sorgulanması ve listeye aktarılması için kullanılmıştır.

Laboratuar sonuçları ve sonuçlarla ilgili verilerin tutulduğu ORDERRESULTS tablosunda bulunan alan isimleri:

Çizelge 4.2. ORDERRESULTS tablosunu oluşturan alanların isim ve veri tipleri.

ORDERDATE(DATE)	*CODETYPE(CHAR)*	*RESULTVALUE(VARCHAR2)*
ORDERNUM(NUMBER)	*TESTNUM(CHAR)*	INSALARMCODE(VARCHAR2)
WORKNUM(NUMBER)	LISALARCODE(NUMBER)	
RESULTNUM(NUMBER)	CONFIRMED(NUMBER)	

ORDERRESULTS tablosunda listelere aktarılacak verileri içeren alanlar, ORDERDATE, ORDERNUM, CODETYPE, TESTNUM, RESULTVALUE

Laboratuar istemleriyle ilgili verilerin tutulduğu ORDERS tablosunda bulunan alan isimleri:

Çizelge 4.3. ORDERS tablosunu oluşturan alanların isim ve veri tipleri.

ORDERDATE(DATE)	PATNUM(NUMBER)	ORDERSTAT(NUMBER)
ORDERNUM(NUMBER)	SERTYPE(CHAR)	DOCTOR(CHAR)
HISORDERCODE(CHAR)	*SERNUM(CHAR)*	
WORKDATE(DATE)	ORDERTYPE(NUMBER)	

Listelere veri aktarımı bu tabloda kullanılan alanlar, ORDERDATE, ORDERNUM, SERNUM alanlarıdır.

Akademistat modülünde sorgulama amaçlı kullanılan tablolardan ORDERS tablosu Çizelge 4.3'de görülmektedir. Bu tablo üzerinde Akademistat modülünün kullanımı açısından önemli olan tanımlı alanlar; hastanın laboratuar isteminin tarihini (ORDERDATE), istemin yapıldığı servisi tanımlayan servis tipi (SERTYPE) ve servis numarası (SERNUM) verilerini, istem numarasını (ORDERNUM) ve hasta dosya numarasını (PATNUM) tanımlayan alanlardır. Diğer bir önemli tablo TESTS tablosudur. Bu tablo üzerinden testlerin kod tipi, test numarası, test ismi alanlarının sorgulanması sonucunda elde edilen veriler programda listelere aktarılmaktadır.

Sorgulama sonucunda testlerin sonuçlarının getirildiği Oracle tablosu ise ORDERRESULTS tablosudur.

Hasta verilerinin bulunduğu HASTA tablosu üzerindeki alanlar:

Çizelge 4.4. HASTA tablosunu oluşturan alanların isim ve veri tipleri.

NDOSYA(NUMBER)	UYRUK(CHAR)	KKURUM(VARCHAR2)
DOSYATAR(VARCHAR2)	OZELKOD(NUMBER)	HASTAKURUMNO(VARCHAR2)
DOSYATAR_DATE(DATE)	DOGUMTAR(CHAR)	KBIRIM(VARCHAR2)
DOSYATAR_TIME(DATE)	DOGUMTAR_DATE(DATE)	CIKISTAR(VARCHAR2)
DOSYATIPI(CHAR)	DOGUMTAR_TIME(TIME)	CIKISTAR_DATE(DATE)
DURUM(CHAR)	DOGUMYER(CHAR)	CIKISTAR_TIME(DATE)
KHASTAKABUL(CHAR)	BABAAD(VARCHAR2)	USERID (CHAR)
AD(VARCHAR2)	ANNEAD(VARCHAR2)	USERDATE(CHAR)
SOYAD(VARCHAR2)	TELEFONNO(STRING)	USERDATE_DATE(DATE)
CINSIYET(CHAR)	ADRES(VARCHAR2)	USERDATE_TIME(DATE)
KANGRUBU(CHAR)	POSTAKODU(STRING)	HASTAKARNENO(CHAR)

4.1.2. Uygulama Ara yüz Yazılımı Olarak Clarion EE 5.5 Açısından Bulgular

Clarion programlama dilini Visual Basic, Delphi, Power Builder ve diğer rakip programlama dillerinden ayıran en önemli fark bu dillerin, bir programlama diline veritabanı geliştirme araçlarının eklenmesi ile geliştirilmiş olmasına karşın Clarion'un programlama araçlarıyla geliştirilmiş olan bir veritabanı geliştirme sistemi olmasıdır. Clarion programlama dilinin temel özellikleri şöyle özetlenebilir: Bir 4GL (dördüncü jenerasyon programlama dili) olması, programların kolay okunması, geliştirilmesi ve gereksiz yazım kurallarından arındırılması amacıyla geliştirildiğinden dolayı Windows ve Dos ortamında hızlı, güçlü, güvenilir ve modüler program geliştirme olanağı tanıması, C programlama dili ile geliştirilmiş olmasından dolayı 4GL olanaklarını C performansı ile kullanması, Clarion 2.0 sürümünden itibaren Windows ortamında Dinamik kütüphane (DLL-Dynamic Link Library) kullanımı, kütüphane (LIB-Library) ve nesne dosyaları oluşturma olanağı sağlanmıştır. Ayrıca RAD, Activex, OLE, DDE, ODBC, SQL ve İstemci/Sunucu desteği de bu sürümden sonra kullanıma sunulmuştur.

Clarion programlama dilinde oluşturulan bir programın genel yapısı şöyle özetlenebilir:

```
PROGRAM
        [MAP                            ! Programın genel yordam listesi
                yordam_prototipleri
                işlev_prototipleri
                [MODULE (kaynakdosyası)! Dış yordam ve işlev varlıklar tanımı
                yordam_prototipleri
                işlev_prototipleri
                END]
        END]
        genel_veriler                   ! Programın genel veri tanımlamaları
                CODE                    ! Program satırları
                program_cümlecikleri    ! İşletilebilir program komutları
                [RETURN]
        etiket  ROUTINE                 ! Yerel altprogram
        program komutları
END
etiket  PROCEDURE (parametre listesi)   ! Yordam tanımı
        yerel_veriler
        CODE
                program cümlecikleri
END
        [RETURN]
etiket  ROUTINE             ! Yerel altprogram
        program komutları
END
etiket  FUNCTION (parametre listesi)    ! İşlev tanımı
        yerel_veriler
        CODE
                program cümlecikleri
END
        [RETURN]
etiket  ROUTINE             ! Yerel altprogram
        program komutları
END
```

4.1.3. Yerel Ağ (Intranet) Açısından Bulgular

Akademistat uygulama programının geliştirildiği morfoloji binasındaki internet bağlantı hızı aktif ağ cihazlarının yeterli olmamasından dolayı çok düşüktür. Bu durum uygulama aracılığıyla ağ üzerinden sorgulama yaparken çeşitli sorunlarla karşılaşılmasına sebep olmuştur. Sorgulamalar sonucunda geri dönen sonuçların sorgunun niteliğine bağlı olarak oldukça büyük hacimde olmasından istemde bulunan bilgisayarın verileri alması çok uzun sürmektedir ya da istem zaman aşımına uğrayabilmektedir.

4.1.4. SQL (Structural Query Language) Açısından Bulgular

Akademistat uygulama programı Akdeniz Üniversitesi Hastanesi'ne ait Merkez Laboratuarı verilerinin akademik çalışmalar yapılmasına olanak tanımak amacıyla geliştirilmesinden dolayı başlangıç için bu veritabanı üzerindeki tabloları kullanacak şekilde geliştirilmiştir. Bu tablolar ORDERRESULTS, TESTS, ORDERS tablolarıdır. Bunların dışında HASTA tablosundan hastalarla ilgili veriler alınmıştır.

4.1.4.1. Sorgu İyileştirme (Query Optimization)

Hastane bilişim sistemi veritabanında yoğun veri trafiğinden, teknik problemlerden veya geri bildirimin oldukça büyük bir hacim kaplayabileceğinden dolayı geri dönecek bilginin en hızlı şekilde dönmesini sağlamak üzere veritabanına gönderilen sorgulamanın en uygun halde gönderilmesi gerekir. Sorgulamanın en iyi hale getirilmesi işlemine sorgu iyileştirme adı verilir. Veritabanına uygulanacak sorgulamanın ve geri dönecek sorgulama sonucunun süresini en aza indirilerek için sorgulamada dikkat edilmesi gereken nokta sorgulamanın hangi tablo üzerinden başlatılacağını belirlemektir. Eğer sorgulama yapılırken bütün istemler (ORDER) ORDER tablosundan sorgulanıp bunların içerisinde Akademistat içerisinden tanımlanan kriterlere göre tekrar sorgulama yapılmaya çalışılırsa, sorgu bütün istem kayıtlarını tarayacak ve taranan bu kayıtların içerisinde istenilen kriterlere uygun sonuçlara ulaşmak için yeni bir sorgulama yapacaktır. Bu durumda sorgunun işletilmesi de elde edilecek çıktının görüntülenmesi de son derece zor olacaktır. Bunun için laboratuar test sonuçlarının bulunduğu ORDERRESULTS tablosundan Akademistat içerisinden tanımlanan kriterlere uygun sadeleştirilmiş bir sorgu sonucu elde edilerek bu sonuç içerisinde istenen testlere yönelik olarak ayrıca bir sorgulama yapılması, elde edilmek istenen sonuçlara daha hızlı ve sorunsuz ulaşmayı sağlar. Yeni yapılan sorgulama ile süzülmüş verilerin içerisinde ORDERDATE ve ORDERNUM alanlarının kullanılması suretiyle ORDERRESULTS tablosu içerisindeki laboratuar sonuçlarının hangi hastalara ait olduğunu tespit edilmektedir.

4.1.4.2. Sorgu Tablolarının Düzenlenmesi

Açıklanan sorgu iyileştirme işlemleri sonucunda elde edilen çıktılar bir düzenleme işlemine tabii tutulmazsa SPSS içerisinde kullanılması mümkün değildir. Çünkü sorgulama sonucunda oluşan çıktı Çizelge 4.5'de de görüldüğü gibi sorgu sonucunda alt alta bütün hastaların istenen bilgileri kullanışsız bir biçimde listelenecektir.

Çizelge 4.5. Sorgulama sonrası çıkan verilerin biçimi

OrderKod	SıraNo	DosyaNo	Cinsiyet	Yaş	Test	Sonuç
051020020..6	1	2563423	E	45	T1	50
051020020..6	2	2563423	E	45	T2	80,45
051020020..6	3	2563423	E	45	T4	121
051020021..3	4	4568215	K	38	T3	80
051020021..3	5	4568215	K	38	T5	12,25
051020025..4	6	1235456	K	40	T1	75
051020025..4	7	1235456	K	40	T5	18

SPSS programına bu verilerin aktarılabilmesi için verilerin belli bir düzende her sütuna o sütunla ilgili veriler yerleştirilecek şekilde düzenlenmesi gerekir. SPSS'e aktarılacak veri tablosu Çizelge 4.6'da gösterilmiştir.

Çizelge 4.6. Sorgu sonucu çıkan verilerin düzenlenmesiyle oluşturulan yeni tablo

OrderKod	SıraNo	DosyaNo	Cinsiyet	Yaş	Test1	Test2	Test3	Test4	Test5	Test6
051020020..6	1	2563423	E	45	50	80,45		121		
051020021..3	4	4568215	K	88			80		12,2	
051020025..4	6	1235456	K	40	75				18	

Tabloyu bu biçime getirmek için programcılık teknikleri kullanılmalıdır. SPSS'e aktarılacak olan yukarıdaki tablo biçiminde sorgu sonucunda hastaların sonuçları mevcut değilse bu hücreler SPSS tarafından kayıp değer (missing value) olarak kabul edilir.

4.1.5. SPSS Açısından Bulgular

SPSS istatistik paket programı bütün dünyada, yapılan akademik çalışmaların istatistiksel sonuçlarını ortaya koymak için kullanılan, oldukça kapsamlı bir istatistiksel paket programdır. Geleneksel istatistiksel çözümleme olanaklarının dışında OLE teknikleriyle, uygulama geliştirmeye olanak sağlayacak nitelikte geniş bir nesne kütüphanesini de bulundurmaktadır. Program geliştiricilerin kendi uygulama programları içerisine yerleştirebilecekleri SPSS otomasyon nesneleri Çizelge 2.4. ve Çizelge 2.5'de gösterilmiştir. Akademistat modülü SPSS paket programının 10.0 sürümü ile geliştirilmiş olmasıyla birlikte modülün kullanılan OLE otomasyon nesneleri açısından 11 ve 11.5 sürümleriyle de uyumlu olasından dolayı bu sürümlerle de çalışma olanağına sahiptir. SPSS'in Base Modülünde kullanılan bütün testler nesne tabanlı programlama açısından kullanılabilir niteliktedir.

SPSS paket programı birçok XLS, TXT, SAV, SPO gibi uzantılara sahip birçok dosya türünü açabilmektedir. Buradaki XLS uzantılı dosyalar Excel elektronik tablolama programının dosya uzantısını,TXT uzantılılar metin dosyası uzantılarını ve SAV ise SPSS'in kendisine ait veri dosya uzantısını ifade etmektedir. Belirtilen program uzantılı dosyalardan herhangi birisi veya birkaçı istatistiklerin yapılmasına yönelik SPSS veri dosyası türünü oluşturabilir. Akademistat uygulama programında TXT metin dosya türünün kullanılmasına karar verilmiştir. Bu tercihin sebebi programın kaynak kod miktarının azaltılmasında fayda görülmesi ve istenildiğinde TXT uzantılı dosyaları SPSS veri dosyasına çevirme olanağı bulunmasıdır.

Elde edilenTXT uzantılı veri dosyalarının boyutu yapılan sorgulamaya bağlı olarak çok fazla büyüyebilmektedir. Bunun için veri deseni (data patern) seçerken sorguların, elde edilecek sonuçları mümkün olduğunca daraltacak şekilde oluşturulması gerekir.

İstatistiklerin yapılmasını sağlayan komut dizini ardından oluşacak istatistik sonuçları SPSS çıktı dosyası (.SPO) olarak meydana gelir. Bu dosya SPSS'in kendi ara yüzündeki pencere mönüsünün altından görüntülenebilir.

Akademistat uygulama programı SPSS programının olanak verdiği bütün yüksek seviye OLE otomasyon nesnelerinin kullanımına olanak verecek şekilde geliştirilmeye çalışılmıştır. Zaman içerisinde SPSS'in sahip olduğu özelliklerden ihtiyaç duyulanlar uygulama içerisine eklenme olanağına sahiptir.

4.1.6. AKADEMİSTAT İstatistik Modülü Açısından Bulgular

Modülün geliştirilmesine karar verilmesindeki en önemli etken hastane ortamındaki yoğun veri trafiği içerisinden ihtiyaç duyulabilecek olanların seçilerek karar mekanizmalara karar destek hizmeti sunmak ve akademik personele yaptıkları çalışmalar sonucunda hastalarına ait laboratuar sonuçlarının değerlendirilerek akademik çalışmalarında belli özelliklere sahip hastalara ait verilerden elde edilen veri desenlerinin anlamlandırılmasını sağlamaktır.

Akademistat istatistik modülü bir veri madenciliği (data mining) uygulamasıdır. Uygulamada göz önünde bulundurulan veri desenini laboratuar tetkikleri sonucunda hastalardan alınan tetkik materyallerinden elde edilen laboratuar sonuçlarıdır. Laboratuar veritabanında birçok farklı klinik (yatan) ve poliklinik (ayakta tedavi) hastasına ait karakter (string), sayısal (numeric) veya binary (boolean) tipte laboratuar verileri mevcuttur. Tabii ki bu veritabanı üzerinden, HIS ortamının ilişkisel yapısından ötürü laboratuar sonuçlarının ait olduğu hastaların kimlik bilgileri ile fatura, tahakkuk bilgileri gibi verilere de ulaşılabilmektedir.

Genellikle hiçbir veritabanı veri madenciliği için özel olarak tasarlanmaz. Ancak veritabanı içerisinde çeşitli veri desenleri oluşturmak suretiyle belli amaca yönelik veriden bilgi elde etmeye yönelik uygulamalar geliştirilebilir. Akademistat da bu amaca hizmet eden bir modüldür.

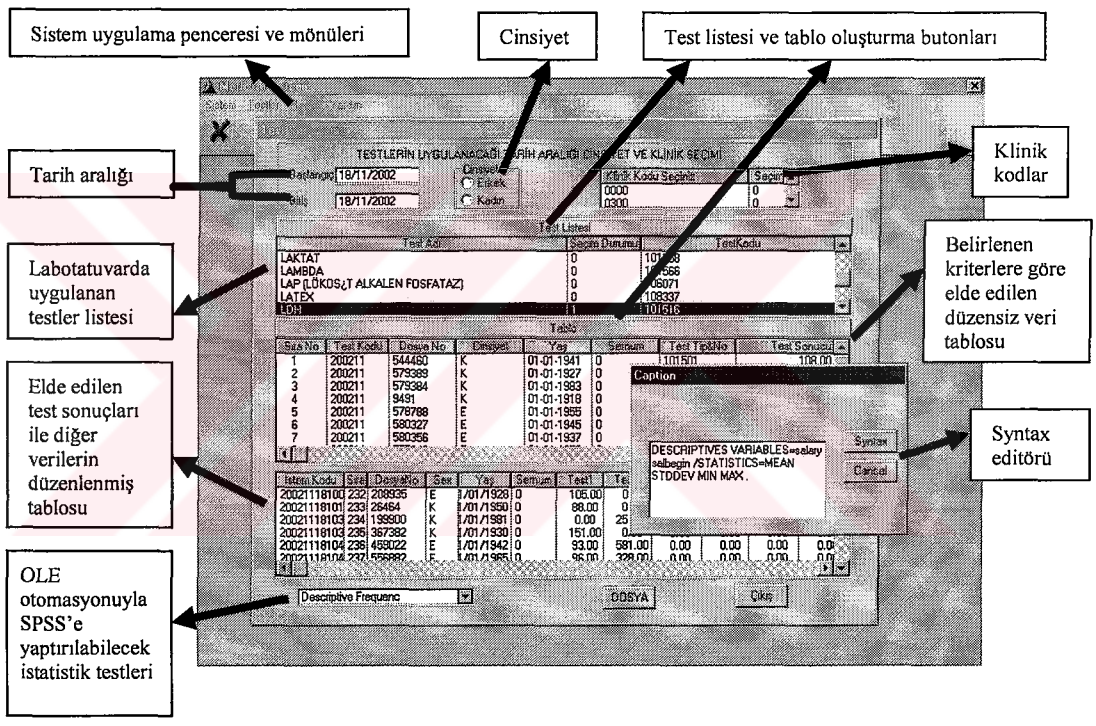
Akademistat istatistik modülü şöyle çalışır:

İki farklı şekilde çalışması düşünülen modülün uygulama penceresinden istenirse tek tek hasta dosya numaraları ve tanımlanan bir veri seti girilmek suretiyle hastalara ait laboratuar verilerine ulaşılarak bir tabloya aktarılır, ya da Merkez Laboratuar'da yapılan testlerin adları sorgulamayla bir listeye aktarıldıktan sonra buradan seçilen en fazla beş tane teste, seçilen iki tarih arasında kalan zamana, cinsiyete, istem yapan klinik veya doktora göre oluşturulan sorgu sonucu bir tabloya aktarılır. Elde edilen sorgu sonuç tablosu Çizelge 4.5.'deki gibi düzensiz bir tablo olduğundan bunu düzenlemek için ayrı bir yordam sorgu anında çalıştırılarak Çizelge 4.6'daki gibi bir tablo oluşturulur. İkinci yöntem rasgele (random) bütün sorgu kriterlerine uyan verileri bizim için veritabanından çektiği için tercih edilmiştir. Ancak diğer metodun da Akademistat'a adaptasyonu son derece kolaydır. Elde edilen veri tablosu hard diskte belirtilen bir dizinine (C:\Program Files\SPSS) "tablo.txt" ismiyle kaydedilir. Bundan sonraki aşamada SPSS'in program içerisinden çalıştırılarak veri dosyasının aktarılması gelir.

Veri dosyasının SPSS'e aktarılmasıyla program içerisinden hangi istatistik testlerin yapılacağı da birlikte verilir. SPSS'in nesne sınıflarıyla (class) gerçekleşen bu aşama sonrasında yapılan istatistik teste ait bir çıktı dosya oluşur. Bu dosyaya Windows mönüsü içerisinden ulaşılabilir.

Şekil 4.1'de modülün ara yüzü ayrıntılı olarak gösterilmektedir. Şekilden de görüleceği üzere laboratuarda uygulanan testlerin görüldüğü bölge bir liste kutusudur (list box). Sorgulama sonrası oluşan listeden seçilecek olan testlerin üzerinde sağ tuşa basılmasıyla o testlerin işleme tabii tutulacağı programa tanımlanmış olur. Testlerin sağındaki sütun seçilip seçilmediğini göstermektedir (1=seçili ; 0=seçili değil).

SPSS'te hangi istatistiklerin yapılacağıyla ilgili seçenekler modül ara yüzünün sol alt köşesinde bulunan aşağı açılır liste (combo box) içerisinde bulunmaktadır. Bu liste içinden bir testin seçilmesiyle o teste ait yordam çalıştırılarak daha önceden sorgulamalarla oluşturulmuş olan TXT dosyasını veri dosyası olarak kabul etmesi sağlanan SPSS'in, OLE otomasyonu aracılığıyla istenen testi yapması sağlanmaktadır.



Şekil 4.1. Akademistat uygulama programının ara yüzü

TARTIŞMA VE SONUÇ

Veri madenciliği uygulamaları son on-on beş yıl içerisinde son derece büyük bir artış göstermiştir. Hemen hemen bütün büyük yazılım şirketleri bu konuya yönelik ar-ge çalışmalarına büyük finansmanlar ayırtmaktadır. Çalışılan alanların çeşitliliği ve veri madenciliği uygulamalarını kullanacak olan işletmelerin sahip olduğu veritabanlarının birbirlerinden farklı olmaları nedeniyle veri madenciliği uygulamalarının standardizasyonunun sağlanması da oldukça büyük zorluklar içermektedir. Veritabanının içerdiği tabloların ve bunlara ait alanların birbirinden farklı olması bile standardizasyonun sağlanmasının karşısındaki en önemli etkendir. Bu sorunu aşmanın birkaç farklı yöntemi vardır. Bunlardan bir tanesi XML (Extensible Markup Language) desteği sağlayan, veritabanı uygulamalarında veri ile tabloların ilişkisel olarak tanımlamasının yapıldığı veritabanlarının oluşturulması ve veri paylaşımındaki problemlere bir standart getirmesi diyebiliriz, yani eskiden olduğu gibi veritabanına ali, veli, 315 tarzında veriler saklanmasının yerine,

<isim>ali</isim>

<soyisim>veli</soyisim>

<no>315</no>

seklinde bir standartlaşma sağlanabildiği ölçüde satır satır kod yazmaktan kurtulup bu ham veriyi her ortamda sorunsuz bir biçimde kullanabilme imkanı sağlanacaktır. Verilerin bu şekilde hiyerarşik olarak saklanması sayesinde istenen seviyedeki bilgiye ulaşmak çok kolaylaşacaktır. Bir işaretleme (markup) dili olmasından dolayı XML platformdan bağımsız çalışma olanağı da sağladığı için her işletim sisteminde, her disk ortamında sorunsuz kullanım olanağı sağlar.

Bir diğer çıkar yol ise veritabanlarının veri madenciliğine uygun biçimde tasarlanmasıdır. Veri madenciliğine uygun tasarım, verilerin belli desenlere (pattern) göre tasarlanarak veri tabanına kaydedilmesini ifade eder. Böyle bir tasarımın beraberinde getireceği sorunlar da vardır. Tasarlanan veri deseni miktarına bağlı olarak veritabanınındaki tabloların sayısı artacaktır. Buna bağlı olarak veritabanı tablolarının ilişkilendirilmesinde güçlükler çıkabilecektir. Genellikle veritabanları, veri madenciliği uygulamalarına yönelik olarak tasarlanmadıkları için geliştirilecek olan uygulama programlarında çeşitli programlama teknikleriyle veri desenlerini oluşturulur.

Bunların dışında sorgulama algoritmalarının optimizasyonu ile veritabanından çekilen verilerin hızla tablolara aktarılması mümkün olabilmektedir. Bu optimizasyon yapılmadığı durumda sorgulama sonucu verilerin dönüş hızı düşer, alınacak verilerin miktar ve özelliği isteğimize cevap vermeyecek veya gereksiz bilgiler sorgulanmış olacağından uygulama programının çalışmasında performans kaybına sebep olur.

Veri madenciliği uygulamaları en çok müşteri eğilimi saptaması, kredilendirme değerlendirmesi web üzerinden amaca yönelik veri toplama, sağlık alanına yönelik verilerin elde edilmesi amaçlarına yönelik olarak geliştirilirler.

Geliştirilen Akademistat istatistik modülü veri toplama değerlendirme açısından oldukça sınırlı bir çalışma niteliği taşır. Bunun sebebi veritabanı üzerinden tanımlanan veri deseninin tek amaca yönelik olarak seçilmesidir. Modül HIS üzerindeki Laboratuar verilerini ve hasta verilerini belli bir düzen içerisinde SPSS programının tanımlayacağı bir veri dosyasına çevirir. Bu dosyanın SPSS içerisine aktarılarak testlerin uygulanması ise OLE otomasyonu denilen programlama tekniği ile gerçekleştirilir. Modülün geliştirilmesi aşamasında yukarıda bahsedilen veri toplama güçlüklerine rağmen veri deseninin mümkün olduğunca sınırlandırılması modülün geliştirilmesini kolaylaştırmıştır.

Modülün çalışma performansına LAN hızı oldukça fazla etki etmektedir. Kliniklerden çok yapılan istemlerden bir güne ait sorgulamada 600-700 farklı durum (case) satırı oluşabilmektedir. Test sayısı artırıldığında bu sayı artmakla birlikte SPSS'e göre düzenlemesinin yapıldığı tabloda modülün performansında yavaşlama görülmektedir. Bu sorunun ağ hızındaki artışla aşılabileceği düşünülmektedir.

Sorgu en iyileştirme (optimisation) aşamasında hız kazancı sağlamak için her iki tablo birbirinden ayrı sorgulanmaktadır. Elde edilen birinci tablodaki veriler SPSS'de işlenemeyecek, ham veriler olduğundan bu veriler yeni bir tabloya SPSS'in kullanabileceği şekilde yeniden düzenlenerek TXT uzantısıyla bir dosyaya kaydedilmektedir. Oluşturulan dosyanın SPSS'e aktarılmasında bir diğer yöntem ise dosyanın SAV veya XLS uzantısıyla kaydedilmesidir. Bu durumda SAV uzantılı dosya oluşturulacaksa SPSS'in XLS uzantılı dosya oluşturulacaksa excelin save as nesneleri kullanılmalıdır.

Akademistat istatistik modülünün sorguları program içerisine yerleştirilmiş SQL komut cümleleriyle işletilmektedir. Daha farklı veri desenlerinin programa aktarılmasıyla birlikte sorgu cümlelerinin PL-SQL ile veritabanında tutulması ve modülden yapılan istemler ile bu sorguların veritabanı üzerinde işletilmesi hız kazancı sağlayacaktır.

Clarion programlama dili veritabanı uygulamalarına yönelik bir dil olduğu için son derece kullanışlı bir uygulama geliştirme aracıdır. Kapsadığı OLE taşıyıcısı nesne tabanlı program geliştirme teknikleri kullanmaya olanak vermekle birlikte SPSS'in bütün nesneleri OLE taşıyıcısına aktarılmamaktadır. Testlere yönelik olarak kullanılan bütün testlerin nesne sınıfları kodlar içerisinde tanımlanmalıdır. SPSS, daha önceden yazılmış script dosyalarının çalıştırılmasıyla da istatistik testlerin yürütülmesini sağlar. Ancak bunun için veri tablosunun alanlarının ve bu alanlarda kullanılan veri desenlerinin her bir script dosyasında açıkça tanımlanması gerekir. Oluşturulan script dosyaları program içerisine kod olarak da yerleştirilebilir -Akademistat içerisinde bu yöntem

kullanılmıştır- PL-SQL ile oluşturulmuş sorgu cümlelerinin veritabanına aktarıldığı gibi birer komut dizini halinde kaydedildiği, veritabanı üzerinden işletilebilir.

Akademistat istatistik modülünün kullanım amacı her ne kadar akademik amaçla kullanılabilecek verilerin veritabanından çekilerek belli istatistik testleri uygulamaksa da Clarion proje dosyasının geliştirilmesinde modülün geliştirilebilmesine olanak verecek biçimde tasarlanmasına dikkat edilmiştir. Clarion gibi 4GL programlama dilleri, geliştirilen uygulamayı bir proje gibi değerlendirir. Bir projeyi kaynak dosyaları (.CLW), sözlük (.DCT) veya kütüphane (.LIB) dosyaları, bağlantı dosyaları (.LNK, .MAP), proje dosyası (.PRJ) ve uygulama dosyası (.EXE) oluşturur. Akademistat geliştirilirken programın mümkün olduğunca modüler olabilmesi için programın bütün işlevsel kısımları ayrı birer kaynak dosyası olarak tanımlanmıştır. Örneğin tek tek hasta bilgileri girilerek hasta verilerini çeken program kodları, SPSS'in her bir testi için gereken program kodları, tablolar için kriterlerin tanımlandığı program kodları ayrı birer kaynak dosyası olarak yazılmıştır. Böylelikle istenildiği taktirde bu modüllere eklemeler yapılabilecek veya kullanımı sağlanamayan kısımlar rahatlıkla değiştirilip düzenlenebilecektir.

KAYNAKLAR

1. Özkaya M.S. "Akdeniz University Hospital Informatıon System", Healthcom 2001 Proc., 29 June 2001, L'Aquila, Italy
2. Altun Ş., "Akdeniz Üniversitesi Tıp fakültesi Hastanesi için Geliştirilen Hastane Otomasyon Sisteminin Dağıtık Veritabanı Modeli İle Tasarlanması", Yüksek lisans Tezi, 1997, Antalya
3. Şarlak S., Akdeniz Üniversitesi Tıp Fakültesi Hastanesi İçin Geliştirilen Hastane Otomasyon Sisteminde Yer Alacak Olan İdari ve Akademik Modüllerin İstemci/Sunucu Mimarisi Kullanılarak Geliştirilmesi", Yüksek Lisans Tezi, 1996, Antalya
4. O'Brien James A., "Introduction to Information Systems", Eight Edition, 1997, USA
5. Şen O.N., "Oracle, Sql, SqlPlus, PL/Sql ve Veritabanı Yönetimi", 2. Baskı, Beta Basım Yayım Dağıtım A.Ş., Mayıs 2000, Cağaloğlu-İstanbul
6. Bobrowski S., Armstrong E., "ORACLE 7 Server Administrator's Guide", December, 1992
7. Topspeed Corporation, "ORACLE Connect Driver Reference" 1994, 1995 USA
8. Bobrowski S., Armstrong E., "ORACLE 7 Server Concepts Manuel" December, 1992
9. Topspeed Corporation, "Clarion For Windows User's Guide" 1994, 1995 USA
10. SPSS Inc., "SPSS 10.0 Developer's Guide" Copyright © 2000 by SPSS Inc. USA
11. Ozkaya M.S., "Linking and Embedding SPSS to Akdeniz University Hospital Information System", Healthcom2002 Proc., 6 June 2002, Nancy, France
12. SPSS Inc., "SPSS 10.0 Syntax Reference Guide" Copyright © 1999 by SPSS Inc. USA
13. J.H. van Bemmel ve arkadaşları "Handbook of Medical Informatics", 1997, Bohn Stafleu Van Loghum, Houten, The Netherlands
14. Neuman L.,Venning S., "SQL*Net Administrator's Guide", 1993, Oracle Corp.
15. Sosa M.-Iudicissa, Levett J., Mandil S., Beales P.F. "Health, Information Society and Developing Countries", Volume 23, 1995, IOS Pres, Amsterdam, Oxford, Tokyo, Washington, DC
16. Coiera E., "Guide to Medical Informatics, The Internet and Telemedicine", First Edition, 1997, Alden Press, Oxford, Great Britain
17. Sokal R.P., Rohlf F.J., "Biometry", Second Edition, 1981, W.H. Friman and Company, USA
18. Han J., "Data Mining Techniques", SIGMOD '96 6/96 Montreal, Canada
19. SPSS Inc. White Papers, "Basic Applied Techniques", ftp://ftp.spss.com/pub/web/wp/BASICWP.pdf , Ekim 2002
20. SPSS Inc., White Papers, "Data Mining: An Introduction" ftp://ftp.spss.com/pub/web/wp/WPDMINTRO-0699.pdf ,Ekim 2002
21. Luan J., White Papers, "Data Mining Applications In Higher Education", ftp://ftp.spss.com/pub/web/wp/DMHEWP-0702.pdf , Ekim 2002

22. SPSS Inc., White Paper, Technical Report, "An introduction to Clementine Server Distributed Architecture", ftp://ftp.spss.com/pub/web/wp/CLSDAWP0999.pdf , Ekim 2002
23. SPSS Inc., Clementine, "Discover Predictive Power In Your Data, CLM6BRO-0201W", 2001 Integral Solution Ltd., USA
24. SPSS Inc., Clementine For Public Sector, "Discover Predictive Power In Your Data, CLMPS6BRO-0401W", 2001 Integral Solution Ltd., USA
25. Lee N., Liao S.-C., Embrechts M., "Data Mining Techniques Applied To Medical Information", Medical Inf. And Internet In Medicine Vol. 25, 2000, No. 2, 81-102
26. Dodgi S., Marathe A. , RAVI S.S. Torney D.C. "Discovery of Association Rules In Medical Data", Medical Inf. And Internet In Medicine Vol. 26, 2001, No. 1, 25-33
27. Computer Science Innovations Inc., "Data Mining in Health Care", September 2001
28. Snook I.D., "Hospitals: What They Are And How They Work", 1981, Apsen System Corporation, USA
29. Demuth H., Beale M., "Neural Network Toolbox for use with Matlab", March 2001, Version 4, Release 12, The MathWorks Inc.
30. Keresteci E., Oktuğ F.S., Ersoy C., Çağlayan M.U., "Generation and Evaluation of Self Similar Traffic in Computer Networks", .Proc. BAS'97 The Second Symposium on Computer Networks, June 12-12, 1997, Ankara, pp 91-97.

ÖZGEÇMİŞ

5 Aralık 1972 tarihinde Kayseri'de doğdu. İlk, ortaokul ve lise eğitimini Antalya'nın Gazipaşa ilçesinde tamamladı. 1989 yılında başladığı Kayseri Meslek Yüksek Okulunu Elektronik Bölümü'nden 1991'de mezun oldu. 1992 yılında Fırat Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Öğretmenliği Bölümü'nü kazandı. 1994 yılında Marmara Üniversitesi'nin aynı bölümüne yatay geçiş yaptı. 1996 yılında bu bölümden mezun olduktan sonra aynı yıl Aydın Anadolu Teknik Lisesi'nde öğretmenliğe başladı. 1998 yılında başladığı askerlik görevini Balıkesir Astsubay Hazırlama ve Eğitim Komutanlığı'nda yedek subay olarak tamamladı. 2000 yılının Güz yarıyılında Akdeniz Üniversitesi Sağlık Bilimleri Enstitüsü Biyoistatistik Anabilim Dalı'nda Tıp Bilişimi yüksek lisans programına başladı. 2003 yılına kadar aynı anabilim dalında araştırma görevlisi olarak çalıştı.

EKLER

EK-1 : Oracle Veri Tiplerinin Clarion Programlama Dilindeki Karşılıkları

Oracle Veri Tipi	**Clarion Veri Tipi**
CHAR	STRING, CSTRING
VARCHAR2	STRING, CSTRING
NUMBER	REAL
NUMBER(n,p)	PDECIMAL
NUMBER(n,0)	BYTE, SHORT, USHORT, LONG
LONG	STRING+GROUP+SHORT
LONG RAW	STRING+GROUP+SHORT
DATE	DATE veya STRING+DATE+TIME
RAW	STRING
ROWID	STRING(18)

EK-2 : ASTAT.PRJ Proje dosyasının içeriği

```
-- astat
#noedit
#system win32
#model clarion dll
#pragma debug(vid=>full)
#pragma optimize(cpu=>386)
#compile "astat.clw"
#compile "astat01.clw"
#compile "astat02.clw"
#compile "astat03.clw"
#compile "astat04.clw"
#compile "astat05.clw"
#pragma link("C%V%ORA%X%%L%.LIB")
#link "astat.exe"
```

EK-3 : ASTAT.CLW Ana kaynak dosya içeriği

```
      PROGRAM

      INCLUDE('KEYCODES.CLW')


      MAP
       MODULE('astat01')
        AnaMenu
       END
       MODULE('astat02')
        HastaIslem
       END
       MODULE('astat03')
        TestListesi
       END
       MODULE('astat04')
        TestSorgulama
       END
       MODULE('astat05')
        ValidateOLE( SIGNED OleControl, <STRING OleCreateName>, <STRING
OleFileName>), BYTE
        stat1
       END
      END


!-------- Global Constants

!-------- Global Variables


!!!!!!!!!!TEST DOSYASI TANIMLAMASI!!!!!

Tests                FILE, DRIVER('ORACLE'), OWNER('serkan/serkan@sundb'),
NAME('MEDILIS.TESTS'), PRE(tes),BINDABLE
CodeKey          KEY(tes:CodeType, tes:TestNum), NAME('pk_TESTS'), PRIMARY
NameKey          KEY(tes:ShortName), DUP, NAME('ndx_TESTNAME')
InsKey           KEY(tes:ModelNum, tes:InsNum), DUP ,NAME('ndx_TESTINS')
LabKey           KEY(tes:LabNum), DUP,NAME('ndx_TESTLAB')
TubeKey          KEY(tes:TubeNum), DUP, NAME('ndx_TESTTUBE')
SpecKey          KEY(tes:SpecNum), DUP, NAME('ndx_TESTSPEC')
Record           RECORD
```

```
TestCode        GROUP
CodeType          STRING(@N02),NAME('CODETYPE')
TestNum           STRING(@N04),NAME('TESTNUM')
                END
Desc            CSTRING(51),NAME('TESTNAME')
ShortName         CSTRING(11),NAME('SHORTNAME')
InsCode         GROUP
ModelNum          STRING(@N02),NAME('MODELNUM')
InsNum          STRING(@N02),NAME('INSNUM')
                END
LabNum            SHORT,NAME('LABNUM')
TubeNum           SHORT, NAME('TUBENUM')
SpecNum           SHORT, NAME('SPECNUM')
LISunit         CSTRING(13), NAME('LISUNIT')
SIunit          CSTRING(13), NAME('SIUNIT')
SIFactor         PDECIMAL(13,6), NAME('SIFACTOR')
TestCnt          SHORT, NAME('VERSIONCNT')
MeasureCnt         SHORT, NAME('MEASURECNT')
ValueType         BYTE, NAME('VALUETYPE')
ActionType        BYTE, NAME('ACTIONTYPE')
LimitType        BYTE, NAME('LIMITTYPE')
nHighLimit        PDECIMAL(13,3), NAME('NHIGHLIMIT')
nLowLimit         PDECIMAL(13,3), NAME('NLOWLIMIT')
wHighLimit         PDECIMAL(13,3), NAME('WHIGHLIMIT')
wLowLimit         PDECIMAL(13,3), NAME('WLOWLIMIT')
SampVolume          PDECIMAL(13,6), NAME('SAMPVOLUME')          !
DateShift        SHORT, NAME('DATESHIFT')          !
Notes           CSTRING(200), NAME('NOTES')
              END
             END

!!!!!!!!!!!TEST DOSYASI SONU!!!!!!!!!!!

!!!!!!!!!!!HASTA DOSYASI TANIMLAMASI!!!

Hasta                       FILE,  DRIVER('Oracle'),  OWNER('serkan/serkan@sundb'),
NAME('MEDILIS.HASTA$'), PRE(pat), BINDABLE
CodeKey          KEY(pat:PatNum),    NAME('PK_HASTA')
NameKey           KEY(pat:LastName),  NAME('NDX_HASTASOYAD'),DUP
Record          RECORD
PatNum            PDECIMAL(8),   NAME('PATNUM')
FirstName          CSTRING(16),   NAME('FIRSTNAME')
LastName           CSTRING(21),   NAME('LASTNAME')
Sex             STRING(1),     NAME('SEX')
BirthDate          DATE,          NAME('BIRTHDATE')
```

```
KabulKodu          STRING(12),   NAME('KABULKODU')
                END
                END
!!!!!!!!!!HASTA DOSYASI SONU!!!!!!!!!

!!!ORDERRESULTS DOSYASI TANIMLANMASI!!

ORDERRESULTS              FILE,DRIVER('ORACLE'),  OWNER('serkan/x@sundb'),
NAME('MEDILIS.ORDERRESULTS'), PRE(ORE), BINDABLE
RECORD          RECORD
ORDERDATE                 DATE                NAME('ORDERDATE')
ORDERNUM                  SHORT               NAME('ORDERNUM')
WORKNUM                   SHORT               NAME('WORKNUM')
RESULTNUM                 SHORT               NAME('RESULTNUM')
CODETYPE                  STRING(2)           NAME('CODETYPE')
TESTNUM                   STRING(4)           NAME('TESTNUM')
LISALARMCODE              SHORT               NAME('LISALARMCODE')
CONFIRMED                 SHORT               NAME('CONFIRMED')
RESULTVALUE               CSTRING(50)         NAME('RESULTVALUE')
INSALARMCODE              CSTRING(10)         NAME('INSALARMCODE')
                END
               END
!!!!!!ORDERRESULTS DOSYASI SONU!!!!!!!

!!!!!!ORDERS DOSYASI TANIMLAMASI!!!!!!

Orders                    FILE,  DRIVER('Oracle'),  OWNER('serkan/serkan@sundb'),
NAME('MEDILIS.ORDERS'), PRE(ord), BINDABLE    ! CREATE,
OrderKey                KEY(ord:OrderDate, ord:OrderNum), NAME('PK_MORDERS'),
PRIMARY
HISKey            KEY(ord:HISOrderCode)          , NAME('NDX_HISORDERCODE')
PatientKey        KEY(ord:PatNum)                , NAME('NDX_ORDPATIENT'), DUP
!ServiceKey                          KEY(ord:SerType,    ord:SerNum)                 ,
NAME('NDX_ORDSERVICE'), DUP
Record            RECORD
Order_Code          GROUP
OrderDate           DATE,       NAME('ORDERDATE')
OrderNum            PDECIMAL(4),  NAME('ORDERNUM')
                END
HISOrderCode        STRING(16),   NAME('HISORDERCODE')
WorkDate            DATE,         NAME('WORKDATE')
PatNum              LONG,         NAME('PATNUM')
SerCode             GROUP
SerType             STRING(1),    NAME('SERTYPE')
SerNum              STRING(@N04), NAME('SERNUM')
```

```
                END
OrderType       SHORT,      NAME('ORDERTYPE')
OrderStat       BYTE,       NAME('ORDERSTAT')
DoctorNum       STRING(8),  NAME('DOKTOR')
              END
             END
!!!!!!!!!ORDERS DOSYASI SONU!!!!!!!!!!

!!!!!!TABLO DOSYALARINA BAĞLANTI!!!!!!
  CODE
 OPEN (TESTS)
 IF ERROR ()
  MESSAGE (ERROR())
 END
 OPEN (HASTA)
 IF ERROR ()
  MESSAGE (ERROR())
 END
 OPEN (ORDERRESULTS)
 IF ERROR ()
  MESSAGE (ERROR())
 END
 OPEN (ORDERS)
 IF ERROR ()
  MESSAGE (ERROR())
 END

 ORDERS{PROP:SQL}      =      'ALTER      SESSION      SET
NLS_DATE_FORMAT="DD/MM/YYYY"'

 ANAMENU
!!TABLO DOSYALARINA BAĞLANTI SONU!!!
```

EK-4 : ASTAT1.CLW Dosyası içeriği

```
MEMBER('astat')
!!!!!!!!!!!!!!APPLICATION FRAME PENCERE TANIMLAMASI!!!!!!!!!!!!
AnaMenu                PROCEDURE
AppFrame     APPLICATION('Medistat     sistemi'),AT(,,301,184),FONT('MS     Sans
Serif',8,,FONT:regular),SYSTEM,MAXIMIZE, |
        DOUBLE
      MENUBAR
        MENU('&Sistem')
          ITEM('Yazıcı Ayarları'),USE(?mPrinterSetup),STD(STD:PrintSetup)
          ITEM,SEPARATOR
          ITEM('Çıkış'),USE(?mExit),STD(STD:Close)
        END
        MENU('&Testler')
          ITEM('Sorgulama'),USE(?MQuery)
          ITEM('İşlemler'),USE(?mActions)
        END
        MENU('&Rapor')
          ITEM('Test Listesi'),USE(?mRapor)
        END
        MENU('&Yardım')
        END
      END
      TOOLBAR,AT(0,0,,25)
        BUTTON,AT(5,2,18,18),USE(?bExit),FLAT,TIP('Uygulamadan
Çıkış'),ICON(ICON:Cross)
      END
    END
!!!!!!!!!!!!!!APPLICATION FRAME PENCERE TANIMLAMASI!!!!!!!!!!!!
!!!!!!!!!!!!!!APPLICATION FRAME PENCERE İŞLEMLERİ!!!!!!!!!!!!!!
  CODE
    OPEN(AppFrame)
      ACCEPT
        CASE ACCEPTED()
          OF ?bExit
            POST(EVENT:CloseWindow)
          OF ?mActions
            HastaIslem       !takip02'de
          OF ?mRAPOR
            TestListesi      !takip03'de
          OF ?mQuery
            TestSorgulama    !takip04'de
        END
      END
  RETURN
```

EK-5 : ASTAT2.CLW Dosyası içeriği

```
  MEMBER('astat')


HastaIslem     PROCEDURE


HASTAG     GROUP
DOSYANO        LONG
AD           STRING(15)
SOYAD          STRING(30)
DOGTAR         LONG
CINS         STRING(1)
KAN          STRING(4)
        END

!!!!!!!!!!HASTA İŞLEMLERİ MENÜSÜNÜN PENCERE TANIMLAMASI!!!!!!!!!!

Window         WINDOW('Hasta          İşlemler'),AT(,,272,114),FONT('MS          Sans
Serif',8,,FONT:regular),GRAY,RESIZE
     PROMPT('Dosya No:'),AT(24,7,52,10),USE(?Prompt1),RIGHT
     ENTRY(@N06),AT(81,7,60,10),USE(HASTAG:DOSYANO),LEFT,INS

ENTRY(@s20),AT(81,20,60,10),USE(HASTAG:AD),LEFT,CURSOR(CURSOR:IBea
m),REQ
     PROMPT('Ad:'),AT(65,22),USE(?Prompt2),RIGHT
     ENTRY(@s20),AT(81,35,86,10),USE(HASTAG:SOYAD),LEFT,UPR
     PROMPT('Doğum Tarihi:'),AT(-6,50,82,10),USE(?Prompt4),RIGHT
     ENTRY(@D8),AT(81,49,86,10),USE(HASTAG:DOGTAR)
     PROMPT('Kan Grubu:'),AT(19,92,57,10),USE(?Prompt5),RIGHT

COMBO(@s10),AT(85,91,114,10),USE(HASTAG:KAN),DROP(10),FROM('ARh+|A
Rh-|BRh+|BRh-|ABRh+|ABRh-|0Rh+|0Rh-')
     PROMPT('Soyad:'),AT(43,37,33,10),USE(?Prompt3),RIGHT
     OPTION('Cinsiyet'),AT(83,62,113,25),USE(HastaG:Cins),BOXED
       RADIO('Erkek'),AT(97,73),USE(?Option1:Radio1),VALUE('E')
       RADIO('Kadın'),AT(139,73,35,10),USE(?Option1:Radio2),VALUE('K')
     END
     BUTTON('Ekle'),AT(223,25,36,14),USE(?add)
     BUTTON('Düzelt'),AT(223,46,36,14),USE(?put)
     BUTTON('Sil'),AT(223,68,36,14),USE(?delete)
     !BUTTON('OK'),AT(222,66,35,14),USE(?OkButton),DEFAULT
     BUTTON('Cancel'),AT(223,90,36,14),USE(?CancelButton)
```

```
    END
!!!!!!HASTA İŞLEMLERİ MENÜSÜNÜN PENCERE TANIMLAMASI SONU!!!!!!

!!!!!!!!!!HASTA İŞLEMLERİ MENÜSÜNÜN KOD TANIMLAMALARI!!!!!!!!!

 CODE

open(window)
accept

  case event()
  end

  case accepted()

  of ?cancelbutton
  POST(EVENT:CloseWindow)
 end

end

 RETURN
```

EK-6 : ASTAT4.CLW Dosyası içeriği

```
  MEMBER('astat')

!!!!!!!!!!TEST SORGULAMA PROSEDÜRÜ DEĞİŞKEN TANIMLAMALARI!!!!!!!!

TestSorgulama   PROCEDURE

TESTSELQ   QUEUE
TESTADI    STRING(40)
TESTSECIM  BYTE
TESTCODE   STRING(6)
        end

INLIST     STRING(50)

TESTQ      QUEUE
SIRANO     LONG
OCODE      STRING(12)
DOSYANO    LONG
SEX        STRING(1)
YAS        LONG
TCODE      STRING(6)
!TISIM     STRING(30)
!TESTADI   STRING(40)
TSONUC     LONG
        END

QC         QUEUE
STATDESC   STRING(50)
TREELEVEL  SHORT
        END

YAS        QUEUE
YASGRUBU   STRING(20)
        END

SYQ        QUEUE
SSIRANO    LONG
SDOSYANO   LONG
SSEX       STRING(1)
SYAS       LONG
SORDERCODE STRING(12)
ST1        REAL
ST2        REAL
```

```
ST3       REAL
ST4       REAL
ST5       REAL
ST6       REAL
        END

SELQ      QUEUE
TESTCODE  STRING(6)
        END

KLINIKQ   QUEUE
KLINIKADI STRING(15)
        END

COMBOC    STRING(20)
SAD       STRING(30)
DT        LONG
SECIM     STRING(25)
COMBOYAS  STRING(20)

BASTAR    LONG
BITTAR    LONG
E         BYTE
K         BYTE
TESTCODE  LONG
OC        STRING(12)
CINSSEC   STRING(1)

!!!!!TEST SORGULAMA PROSEDÜRÜ DEĞİŞKEN TANIMLAMALARI SONU!!!!!

!!!!!!!!!!!!!TEST SORGULAMA PENCERESİ TANIMLAMASI!!!!!!!!!!!!!!

Window    WINDOW('Test Sorgulama'),AT(,,409,281),FONT('MS Sans
Serif',8,,FONT:regular),GRAY,RESIZE
    PANEL,AT(21,2,370,41),USE(?Panel1),BEVEL(-1)
    STRING('TESTLERİN UYGULANACAĞI TARİH ARALIĞI CİNSİYET VE
KLİNİK SEÇİMİ'),AT(31,3,349,10),USE(?String3), |
      CENTER
    OPTION('Cinsiyet'),AT(157,13,70,21),USE(cinssec),BOXED
     RADIO('E'),AT(169,22,21,10),USE(?cinssec:Radio1),VALUE('E')
     RADIO('K'),AT(197,22),USE(?cinssec:Radio2),VALUE('K')
    END
    STRING('Başlangıç'),AT(42,16,31,10),USE(?bas),CENTER
    STRING('Bitiş'),AT(43,28,31,10),USE(?bit),LEFT
    STRING('KLİNİK'),AT(237,20,26,11),USE(?String5)
```

```
    ENTRY(@D6),AT(74,16,57,10),USE(bastar),REQ,OVR
    ENTRY(@D6),AT(74,28,57,10),USE(bittar),REQ,OVR
    LIST,AT(268,20,110,11),USE(?klinik),DROP(10),FROM(KLINIKQ)

LIST,AT(21,54,370,50),USE(?testad),SCROLL,VSCROLL,ALRT(MouseRight),FORM
AT('197L(2)|M~Test                    Adı~C(0)@s40@#1#49L(2)|M~Seçim
Durumu~C(1)@s1@#2#49L(2)|M~TestCod' &|
       'u~C(1)@s6@'),FROM(TESTSELQ)

LIST,AT(21,117,370,77),USE(?TESTSORGU),HVSCROLL,FORMAT('34L(2)|M~Sıra
No~C@n5@42L(2)|M~Test                    Kodu~C@s6@45L(2)|M~Dosya
No~C@s10@45L(2)|M~C' &|
       'insiyet~C@s10@45L(2)|M~Yaş~C@D06-@73L(2)|M~Test Tip ve
Numarası~C@s10@249R(2)|M~' &|
       'Test Sonucu~@N_8.2@'),FROM(TESTQ)

COMBO(@s30),AT(22,267,120,10),USE(?Combo1),HVSCROLL,FORMAT('20L|MT
~Yapmak istediğiniz istatistik türünü seçin~@s20@'), |
       DROP(5),FROM(qc)
    BUTTON('Sorgula'),AT(203,266,35,14),USE(?Sorgula)

LIST,AT(21,209,370,50),USE(?List3),FORMAT('51L(2)|M@s12@46L(2)|M@N_9.2@
37L(2)|M@N_9.2@34L(2)|M@N_9.2@36L(2)|M@N_9.2@37L(2)|' &|
       'M@N_9.2@20L(2)|M@N_9.2@'),FROM(syq)
    BUTTON('Çıkış'),AT(314,266,35,14),USE(?bCancel)
    BUTTON('Test Listesi'),AT(21,45,370,10),USE(?TestList)
    BUTTON('Seçilen Özelliklere Göre Yeni Tablo'),AT(21,107,370,10),USE(?Tablo)
  END

!!!!!TEST SORGULAMA PENCERESİ KOD GİRİŞİ!!!!!!

 CODE
 bastar = TODAY()
 bittar = TODAY()

!!!!!!!!!QC QUEUE'SUNA UYGULANABİLİR TESTLERİN AKTARILMASI!!!!!!!

    QC.STATDESC = 'Descriptive Frequencies'
   ADD(QC)
    QC.STATDESC = 'Descriptive Explore'
   ADD(QC)
    QC.STATDESC = 'Descriptive Crosstabs'
   ADD(QC)
    QC.STATDESC = 'Compare Means-Independent Samples T-Test'
   ADD(QC)
```

```
 QC.STATDESC = 'Compare Means-Paired Samples T-Test'
ADD(QC)
 QC.STATDESC = 'Compare Means-ANOVA'
ADD(QC)
 QC.STATDESC = 'Correlate-Bivariate'
ADD(QC)
 QC.STATDESC = 'Regression-Linear'
ADD(QC)
 QC.STATDESC = 'Regression-Binary Logistic'
ADD(QC)
 QC.STATDESC = 'Non-parametric-Chi Square'
ADD(QC)
 QC.STATDESC = 'Non-parametric-2 Independent Samples'
ADD(QC)
 QC.STATDESC = 'Non-parametric-K Independent Samples'
ADD(QC)
 QC.STATDESC = 'Non-parametric-2 Related Samples'
ADD(QC)
 QC.STATDESC = 'Non-parametric-K Related Samples'
ADD(QC)

!!!!!!!!!QC QUEUE'SUNA UYGULANABİLİR TESTLERİN AKTARILMASI
SONU!!!!!!!

!!!!!!!!YAS QUEUE'SUNA YAS GRUPLARININ AKTARILMASI!!!!!!!!!
!
!    YAS.YASGRUBU = 'HEPSİ'
!   ADD(YAS)
!    YAS.YASGRUBU = '0-1 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '0-3 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '0-15 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '0-18 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '5-12 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '5-15 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '12-18 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '18-25 YAŞ'
!   ADD(YAS)
!    YAS.YASGRUBU = '18-36 YAŞ'
```

```
!    ADD(YAS)
!     YAS.YASGRUBU = '25-40 YAŞ'
!    ADD(YAS)
!     YAS.YASGRUBU = '30-45 YAŞ'
!    ADD(YAS)
!     YAS.YASGRUBU = '30-60 YAŞ'
!    ADD(YAS)
!     YAS.YASGRUBU = '45-60 YAŞ'
!    ADD(YAS)
!     YAS.YASGRUBU = '20 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!     YAS.YASGRUBU = '30 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!     YAS.YASGRUBU = '35 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!     YAS.YASGRUBU = '40 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!     YAS.YASGRUBU = '45 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!     YAS.YASGRUBU = '50 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!     YAS.YASGRUBU = '55 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!     YAS.YASGRUBU = '60 YAŞINDAN BÜYÜKLER'
!    ADD(YAS)
!
!!!!!!YAS QUEUE'SUNA YAS GRUPLARININ AKTARILMASI SONU!!!!!!!

!!!!!!!!TEST SORGULAMA PENCERESİ İŞLEMLERİ!!!!!!!!

 OPEN(window)
  ACCEPT
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    CASE FIELD()
!!!!!!!!!CANCEL BUTONU!!!!!!!!!!!!!!!!
      OF ?BCANCEL
      CASE EVENT()
      OF event:accepted
        POST(event:closewindow)
      END

!!!!!TEST SEÇİM LİSTESİNİN OLUŞTURULMASI!!!!!!!
      OF ?TESTLIST
      CASE EVENT()
      OF event:accepted
```

```
        DO TestSorgu
      END

!!!!!!TESTLERİN TESTAD LİSTESİNDEN SEÇİMİ!!!!!!
      OF ?TESTAD
      CASE EVENT()
      OF event:newselection
        get(TESTSELQ, CHOICE(?Testad))
       of event:alertkey
        if keycode()=mouseright
          if  TESTSELQ.TESTSECIM = 1
            TESTSELQ.TESTSECIM = 0
          else
            TESTSELQ.TESTSECIM = 1
          end
         put(TESTSELQ)
        end
      END
!!!!SPSS İSTATİSTİKLERİNİN SEÇİLMESİ!!!!!!!!!!!
      OF  ?combo1
        CASE EVENT()
      OF EVENT:NEWSELECTION
        comboc = Choice (?combo1)
          GET (qc, comboc)
          DO RCOMBO
      END
!!!!!!!!!!YENİ TABLONUN OLUŞTURULMASI!!!!!!!!!
      OF ?TABLO
      CASE EVENT()
      OF event:accepted
        DO TabloYap
      END
   END
  END

!!!MEVCUT TESTLERİN TESTSELQ LİSTESİNE DOLDURULMASI TEST
SORGULAMA RUTİNİ!!!
TestSorgu      ROUTINE
      FREE(TESTSELQ)
        TESTS{PROP:SQL}  =  'SELECT  *  FROM  MEDILIS.TESTS  WHERE
CODETYPE="10"'
        setcursor(cursor:wait)
          LOOP
            NEXT(TESTS)
            IF ERROR() THEN BREAK.
```

```
            testselq:testadi = tes:Desc
            testselq:testcode = tes:TestCode
            testselq:testsecim=0
            ADD(testselq)
          END
        setcursor()
!!SPSS İSTATİSTİK COMBOSUNDAN TEST SEÇİMİ!!!!!
RCOMBO   ROUTINE
      CASE comboc
      of 1
            stat1
!           of 2
!           stat2
!           of 3
!           stat3
!           of 4
!           stat4
!           of 5
!           stat5
!           of 6
!           stat6
!           of 7
!           stat7
!           of 8
!           stat8
!           of 9
!           stat9
!           of 10
!           stat10
        END
!!!!!!!!!!!!SEÇİLEN KRİTERLERE GÖRE YENİ TABLONUN
OLUŞTURULMASI!!!!!!!!!!!!

TabloYap    ROUTINE

      FREE(TESTQ)
      CLEAR(TESTQ)
      FREE(SELQ)
      inlist = ''
      LOOP i# = 1 TO RECORDS(testSelQ)
       GET(testSelQ, i#)
       IF testselq.testsecim=1
        inlist = CLIP(inlist) & '''' & CLIP(testSelQ.TestCode[3:6]) & '''' & ','
        SELQ:TestCode = testSelQ.TestCode
        ADD(SELQ)
```

```
        END
       END
       inlist = inlist[1 : LEN(CLIP(inlist))-1]

       setcursor(cursor:wait)
       ORDERRESULTS{PROP:SQL}      =      'SELECT      *      FROM
MEDILIS.ORDERRESULTS WHERE' |
       & ' ORDERDATE >='" & FORMAT(bastar,@D06-)   & "" & ' AND
ORDERDATE <='" & FORMAT(bittar,@D06-) & "" |
       & ' AND TESTNUM IN (' & CLIP(inlist) & ')'
      !setclipboard(ORDERRESULTS{PROP:SQL})
      LOOP
       NEXT(OrderResults)
       Orders{PROP:Sql} = 'SELECT * FROM MEDILIS.ORDERS WHERE' |
       & ' ORDERDATE ='" & FORMAT(oRe:OrderDate,@D06-) & "" & ' AND
ORDERNUM = ' & ore:OrderNum
       NEXT(Orders)
       IF NOT ERROR()
        !
        ! klinik filtresi
        !
        Hasta{PROP:Sql} = 'SELECT * FROM MEDILIS.HASTA$ WHERE
PATNUM=' & ord:PatNum
        NEXT(Hasta)
        IF NOT ERROR()
         IF cinssec
          IF pat:Sex <> cinssec THEN CYCLE.
         END
         testq:SiraNo += 1
         testq:ocode = FORMAT(oRe:OrderDate,@D12) & FORMAT(oRe:OrderNum,
@N04)
         testq:dosyano = ord:patnum
         testq:sex = pat:Sex
         testq:yas = pat:Birthdate
         testq:tCODE  = ore:codeType & ore:TestNum
         testq:tSonuc = LEFT(oRe:ResultValue)
         ADD(TESTQ)
        END
       END
      END
      SETCURSOR()

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      OC = ''
      SORT(TestQ, testq:oCode, testq:tCODE)
```

```
LOOP i# = 1 to records(TestQ)
 GET(testQ, i#)
 IF testq:oCode <> OC
  IF OC <> ''
   add(SYQ)
  END
  OC = testq:oCode
  SYQ.SORDERCODE = testq:oCode
 END
 LOOP j# = 1 TO RECORDS(SelQ)
   GET(SelQ, j#)
   IF testq:TCode = selq:testCode
    CASE j#
    OF 1
     SYQ:ST1 = testq:tSonuc
    OF 2
     SYQ:ST2 = testq:tSonuc
    OF 3
     SYQ:ST3 = testq:tSonuc
    OF 4
     SYQ:ST4 = testq:tSonuc
    OF 5
     SYQ:ST5 = testq:tSonuc
    OF 6
     SYQ:ST6 = testq:tSonuc
    END
   END
 END
END
```

!!!!!!!!!!!!!!!!!!!!MODÜLÜN SONU!!!!!!!!!!!!!!!!!!!!!!!!!!!!

EK-7 : ASTAT5.CLW Dosyası içeriği

```
   MEMBER('astat')

STAT1     PROCEDURE

strFileName string(255)
strCommand  string(255)

!!!!!!!!!!SPSS SYNTAX KOMUT DİZİSİ PENCERESİ!!!!!!!!!

Window              WINDOW('Caption'),AT(,,185,92),FONT('MS              Sans
Serif',8,,FONT:regular),GRAY
     OLE,AT(71,9,31,13),USE(?SPSS),HIDE
     END
     BUTTON('Cancel'),AT(144,47,36,14),USE(?EXIT)
     BUTTON('Syntax'),AT(144,28,36,14),USE(?syntax)
     TEXT,AT(12,35,120,48),USE(strCommand)
    END

  CODE
  OPEN(wINDOW)
  ACCEPT
   CASE EVENT()
   OF EVENT:OpenWindow
!!!!!!!!!!!!SPSS'İ BAŞLATMA!!!!!!!!!!

    IF ~ValidateOLE(?SPSS, 'SPSS.Application')
     POST(EVENT:CloseWindow)
    END

!!!!!!!!!!VERİ DOSYASINI AÇMA!!!!!!!!!!

    strFileName = CLIP(?SPSS{'GetSPSSPath'}) & 'employee data.sav'
    strFileName = 'OpenDocument(' & CLIP(strFileName) & ',1)'
    ?SPSS{strFileName}

!!!!!!!!!KOMUT SYNTAX'INI KULLANARAK DESCRIPTIVE PROSEDÜRÜN
ÇALIŞTIRILMASI!!!!!!!!!
     strCommand = 'DESCRIPTIVES'
     strCommand = CLIP(strCommand) & ' VARIABLES=salary salbegin'
     strCommand = CLIP(strCommand) & ' /STATISTICS=MEAN STDDEV MIN
MAX .'
     ?SPSS{'ExecuteCommands(' & CLIP(strCommand) & ',1)'}
```

```
  OF EVENT:CloseWindow
  END

  CASE FIELD()

  OF ?syntax
   CASE EVENT()
   OF EVENT:ACCEPTED
    strFileName = CLIP(?SPSS{'GetSPSSPath'}) & 'anestezi2.sps'
    strFileName = 'ExecuteInclude(' & CLIP(strFileName) & ',0)'
    ?SPSS{ strFileName }
   END

  OF ?Exit
   CASE EVENT()
   OF EVENT:ACCEPTED
    ?SPSS{PROP:DEACTIVATE} = 1
    POST(EVENT:CloseWindow)
   END
  END
 END

!=============================================================================
=============================================================================
ValidateOLE                    PROCEDURE( SIGNED OleControl, <STRING
OleCreateName>, <STRING OleFileName>)
!=============================================================================
=============================================================================
lCallRet            LONG

 CODE
 IF OleControl{PROP:OLE} = FALSE AND OleControl{PROP:Object} = ''
  IF ~OMITTED( 2 )
   OleControl{PROP:Create} = CLIP( OleCreateName )
   IF OleControl{PROP:OLE} = FALSE AND OleControl{PROP:Object} = ''
    IF ~OMITTED( 3 )
     lCallRet = CALL( CLIP( OleFileName ),'DllRegisterServer' )
     IF lCallRet
      MESSAGE('Can not Register OLE Control ' & CLIP( OleFileName ) & ' Error: '
& CLIP( lCallRet ) & '.  Please re-install Legal Files.','Fatal OLE
Error',ICON:Exclamation)
      RETURN FALSE
     END
     UNLOAD( CLIP( OleFileName ) )
     OleControl{PROP:Create} = CLIP( OleCreateName )
```

```
      ELSE
        RETURN FALSE
      END
      IF OleControl{PROP:OLE} = FALSE AND OleControl{PROP:Object} = ''
        MESSAGE('Can not Find OLE Control ' & CLIP( OleFileName ) & ' !  Please re-
install.','Fatal OLE Error',ICON:Exclamation)
        RETURN FALSE
      ELSE
        RETURN TRUE
      END
    ELSE
      RETURN TRUE
    END
  ELSE
    RETURN FALSE
  END
 ELSE
  RETURN TRUE
 END
```