T.C.
İNÖNÜ ÜNİVERSİTESİ
SAĞLIK BİLİMLERİ ENSTİTÜSÜ

TURGUT ÖZAL TIP MERKEZİ HASTANE OTOMASYONUNDAKİ LABORATUAR SONUÇLARININ SPSS PAKET PROGRAMINA AKTARILABİLMESİ İÇİN ARA YÜZ YAZILIMI GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Emek GÜLDOĞAN
BİYOİSTATİSTİK ANABİLİM DALI

DANIŞMAN
Doç. Dr. Saim YOLOĞLU

MALATYA-2005

Sağlık Bilimleri Enstitüsü Müdürlüğüne

Emek GÜLDOĞAN'a ait bu bilimsel çalışma jüri üyeleri olarak tarafımızdan Biyoistatistik Anabilim Dalında **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

İMZA

Başkan Doç.Dr.Saim YOLOĞLU

Üye Prof.Dr.Tayfun GÜLDÜR

Üye Yrd.Doç.Dr.Mustafa KARAKAPLAN

Yukarıda imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

...9.../...9.../ 2005

Prof.Dr.Tayfun GÜLDÜR
Enstitü Müdürü

TEŞEKKÜR

Bu tezin konu seçimi ve yönlendirmelerindeki yardımlarından dolayı tez danışmanım Doç. Dr. Saim Yoloğlu'na teşekkür ederim. Bu tezin hazırlanması ve deneysel çalışmaların yapılması sırasında destek ve yardımlarını gördüğüm Simay Güçlüer, Yılmaz Akıncı ve Cengiz Karaduman'a teşekkür etmeyi bir borç bilirim.

İçindekiler

Şekiller Dizini

Simgeler ve Kısaltmalar

TOTM	Turgut Özal Tıp Merkezi
VM	Veri Madenciliği
VTYS	Veri Tabanı Yönetim Sistemi
VTBK	Veri Tabanlarında Bilgi Keşfi
XML	Extensible Markup Language
GIS	Geographical Information System
EHK	Elektronik Hasta Kayıt
SQL	Structered Query Language
SPSS	Statistical Package for Social Science
PHP	Personal Home Page
HTTP	Hyper Text Tarnsfer Protocol
HL7	Health Level Seven
POMR	Problem Yönelimli Tıbbi Kayıt
SOAP	Subjective, Objective, Assestment, Plan

1 GİRİŞ

Son yirmi yılda bilgi artışıyla beraber bu bilgileri toplama ve saklama konusunda çok büyük bir büyümeye şahit olmaktayız. Var olan bilgiler yanında her yıl üretilen bilgiler bilgisayarların işleyebileceğinden oldukça fazladır. İnsanoğlunun ürettiği bilgi her 20 ayda ikiye katlanmaktadır. Bu bilgi birikimimizi eş zamanlı olarak yorumlamak, özümsemek ve geleceğe yön vermek için kullanmak gereklidir. Bu nedenle daha akıllı ve devinimli veri saklama ve analizi için yeni nesil araçlara ihtiyaç doğdu. Bilginin analizi için kullanılacak araçlar ne kadar önemli ise bu bilgi uygun olarak saklayacak araçlar da o kadar önemlidir. Gelişen dünyada veri tabanlarından bilgi keşfi aktif bir araştırma alanına evrimleşti. Veri Madenciliği her araştırma alanında önemli bir yere sahip oldu. Bilgi artışındaki en büyük sıçrama sağlık biliminde yaşanmıştır. Biyoinformatik gibi veri madenciliğine dayanan bir bilim dalı doğdu. Gen teknolojisi tamamen çok büyük miktardaki verinin depolanması ve bunun işlenmesi ile gelişmiştir [1].

Bilişim sistemlerinin Türkiye'de sağlık bilimlerinde kullanılmaya başlanması son 15 yıl içerisinde gerçekleşmiştir. Devlet hastaneleri, üniversiteler, özel hastaneler ve sağlık kuruluşları geçen bu süreçte kendi olanaklarıyla bünyelerinde sağlık bilişim sistemleri oluşturmaya çalışmışlardır.Yapılan bu çalışmalara rağmen ülkemizde halen ulasal bir sağlık bilgi sistemi oluşturulamamıştır.Ülkemizde şu süreçte eczanelerde Bağkur, SSK, Emekli Sandığı ve TSK otomasyon sistemleri kullanılmaktadır.Bu sayede bu kurumlara ait hastaların kullandıkları ilaç bilgileri veritabanlarında tutulmaktadır. İlerleyen süreçte

hastaların kullandıkları ilaçlara ait bilgilerin yanı sıra hastalık teşhisleri , laboratuar sonuçları bilgileride veritabanlarında tutulacaktır.

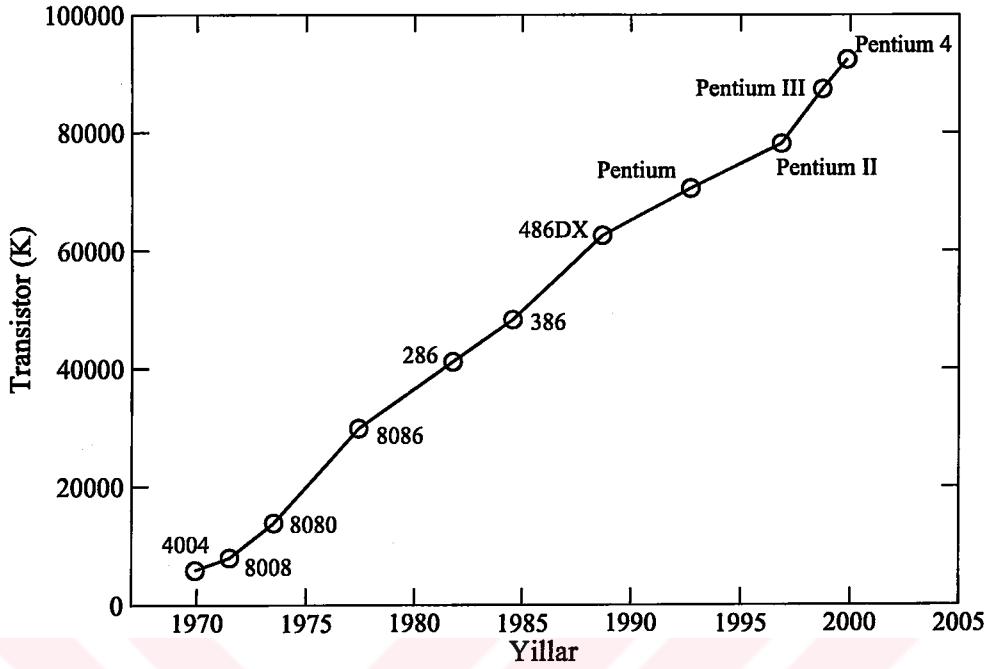
1.1 GENEL BİLGİLER

1.2 Bilgi Teknolojileri

Hızla artış gösteren bilgiyi toplamak ve işlemek için bilgisayar ve yardımcı araçlar bir çok alanda kullanılmaya başlandı. Artan bilgi karşısında daha zeki veri analiz sistemleri de artış göstermiştir [2]. Büyük oranda veri birikimi bilgisayarların kapasitesinin üzerinde yer almaktadır. Teknolojik gelişmeler ile beraber gelişen veri madenciliği geçmişe ait veriler üzerinde dahi yeniden sonuç çıkarmak için uygulama alanı bulmaktadır. Verilerin güvenirliliğine, düzenli toplanıp toplanmadığına bakılmaksızın bu veriler üzerinde yeniden istatistik çalışmalar yapılabilmektedir [3].

1965 yılında Intel'in kurucularından Gordon Moore'un ortaya attığı Moore Yasası'na göre işlemcilerdeki transistor sayısı 18 ayda bir ikiye katlanır. Moore, bu yasanın sonraki on yıl boyunca geçerliliğini koruyacağını tahmin etmişti ama Intel bu yasayı günümüze kadar çiğnemeden devam ettirmeyi aşağıdaki grafikte de görebileceğiniz gibi başardı [4].

Bilgi teknolojilerindeki gelişme, bilgisayarların ve otomatik veri toplama araçlarının geniş bir alanda uygulanmasını sağlamıştır. Yaygın bilgisayar kullanımı sonucunda, çeşitli ortamlarda ve/veya biçimlerde çok büyük ölçekli işletimsel veri birikmiştir. Veri saklama kapasitesi her 9 ayda bir iki kat artmakta buna karşılık ise, aynı periyotta, Moore kanununa göre hesaplama gücü iki kat daha az büyümektedir. Bir çok veri bir daha erişilmeyecek veya işlenmeyecek şekilde saklanmaktadır. Saklanan verinin tekrar kullanılabilmesi ve değerini yitirmemesi için yeni standartlar ve teknolojiler geliştirilmektedir. Veri tabanı sistemleri olarak adlandırılan bu teknoloji bilişim teknolojilerinde



Şekil 1.1: Moore Yasası

ön sıralarda yer almaktadır.

Gerek bilimsel veritabanlarında gerekse de günlük iş aktiviteleri etrafında modellenmiş ticari veritabanlarında bu çok büyük miktardaki verilerin analizi veri tabanı uzmanlarının kapasitesini çoktan aşmıştır. Bu nedenle gerçek hayat verilerinin otomatik veya yarı otomatik tekniklerle kullanıcı açısından ilginç ve önemli bilgilere dönüştürülmesi gerekmektedir. Verilerin analizi ve işlenmesi bugünün veritabanı yönetim sistemlerinin tipik işlevleriyle gerçekleştirmek olanaksızdır. Bugünün veri tabanları bilinen veri tiplerine (örneğin, ad/soyad, tarih, dosya no) göre ayarlanmıştır. Bunun dışında GIS (Geographical Information Systems) amaçlı veri tabanı sistemlerinde daha kompleks veri tipleri saklanabilmektedir. Veri tabanlarında saklanacak veri çeşitliliğinin çok fazla oluşu saklanacak verinin de bir ön işlemden geçirilip kodlanmasını gerektirmektedir. XML (Extended Markup Language) verinin kodlanması için geliştirilmiş bir dildir. XML ile saklanan verinin değerini yitirmeden tekrar

kullanabilmesi sağlanmaktadır. Verilerin saklanması ve güvenliği de diğer taraftan göz önünde bulundurulması gereken bir durumdur. Verilerin bağlı bulunduğu alana göre veri işleme ve saklama yöntemleri de değişiklik göstermektedir. Sosyal ve politik alanlarda bilgi kaydetmeye yönelik hazır uygulamalar geliştirilmiştir [5]. Veri tabanı verinin saklanması dışında aşağıdaki işlemleri gerçekleştirmektedir.

- Veri ekleme veya güncelleme ya gerçekleşti yada gerçekleşmedi işleminin kontrolü
- Önceki tutarlılığın oturum sonucunda korunabilmesi.
- Veriler üzerinde eş zamanlı olarak paralel çalışabilmesi.
- Birbirinden izole işlemler gerçekleştirmek.
- Verilerin sorgulanması sırasında temel işlemler gerçekleştirmek
- Veri sorgulamalarında karar verme mekanizmaları sağlamak.

1.3 Veri Tabanlarında Bilgi Keşfi (VTBK)

Literatürde, işletimsel veri içinden faydalı örüntülerin bulunması işlemine bir çok terim bulunmuş ve bunlardan ikisi çok kullanılmaktadır [6]. Veri madenciliği (VM) (data mining), bilgi harmanlama (information harvesting) veri işlemede en çok kullanılan terimlerdir [7].

Veri madenciliği (VM) terimi sosyal bilimler dahil teknik ve iş dünyası gibi bir çok alanda bilgi keşfi metotlarıyla uğraşan bir alandır ve aşağıdaki adımları içerir [5].

- Veri Seçimi (Data Selection): Bu adım birkaç veri kümesini birleştirerek, sorguya ve elde edilecek bilgiye uygun veri kümesini elde etmeyi içine alır.

- Veri Temizleme ve Önişleme (Data Cleaning & Preprocessing): Seçilen veri alanında yer alan hatalı girişlerin çıkarıldığı ve eksik ve uygun olmayan nitelendirme değerlerinin değiştirildiği aşamadır. Bu aşama ortaya çıkacak bilginin kalitesini ve güvenirliliğini arttırır.

- Veri İndirgeme (Data Reduction): Seçilen veri alanında gereksiz ve uygun olmayan niteliklerin atıldığı ve tekrarlı kayıtların ayıklandığı adımdır. Bu aşama seçilen veri madenciliği sorgusunun çalışma zamanını iyileştirir ve sorgu sırasında ortaya çıkabilecek hataları engeller.

- Veri Madenciliği (Data Mining): Verilen bir veri madenciliği sorgusunun (sınıflama, güdümsüz öbekleme, eşleştirme, vb.) işletilmesidir.

- Değerlendirme (Evaluation): Keşfedilen bilginin geçerlilik, yenilik, yararlılık ve basitlik kıstaslarına göre değerlendirilmesi aşamasıdır. Veriler istatiksel olarak değerlendirmeye alınır.

Aktif araştırma alanlarından biri olan veri tabanlarında bilgi keşfi alanı (VTBK) çok büyük hacimli verileri tam ya da yarı otomatik bir biçimde analiz eden yeni kuşak araç ve tekniklerin üretilmesi ile ilgilenen son yılların gözde araştırma konularından biridir. VTBK, veri seçimi, veri temizleme ve ön işleme, veri indirgeme, veri görüntüleme, korelasyon analizi, yapay sinir ağlarının da kullanıldığı veri madenciliği ve değerlendirme aşamalarından oluşan bir süreçtir [8], [9]. Veri Madenciliği (VM), önceden bilinmeyen, veri içinde gizli, anlamlı ve yararlı örüntülerin büyük ölçekli veritabanlarından otomatik biçimde elde edilmesini sağlayan VTBK süreci içinde bir adımdır. VM, makine öğrenimi, istatistik, veritabanı yönetim sistemleri, veri ambarlama, koşut programlama gibi farklı disiplinlerde kullanılan yaklaşımları birleştirmektedir. Makine öğrenimi, istatistik ve VM arasındaki yakın bağ kolaylıkla görülebilir.

Bu üç disiplin veri içindeki ilginç düzenlilikleri ve örüntüleri bulmayı amaçlar. Makine öğrenimi yöntemleri VM algoritmalarında kullanılan yöntemlerin çekirdeğini oluşturur. Makine öğreniminde kullanılan karar ağacı, kural tümevarımı pek çok VM algoritmasında kullanılmaktadır. Makine öğrenimi ile VM arasında benzerliklerin yanı sıra farklılıklar da göze çarpmaktadır. Öncelikle VM algoritmalarında kullanılan örneklem boyutu, makine öğreniminde kullanılan veri boyutuna nazaran çok büyüktür. Genellikle makine öğreniminde kullanılan örneklem boyu 100 ile 1000 arasında değişirken VM algoritmaları milyonlarca gerçek hayat nesneleri üzerinde uğraşmaktadır ki bunların karakteristiği boş (boş), artık, eksik, gürültülü değerler olarak belirlenebilir. Aynı zamanda VM algoritmaları bilgi keşfetmeye uygun nesne niteliklerinin elde edilme sürecindeki karmaşıklıkla baş etmek zorundadır. Olasılıksal veri nedenlemede VM, istatistik alanındaki bir çok metodu kullanmasına rağmen, nesnelerin nitelik(ler) değer(ler|in)e bağlı çıkarsama yapmada bilinen istatistiksel metotlardan ayrılmaktadır [10].

Örneğin, x-kare veya t testi gibi istatistiksel test yöntemleri birden fazla nitelik arasında korelasyon derecesini belirli bir güvenlik arasında verebilmesine karşılık, belirli nitelik değerleri arasındaki ilişkinin derecesini açığa çıkaramazlar. İstatistiksel yöntemler karar verme mekanizmasında VM disiplini ortaya çıkmadan önce çok sık kullanılırdı. Ancak bu yöntemlerin kullanım zorluğu (uzman kişileri tutma/başvurma), VM algoritmalarının uygulama kolaylılığı ile karşılaştırıldığında, veri nedenleme sürecindeki en güç adımı oluşturuyordu. Veritabanı yönetim sistemleri (VTYS) büyük miktardaki yapısal bilgiyi saklama ve etkin bir biçimde erişim sağlamakla yükümlüdür. VTYS'lerde veri düzenlemesi, ilgili organizasyonun işletimsel veri ihtiyacı doğrultusunda gerçekleştirilir ki bu her zaman bilgi keşfi perspektifi ile bire-bir çakışmaz. Bu açıdan veritabanındaki veriler temizleme, boyut indirgeme, transfer, vb. işlemlerinden geçirilerek VM kullanımına sunulurlar. VM teknikleri ayrı araç olarak

sağlanabileceği gibi bir VTYS ile de entegre olabilirler.

1.3.1 Veri Madenciliğinde Karşılaşılan Problemler

Küçük veri kümelerinde hızlı ve doğru bir biçimde çalışan bir sistem, çok büyük veritabanlarına uygulandığında tamamen farklı davranabilir. Bir VM sistemi tutarlı veri üzerinde mükemmel çalışırken, aynı veriye gürültü eklendiğinde kayda değer bir biçimde kötüleşebilir [11]. İzleyen kesimde günümüz VM sistemlerinin karşı karşıya olduğu problemler incelenecektir.

1.3.1.1 Veritabanı Boyutu

Veritabanı boyutları inanılmaz bir hızla artmaktadır. Pek çok makine öğrenimi algoritması bir kaç yüz tutanaklık oldukça küçük örneklemleri ele alabilecek biçimde geliştirilmiştir. Aynı algoritmaların yüz binlerce kat büyük örneklemlerde kullanılabilmesi için azami dikkat gerekmektedir. Örneklemin büyük olması, örüntülerin gerçekten var olduğunu göstermesi açısından bir avantajdır ancak böyle bir örneklemden elde edilebilecek olası örüntü sayısı çok büyüktür. Bu yüzden VM sistemlerinin karşı karşıya olduğu en önemli sorunlardan biri veritabanı boyutunun çok büyük olmasıdır. Dolayısıyla VM yöntemleri ya sezgisel/buluşsal bir yaklaşımla arama uzayını taramalıdır ya da örneklemi yatay/dikey olarak indirgemelidir. Yatay indirgeme çeşitli biçimlerde gerçekleştirilebilir. İlkinde, belirli bir niteliğin alan değerleri önceden sıradüzensel olarak sınıflandırılır (ya da kategorize edilir) ki buna genelleştirme işlemi de denilmektedir. Sonrasında, ise ilgili niteliğin değerleri önceden belirlenmiş genelleme sıradüzeninden aşağıdan yukarıya doğru seviye seviye günlenir (yani, üst nitelik değeri ile değiştirilir) ve tekrarlı (mükerrer) çoklular çıkarılır. İkincisinde, oldukça sağlam olan örnekleme kuramı kullanılarak çok büyük oylumlu veri öyle bir boyuta indirgenir ki hem kaynak veri belirli bir güven aralığında temsil edebilir hem de indirgenen veri kümesinin oylumu makine öğrenimi

algoritmalarınca işlenebilir olması olurlu olabilir. Sonuncusunda ise, sürekli değerlerden oluşan bir alana sahip nitelik üzerine kesikleştirme tekniğinin uygulanmasıdır. Sürekli değerlerin belirli aralık değerlerine dönüştürülmesi ile tekrarlılık arz eden çoklular ortadan kaldırılarak yatay indirgeme sağlanabilir. Aslında bu kesikleştirme tekniği, sürekli sayısal değerler için geçerli olmayan makine öğrenim algoritmaları için bir önkoşul ya da ön işlemedir ki bu konu ayrı bir alt başlık olarak verilecektir. Dikey indirgeme, artık niteliklerin indirgenmesi işlemidir ki bu artık işleme alt başlığında tartışılacaktır.

1.3.1.2 Gürültülü Veri

Büyük veritabanlarında pek çok niteliğin değeri yanlış olabilir. Bu hata, veri girişi sırasında yapılan insan hataları veya girilen değerin yanlış ölçülmesinden kaynaklanır. Veri girişi ya da veri toplanması sırasında oluşan sistem dışı hatalara gürültü adı verilir. Ancak günümüzde kullanılan ticari ilişkisel veritabanları veri girişi sırasında oluşan hataları otomatik biçimde gidermek konusunda az bir destek sağlamaktadır. Hatalı veri gerçek dünya veritabanlarında ciddi problem oluşturabilir. Bu durum, bir VM yönteminin kullanılan veri kümesinde bulunan gürültülü verilere karşı daha az duyarlı olmasını gerektirir. Gürültülü verinin yol açtığı problemler tümevarımsal karar ağaçlarında uygulanan metotlar bağlamında kapsamlı bir biçimde araştırılmıştır. Eğer veri kümesi gürültülü ise sistem bozuk veriyi tanımalı ve ihmal etmelidir.

1.3.1.3 Boş Değerler

Bir veritabanında boş değer, birincil anahtarda yer almayan herhangi bir niteliğin değeri olabilir. Boş değer tanımı gereği kendisi de dahil olmak üzere hiç bir değere eşit olmayan değerdir. Bir çokluda eğer bir nitelik değeri boş ise o nitelik bilinmeyen ve uygulanamaz bir değere sahiptir. Bu durum ilişkisel veritabanlarında sıkça karşımıza çıkmaktadır. Bir ilişkide yer alan tüm çoklular

aynı sayıda niteliğe, niteliğin değeri boş olsa bile, sahip olmalıdır. Örneğin kişisel bilgisayarların özelliklerini tutan bir ilişkide bazı model bilgisayarlar için ses kartı modeli niteliğinin değeri boş olabilir. Boş değer, (1) bilinmeyen, (2) uygulanamaz, ve (3) bilinmeyen veya uygulanamaz olacak biçimde üçe ayıran bir yaklaşımı ilişkisel veritabanlarını genişletmek için öne sürülebilir.

1.3.1.4 Eksik Veri

Evrendeki her nesnenin ayrıntılı bir biçimde tanımlandığı ve bu nesnelerin alabileceği değerler kümesinin belirli olduğu varsayılsın. Verilen bir bağlamda her bir nesnenin tanımı kesin ve yeterli olsa idi, sınıflama işlemi basitçe nesnelerin alt kümelerinden faydalanılarak yapılırdı. Bununla birlikte, veriler kurum ihtiyaçları göz önünde bulundurularak düzenlenip, toplandığından, mevcut veri gerçek hayatı yeterince yansıtmayabilir. Örneğin hastalığın tanısını koymak için kurallar sadece çok yaşlı insanların belirtilerinin bulunduğu bir veri kümesi kullanılarak üretilseydi, bu kurallara dayanarak bir çocuğa tanı koymak pek doğru olmazdı. Bu gibi koşullarda bilgi keşfi modeli belirli bir güvenlik (ya da doğruluk) derecesinde tahmini kararlar alabilmelidir.

1.3.1.5 Artık Veri

Verilen veri kümesi, eldeki probleme uygun olmayan veya artık nitelikler içerebilir. Bu durum pek çok işlem sırasında karşımıza çıkabilir. Örneğin, eldeki problem ile ilgili veriyi elde etmek için iki ilişkiyi ortak nitelikler üzerinden birleştirirsek sonuç ilişkide kullanıcının farkında olmadığı artık nitelikler bulunur. Artık nitelikleri elemek için geliştirilmiş algoritmalar özellik seçimi olarak adlandırılır. Özellik seçimi, tümevarıma dayalı öğrenmede budama öncesi yapılan bir işlemdir. Başka bir deyişle, özellik seçimi, verilen bir ilişkinin içsel tanımını, dışsal tanımın taşıdığı (veya içerdiği) bilgiyi bozmadan onu eldeki niteliklerden daha az sayıdaki niteliklerle (yeterli ve gerekli) ifadeleyebilmektir. Özellik

seçimi yalnızca arama uzayını küçültmekle kalmayıp, sınıflama işleminin kalitesini de arttırır.

1.3.1.6 Dinamik Veri

Kurumsal çevrim-içi veritabanları dinamiktir, yani içeriği sürekli olarak değişir. Bu durum, bilgi keşfi metotları için önemli sakıncalar doğurmaktadır. İlk olarak sadece okuma yapan ve uzun süre çalışan bilgi keşfi metodu bir veritabanı uygulaması olarak mevcut veri tabanı ile birlikte çalıştırıldığında mevcut uygulamanın da performansı ciddi ölçüde düşer. Diğer bir sakınca ise, veritabanında bulunan verilerin kalıcı olduğu varsayılıp, çevrimdışı veri üzerinde bilgi keşif metodu çalıştırıldığında, değişen verinin elde edilen örüntülere yansıması gerekmektedir. Bu işlem, bilgi keşfi metodunun ürettiği örüntüleri zaman içinde değişen veriye göre sadece ilgili örüntüleri yığmalı olarak günleme yeteneğine sahip olmasını gerektirir. Aktif veritabanları tetikleme mekanizmalarına sahiptir ve bu özellik bilgi keşif metotları ile birlikte kullanılabilir.

1.3.1.7 Farklı tipteki verileri ele alma

Gerçek hayattaki uygulamalar makine öğreniminde olduğu gibi yalnızca sembolik veya kategorik veri türleri değil, fakat aynı zamanda tamsayı, kesirli sayılar, çoklu ortam verisi, coğrafik bilgi içeren veri gibi farklı tipteki veriler üzerinde işlem yapılmasını gerektirir. Kullanılan verinin saklandığı ortam, düz bir kütük veya ilişkisel veritabanında yer alan tablolar olacağı gibi, nesneye yönelik veritabanları, çoklu ortam veritabanları, coğrafik veritabanları vb. olabilir. Saklandığı ortama göre veri, basit tipte olabileceği gibi karmaşık veri tipleri (çoklu ortam verisi, zaman içeren veri, yardımlı metin, coğrafik, vb.) de olabilir. Bununla birlikte veri tipi çeşitliliğinin fazla olması bir VM algoritmasının tüm veri tiplerini ele alabilmesini olanaksızlaştırmaktadır. Bu yüzden veri tipine özgü adanmış VM algoritmaları geliştirilmektedir.

1.4 Elektronik Hasta Kayıtları

1.4.1 Tıp ve Enformasyon

Tıp, sağlık bakım sektörü enformasyon yoğunlukludur. Tıpta bilgi toplanır, ayıklanır işlenir, istatistik yapılır ve karar verilir [12]. Tıp alanında bilgiye erişim ve değerlendirilmesinin önemi nedeniyle biyoenformatik araştırma dalının doğmasına neden olmuştur [13]. Geleneksel olarak tıp araştırmalarında veri toplanır ve istatistikler yardımıyla bir hastalık konusunda daha belirleyici sonuçlara gidilir. Hastanelerde kullanılan veri tabanlarında tutulan kayıtlar birçok kişi tarafından değişik amaçlarla kullanılabilmektedir. Finansal analizler yapılarak hastanenin gelecekteki yatırımlarına yön verilebilir. Daha etkin hizmet için bu veriler hastane yöneticileri tarafından değerlendirilmektedir. Doktorlar, eczacılar, medikal firmaları, sigorta şirketleri, başta sağlık bakanlığı olmak üzere diğer kamu kuruluşları bu verilerden yararlanmaktadır [8]. Sadece iki parametre üzerinde istatistik yapılarak kalp hastalığı riski olan insanları çıkarmak olasıdır [14]. Çok temel parametrelerle yapılan bir istatistikle;

1. Önemli hastalıkların ortaya çıkış faktörleri bulunabilir.
2. Nüfus içerisindeki risk taşıyan kişiler, aileler veya topluluklar belirlenebilir.
3. Sağlık giderleri daha aza indirilebilir.
4. Bir ameliyat işleminde temel istatistik veriler ortaya konabilir.

Daha ileri düzeydeki veri toplama ve işleme sistemleri ile,

1. Teşhis çalışmaları
2. Tedavi kararları
3. Sigorta sahtekarlıklarının bulunması

4. Gen araştırmaları

çalışmaları yapmak olanaklıdır [15]. Son yılda gelişen teknolojilerle tıpta enformasyon patlaması yaşanmıştır [16]. Bununla birlikte her yıl sağlık alanındaki makale sayısı da hızla artmaktadır. Tıbbı bilginin yarılanma ömrünün 4 yıl olduğu söylenmektedir [17].

1.4.2 Elektronik Hasta Kayıtları

Amerika birleşik devletlerinde her yıl hastane hataları ile ölen hasta sayısı 900.000 kişidir [18]. Bu ölümlerin başlıca sebepleri başlıca reçete hataları, hastanede enfeksiyon, yanlış tedavi sayılabilir. Hasta Kayıtları bu hataların en aza indirgenmesi açısından hayati önem taşımaktadır. Elektronik hasta kayıtları ve buna bağlı hatalar genelde;

- Kayıt tablolarındaki eksiklik, uygun düzenlememe ve yetersiz bilgi kaydı.
- Kayıtların kaybolması veya karışıklık.
- Bir hastanın tüm geçmişine ulaşamama.
- Tıbbi probleme göre kayıt tutulmaması.
- Kaynağa ve tarihe göre kayıt tutulmaması.

sayılabilir. Çözüm olarak düzgün bilgi kaydı, bir kişinin yaşamı boyunca sağlık durumunun ve aldığı sağlık bakımı ile ilgili bilgilerin tutulmasıdır.

Elektronik hasta kaydı ile;

- Verilere daha iyi ve daha hızlı ulaşım
- Daha iyi kalitede veriler
- Verileri çok yönlü olarak sunma olanağı

- Sağlık bakımının sonuçlarının (outcomes) ölçülebilmesi ve değerlendirilmesi

sağlanabilir. EHK'nın temel amacı hasta bakımını desteklemektir. Faturalama, yasal kayıtlar gibi amaçlar için gerekli tüm kayıtlar EHK'dan üretilmelidir. Her birinin amacı diğerlerinden farklı olan birden fazla EHKS olabilir. EHKS'nin geliştirilmesinde belirleyici olan sağlık bakımı verilen birimin özellikleridir. Bütün kayıtların temel bileşenleri, veri elementleri ve veri elementlerinin gösterimi açısından aynı olmalıdır. Veriler bir kez girilmeli, ihtiyaç duyulduğu her zaman ve tüm amaçlar için kullanılabilmelidir. Her tip kayıt birbirleri ile ilişkili olmalıdır.

1.4.3 EHK Sistemleri

1. Yatan hasta tıbbi kayıtları,
2. Ayaktan hasta tıbbi kayıtları,
3. Birinci basamak sağlık kayıtları,
4. Hastalığa özel tıbbi kayıtlar (kardiyoloji, hipertansiyon, diyabet, vs.),
5. Yoğun bakım kayıtları,
6. Hemşirelik hizmetleri kayıtları,
7. Radyolojik görüntü kayıtları,
8. Fatura kayıtları vb.

1.4.4 Sağlık Bilişimi Standartları

Sağlık bilişiminde yer alan verilerin çok çeşitli olması ve bu verilerin ifade edilmesi, değerlerini koruyarak saklanması için sağlık bilişimi standartlarının da oluşturulması gerekliliğini ortaya koymuştur. Kompleks yapısı nedeniyle,

sağlık bakım ortamı fazlasıyla heterojen bir veri işleme ihtiyacına sahiptir. Hiçbir yazılım firması veya in-house yazılım üreten grup, bu gereksinimlerin hepsini birden karşılayamaz.Sağlık Enformasyon Sistemlerindeki çeşitlilik bir zorunluluktur. Elektronik hasta kayıtları ile kağıt dosya sistemine dayalı tıbbi kayıt sisteminin uyumunun sağlanması gerekir. Bunun dışında enformasyonun kurum içerisinde ve kurumlar arasında paylaşılması (iletişim) gerekmektedir. Hasta başka bir kurumda tedavi edildiğinde bu kurumda eski bilgilerine ihtiyaç duyulacaktır. Ayrıca farklı sistemlerin birlikte çalışması (interoperability) gerekebilir. Fonksiyonel, semantik ve en önemlisi, karar destek sistemlerinin geliştirilmesi ve kullanılması için tıbbı bilişimde standartlaşma gereklidir. Bu amaçla içerik ve yapı standartları geliştirilmiştir.

1.4.5 Standartlar

Tanımlayıcı standartlara ASTM E1384 Standard Guide for Content and Structure of the Electronic Health Record gösterilebilir. HL7 Health Level 7 olarak adlandırılan diğer bir standart da verilerin paylaşımı için geliştirilmiş bir standarttır. CEN TC 251 Electronic healthcare record communication (ENV 13606) de kayıtlar için geliştirilmiş bir standarttır.

1.4.6 Klinik Verilerin Kodlanması ile İlgili Standartlar

Tıbbi Kayıt Modelleri

- Zaman-yönelimli (time-oriented) tıbbi kayıt
- Kaynak-yönelimli (source-oriented) tıbbi kayıt
- Problem-yönelimli (problem-oriented) tıbbi kayıt

Lawrence L. Weed tarafından 1960'larda geliştirildi ve her hasta için bir Problem Listesi içerir. Klinik notlar SOAP (Subjective, Objective, Assesment, Plan) yapısı ile tutulur

1.4.6.1 Problem-Yönelimli Tıbbi Kayıt

SOAP, Öznel (Subjective), Nesnel (Objective), Değerlendirme (Assesment), Planlama (Plan), İzleme veya Klinik Notlar (SOAP) şeklinde tanımlanabilir.

1.4.6.2 Problem-Yönelimli Tıbbi Kayıt (POMR)

Öznel (Subjective) olarak bu kayıt şu bilgileri içerir;

- Başvuru nedeni ve şikayetleri
- Şimdiki hastalık öyküsü
- Aile öyküsü
- Özgeçmişi
- Alerji öyküsü
- Kullandığı ilaçlar
- Sistemlerin sorgulanması
- Sağlığı etkileyen riskler

Nesnel olarak

- Vital bulgular
- Sistemik muayene bulguları
- Uzmanlık ve şikayete yönelik muayene bulguları
- Değerlendirme (Assesment);
- Problem listesi
- Ön tanı

- Ayırıcı tanı

Planlama olarak;

- Ayrıntılı tanısal tetkikler
- Tedavi
- Hasta eğitimi
- İzleme veya Klinik Notlar (SOAP);
- Yeni öznel bulgular
- Yeni nesnel bulgular
- Yeni değerlendirme
- Yeni plan

1.5 Esnek Veri Girişi Teknolojileri

1. Dinamik yapılandırılmış veri girişi (Structured Data Entry) araçları
2. Doğal dil işleme (Naturel Language Processing) araçları
3. Sürekli konuşmayı tanıma (Continuous Speech Recognition) araçları

1.6 SQL

SQL (Structured Query Language) kendisi bir programlama dili olmamasına rağmen bir çok kişi tarafından programlama dili olarak bilinir. SQL herhangi bir veri tabanı ortamında kullanılan bir alt dildir (sub language). SQL ile yalnızca veri tabanı üzerinde işlem yapabiliriz. SQL cümlecikleri kullanarak veri tabanına kayıt ekleyebilir, olan kayıtları değiştirebilir silebilir ve bu kayıtlardan

listeler oluşturabiliriz. SQL cümlecikleri genellikle aynı olmakla birlikte farklı veri tabanı ortamlarında değişebilmektedir. Ayrıca veri tabanlarının kendilerine özgü SQL komutlarında vardır.

1.6.1 Veri Tabanlarının Sağladığı Faydalar

1. Minimum Veri Tekrarı: Tam fonksiyonlu veri tabanlarında farklı tablolarda tekrarlı verilerin içerilmesi durumunda bu tablolar arasında mantıksal bütünlük çerçevesinde veritabanı yinelemelerden kurtarılır. Böylece veritabanı denetim altında tutulmuş olur.

2. Verinin Bütünlüğü: Veritabanları sahip oldukları anahtarlarla birbirleriyle bağ oluştururlar. Böylece veriler arasındaki ilişkiler belli kısıtlar kullanılarak veri bütünlüğü sağlanır.

3. Verinin Tutarlılığı: Veritabanında yapılan değişiklik tüm veritabanını etkiler. Veri tutarlılığı için;

- Belli bir andaki işlemlerin tamamı (transaction) son bulmadan veriler üzerinde değişiklik yapılmamasını sağlar.
- Aynı veri üzerinde sorgu gerçekleştirenler yazma yapanları beklemez.
- Aynı veri üzerinde yazma yapanlar okuma yapanları beklemezler.
- Aynı veri üzerinde yazma yapanlar aynı veri üzerinde yine yazma yapanların işlemlerinin sonuçlanmasını beklerler.

1.6.2 Veri Tabanı Sistem Hakları

Sistem hakları veritabanının kullanımına yönelik kullanıcılara sistem tarafından verilen yetkilerdir.

- CREATE SESSION: Veritabanına bağlanma hakkını içerir. Bu hakka sahip olmayan kullanıcı veritabanına bağlanamaz.
- CREATE TABLE: Tablo yaratabilmek için gerekli olan yetkidir.
- CREATE VIEW: Görüntü yaratmak için gerekli olan haktır.
- CREATE USER: Kullanıcı yaratma hakkıdır.
- ALTER TABLESPACE: Tablespacelerin yapısını değiştirme hakkıdır.
- DROP ANY INDEX: Veritabanındaki indeksi silme yetkisidir.

1.6.3 SQL Tanımlama

SQL'de kullanılan create, alter, drop komutları ile veritabanı nesneleri üzerinde yapılan işlemlere veri tanımlama (data definition) denir. Veritabanı nesnelerin oluşturulması, değiştirilmesi, silinmesi, tablo, index ve clusterların oluşturulması için kullanılır.

SQL veritabanı ile iletişim kurulmasını sağlayan standartlaştırılmış sorgulama dilidir.

SQL sorgulama dili, ANSI standardıdır ve MySQL veritabanı, Access, Microsoft SQL sunucu, Oracle, Sybase gibi veritabanı yönetim sistemlerini de kullanır.

1.6.4 Veri Tanımlama Dili Olarak SQL

Bir SQL server içersinde tüm veriler tablolar içerisinde bulunmaktadır ve tablolar da veritabanı grupları içerisinde bulunmaktadır.

CREATE Bir uygulama için tablolar oluşturulmadan önce database oluşturulmalıdır. Create nesne yaratmak için kullanılır.

DROP Oluşturulmuş olan database ve içindeki tüm tabloları silmek için kullanılır.

ALTER Oluşturulan nesnenin yapısını değiştirmek için kullanılır.

GRANT Hak vermek için kullanılır.

REVOKE Verilen hakkı geri almak için kullanılır.

ANALIZE İndeksler hakkında istatistik hazırlamada kullanılır.

AUDIT Veritabanı nesnelerinin kontrol işlemleri için kullanılır.

COMMENT Nesneler için yorum yazmada kullanılır.

1.6.5 Veri Erişme Dili Olarak SQL

SQL ile veritabanına erişip, tablo ve görüntülerin içerisindeki değişiklikler yapmak için bazı temel komutlar vardır.

1.6.5.1 SELECT

Select yapısı ile hazırlanan SQL sorgu cümlecikleri, veritabanımızdan istediğimiz alanlardaki istediğimiz özelliklere sahip verilerin sütunlar halinde çekilmesine olanak sağlar. İç içe alt sorgulu (subquery) olarak da kullanılabilir. Genel yapısı;

SELECT [ALL | DISTINCT [ON (expression [, ...])]]

* | expression [AS output_name] [, ...]

[FROM from_item [, ...]]

[WHERE condition]

[GROUP BY expression [, ...]]

[HAVING condition [, ...]]

[{ UNION | INTERSECT | EXCEPT }

[ALL] select] [ORDER BY lexpression

[ASC | DESC | USING operator] [, ...]] [LIMIT { count | ALL }];

şeklindedir.

Bütün verilerin listelenmesi "*" işareti kullanarak gerçekleştirilir.

1.6.5.2 INSERT

Tabloya yeni kayırt eklemek için kullanılan komuttur. Yapısı;

INSERT INTO table [(column [, ...])]

{ DEFAULT VALUES | VALUES

({ expression | DEFAULT } [, ...]) | query } şeklindedir.

1.6.5.3 UPDATE

SQL sorgu cümleleriyle tabloya daha önceden nasıl veri ekliyorsak var olan veriyi de değiştirmemize imkan sağlayan yapıdır. Genel yapısı;

UPDATE [ONLY] table SET column = { expression | DEFAULT }

[, ...] [FROM fromlist] [WHERE condition] şeklindedir.

1.6.5.4 DELETE

Veritabanından veri silinebilmesini sağlar. Bu işlem gerçekleştirilirken veritabanındaki bilgiler olduğu gibi silinebileceği gibi sadece belirli özelliklere sahip veriler de silinebilir. Genel yapısı;

DELETE FROM [ONLY] table [WHERE condition] şeklindedir.

1.6.6 Şartlı İfadeler;

Bu komutlar kullanılırken belli şartları sağlayan verilerin çekilmesi de istenebilir. Şartlı ifadeler için;

Operatör	Şart
=	Eşittir
<>	Eşit Değildir
>	Büyüktür
<	Küçüktür
>=	Büyüktür ya da Eşittir
<=	Küçüktür ya da Eşittir
BETWEEN..AND	Belirli değerler arasındadır
LIKE	Belirli bir karakter dizisi parçası aranır

1.6.6.1 AND ve OR

AND ve OR ifadeleri birden fazla şartın bir araya getirilmesini sağlar. AND operatörü koyduğumuz iki şart birden doğru ise, OR operatörü ise koyduğumuz şartlardan her hangi biri doğru ise gerçekleşir.

1.6.6.2 ORDER BY

Select ile istediğimiz verileri çekebiliyoruz. Fakat bu çektiğimiz verilerin belirli alanlara göre sıralanmasını istersek ORDER BY komutunu kullanabiliriz.

ORDER BY yanında DESC kullanılırsa büyükten küçüğe,

ASC kullanılırsa küçükten büyüğe sıralanmış olur.

Default olarak DESC gelir.

1.6.6.3 LIMIT

Hazırlanan sorgu cümlelerinde edindiğimiz veri sayısını kısıtlamak için LIMIT komutu kullanılır.

1.7 POSTGRESQL

PostgreSQL, ilişkisel (relational) modeli kullanan ve SQL standart sorgu dilini destekleyen bir veritabanı yönetim sistemidir [19].

Hemen hemen tüm UNIX, Linux, FreeBSD gibi işletim sistemlerinde

çalışır. Ayrıca NT çekirdekli tüm Windows sistemlerde de doğal olarak çalıştırılabilir. Ücretsiz ve açık kaynak kodludur.

PostgreSQL'in güvenilirliği kanıtlanmıştır. Her bir sürümü defalarca kontrollerden geçirilmiştir. Geniş kullanıcı grubu ve kaynak koduna dünyanın her yerinden erişilebilir olması nedeniyle olası hatalar çok çabuk kapatılmaktadır.

PostgreSQL son sürümü ile birlikte kurumsal gereksinimlere yanıt verebilecek hale gelmiştir.

Bugün, takım halinde çalışan geliştiriciler PostgreSQL'i Perl, Apache ve PHP gibi açık kodlu yazılımlar gibi geliştirmektedirler. Kullanıcılar kaynak koda erişebilmekte ve açıkların kapanmasına, kodun geliştirilmesine ve önerilerle PostgreSQL'in geliştirilmesine yardımcı olmaktadırlar.

PostgreSQL'in istemci-sunucu mimarisine sahip olması iş gücünün bölünmesine yardımcı olur. Büyük miktarda veriyi tutabilecek ve erişebilecek şekilde düzenlenmiş bir sunucu makinesi güvenli bir veri deposu olarak kullanılabilir.

PostgreSQL Sınırları; Bilgileri saklamak için tablolar yaratıp onlara veri ekleyerek veritabanı kullanıldığında, hiç bir platformda sınırsız veri saklamamızın mümkün olmadığı ortadadır. Tüm veritabanı sistemleri bir şekilde sınırlandırılmaktadır, PostgreSQL'in bu konuda bir ayrıcalığı yoktur. Tek bir kolonda saklanabilecek veri miktarı, tabloda izin verilen en fazla kolon sayısı ve tablonun toplam sayısı bunların hepsinin bir sınırı vardır.

En son PostgreSQL sürümleri tüm sınırlarda esneklik getirmiştir. Hatta bazı sınırları kaldırmıştır.

Bir büyüklük için sınırsız denmesi PostgreSQL'in buna bir sınırlandırma koymaması demektir. Sınıra yaklaşıldığında veritabanının performan-

sında düşecektir.

1.7.0.3.1 Özellikleri;

Veritabanı büyüklüğü sınırsız; PostgreSQL bir veritabanının toplam büyüklüğü için herhangi bir sınır koymaz. Şu anda bilinen 32 TB'lik bir veritabanı vardır.

Tablo Büyüklüğü; (16Tb - 64Tb) PostgreSQL normalde tablo verilerini 8kb'lık parçalarda tutar. Temel blok büyüklüğü PostgreSQL kurulurken 32k ya kadar yükseltilebilir ve bu da teorik olarak 64 TB'lık bir sınır getirir.

PostgreSQL tablo verilerini her biri en fazla 1GB büyüklükte olabilecek çoklu dosyalarda tutar. Bu bir çok dosya anlamına gelir ve sistem başarımının düşmesine neden olur.

Büyüklük işletim sisteminden bağımsızdır.

Tablodaki satır sayısı; Sınırsızdır.

Tablo indexleri; Bunda da bir sınır yoktur. Fakat fazla index tablo başarımını düşürecektir.

Alan Büyüklüğü; Tablodaki alan büyüklüğü 1Gb ile sınırlandırılmıştır.

Tablodaki kolon sayısı; Yapılandırılmış blok büyüklükleri ve kolon tiplerine bağlı olarak değişiklik gösterir. Varsayılan değer olarak blok büyüklük olan 8k'da 250 kolon saklanabilir, bu sayı eğer alanlar basit ise (tamsayı

değerleri gibi) 1600 e kadar çıkabilir. Blok büyüklüklerini arttırmak eş zamanlı olarak bu sınırları da arttırır.

Satır büyüklüğü ; 1.6 Tb

1.7.1 PostgreSQL Kurulumu

Bütün özellikleri ile PostgreSQL'i kurabilmek için şu paketler gerekmektedir;

1.7.1.0.2 postgresql-libs Library dosyaları

1.7.1.0.3 postgresql-devel Development için gereken dosya ve kitaplıklar.

1.7.1.0.4 postgresql-jdbc PostgreSQL için Java database connectivity

1.7.1.0.5 postgresql-pl PostgreSQL için PL/Perl, PL/Tcl, and PL/Python desteği

1.7.1.0.6 postgresql-docs PostgreSQL'in SGML ve diğer biçimlerdeki belgeleri

1.7.1.0.7 postgresql-python Python için PostgreSQL arayüzü

1.7.1.0.8 postgresql-server Bir sunucuyu yaratmak için gerekli programlar

1.7.1.0.9 postgresql-tcl Tcl için PostgreSQL arayüzü

1.7.1.0.10 postgresql-test PostgreSQL test suite

1.7.1.0.11 Postgresql-contrib PostgreSQL ile dağıtılan contributed source.

PostgreSQL'i RPM den kurulduğunda kaynak koddan kuruluma oranla birçok işlemi yapmaya gerek kalmaz. Postgres kullanıcısı, veri dizini ve başlatma scripti oluşturulur. Sadece ntsysv ya da chkconfig ile sistem her başladığında PostgreSQL'in başlamasını sağlamak gerekir.

İlk çalıştırma sırasında ilklendirme (initialization) işlemi yapılacaktır.

/var/lib/pgsql/data dizinine geçilir. İlklendirme işleminden sonra bu dizinde bazı dosyalar oluşur:

Bunlar;

base global pg_hba.conf pg_ident.conf PG_VERSION pg_xlog postgresql.conf

postmaster.opts postmaster.pid

1.8 PHP

Rasmus Lerdorf tarafından kendi kişisel web sayfalarını yazmak için geliştirilmiştir. 'P'ersonal 'H'ome 'P'ages adının kısaltması ile PHP adı ortaya çıkmıştır. HTML ile yazılmış kodlar içine PHP kodlarını yazabildiğimiz için PHP bir script dilidir. İçeriği zamanla değişebilen dinamik web uygulamalarında sıklıkla kullanılır. Dilin kodlama yapısı C, Java ve Perl'e benzerlik gösterir.

1.8.1 Sunucu Tabanlı Uygulama Geliştirme

Internetin gerçek kimliğine kavuşması HyperText Transfer Protocol (HTTP)'nin ortaya çıkışı ile gerçekleşti. HTTP sayesinde web tarayıcılar web sunucularla konuşarak sunucuda yer alan imaj, ses, video gibi verileri kullanıcılara ulaştırır hale geldiler.Zamanla kullanıcılar içeri değişmeyen sayfalardan sıkıldılar.Bu

yüzden içeriği otomatik değişebilen hatta veri tabanı erişimi sağlayan sayfalar oluşturmak için çok çeşitli teknolojiler geliştirildi.

Web de dinamik sayfalar oluşturmanın temel olarak iki yolu vardır:

1. Sunucu tarafında çalışan (server-sided) uygulamalar kullanmak
2. İstemci tarafında çalışan (client-sided) uygulamalar kullanmak

İstemci tarafında çalışan uygulama olarak Java Appletleri veya Netscape'in JavaScript'i veya Microsoft'un VBScript'i gibi script dilleri kullanılabilir. Avantajları:

1. Sunucuyu meşgul etmemesi. Veri girişi kontrolleri, menüler, genişleyebilir listeler gibi istemci tarafında yapılabilecek işler için sunucu meşgul edilmemiş olur.
2. Bant genişliğini etkili kullanma. Yukarıdaki avantaj bant genişliğini de etkilemiş oluyor.

Sunucu tarafında çalışan (server-sided) uygulamalar kullanmak ise şu noktalarda istemci tarafı uygulamalara üstünlük sağlıyor:

1. Web tarayıcılarında scriptler için standart bulunmamaktadır. Bu nedenle bir tarayıcıda çalışan bir script diğerinde çalışmayabilir. Tarayıcıların java appletlerini yorumlamada kullandıkları java sınıflarının versiyonu sizin appletinizi çalıştıramayabilir. Kullanıcıdan yeni sınıfları download etmesini sağlamanız gerekebilir.
2. Sunucu tarafta çalışması zorunlu bazı uygulamalar olabilir (veri tabanı erişimi, işletim sistemi komutları veya başka bazı araçlar ancak sunucu üzerinde çalıştırılabilirler).

3. Bant genişliğini kullanım açısından işlemlerin önemli bir bölümünün sunucu tarafında yapılması gerekebilir. (Bir veri tabanı sunucusuna evimden sunucuya ait bir istemci yazılımla bağlandığım taktirde pek çok ara komutun iletilmesi söz konusu olabilir. Sunucu tabanlı uygulamam ise benden sadece sorguyu alacak ve bana da sadece sonucu gönderecektir. Bu noktada HTTP protokolünün bindireceği yük de hızı azaltabilir. Kurulacak denge önemli.)

4. Uygulamaların güncel tutulması ve bunu yaparken de istemciler üzerinde değişiklik yapılmayıp sadece sunucu üzerinde (tek bir merkezden) gereken değişikliği yapmak tercih edilebilir.

Günümüzde programcıların sunucu tarafı uygulama geliştirmeyi tercih ediyorlar. Uygulamaların tek bir merkezden sunumu ve hatta kullanıcılara program değil network üzerinden hizmet satma giderek daha çok önem kazanıyor.Ancak web tarayıcılarının neredeyse bir işletim sistemi kadar şiştiği günümüzde hem istemci hem de sunucu taraflarda yukarıdaki faktörleri göz önüne alarak dengeyi koruyacak şekilde uygulamalar geliştirmek gerekiyor.

1.9 Neden PHP?

1. Platform Çeşitliliği (Hem UNIX, hem Linux, hem de Windows için hazır)

   Kaynak kodu açık olarak dağıtılan PHP, Linux, Solaris, HP-UX, IRIX, FreeBSD vb üzerinde rahatlıkla derlenip çalıştırılabilir. Kendi başına derlenip web serverden cgi programı olarak çağrılabilir. Ancak hız ve güvenlik açısından sakınca oluşturduğu için özellikle Apache web sunucusu ile birlikte modül olarak derlenmesi tavsiye ediliyor.Windows-NT ortamında da, Internet Information Server(IIS) ile çalışmak üzere kullanılabilmektedir [20].

2. Performans (Zend yorumlayıcı motoru ile şimdi çok hızlı) PHP 3.x versiyonları popüler olmalarına rağmen ASP yorumlayıcısına karşı yavaş kalıyorlardı. PHP, 4.0 versiyonunda Zend firmasının script yorumlayıcı motorunu kullanılarak tamamen yenilendi ve hız olarak ASP ile yarışır hale geldi. Ayrıca ASP özellikle yoğun hit alınan durumlarda web sunucusunu yavaşlatırken, modül olarak kurulu Apache-PHP ikilisinde bu sorun pek yaşanmıyor.

3. Büyük ölçekli veri tabanı uygulamaları için ideal (ODBC, MySQL, PosgreSQL, Oracle ve diğerleri için arayüz desteği) ASP ve Cold Fusion gibi araçlar özellikle veri tabanlarına bağlanıp ve kullanabilme gibi özellikleri ile ön plana çıkmışlardır. PHP de arayüz oluşturduğu veritabanı yönetim sistemlerinin çeşitliliği ile ön plana çıkıyor. PHP ile birlikte:

   - MySQL
   - mSQL
   - PosgreSQL
   - ORACLE
   - MS-SQL Server
   - Sybase
   - Informix
   - InterBase
   - Solid

   gibi popüler veri tabanları kullanılabiliyor.

4. Internet standartlarına uyumluluk (LDAP, IMAP, FTP, news gibi standartlara sonuna kadar açık bir kapı)

PHP açık standartları desteklemektedir. Ona http protokolü ile ilişkisi olmayan pek çok işi yaptırabilirsiniz. Üstelik tüm bu standartların web'e çıkış kapısı olabilir. PHP derlenirken bazı opsiyonlar ve kütüphaneler kullanılarak çok fonksiyonlu bir araç haline getirilebilir.Kurum içi personel bilgilerinin tutulduğu bir LDAP sunucusuna erişim, sorgulama ve değişiklik yaptırılabilir, tamamen özelleşmiş web mail veya web news arabirimleri oluşturulabilir, FTP ve Telnet istemcilerinin yerini alabilir. TCP soketleri kullanarak kendinize özgü istemci sunucu uygulamalar bile geliştirebilirsiniz.Bunları yaparken muhtemelen Internet'e iyi bir bağlantısı olan ve güçlü donanım özelliklerine sahip bir sunucu tarafında uygulamaları çalıştırıyor olmanın avantajlarından yararlanırsınız.Üretilen uygulamaların yönetimi de tamamen otomatik veya yine web üzerinden çok az yönetimsel fonksiyon gerektirecek hale getirilebilir.

5. Gelişmiş Özellikler (Oturum Yönetimi, Semafor ve Paylaşımlı Hafıza kullanım, Sürekli Veritabanı Bağlantıları, Regular Expressions)

   PHP, oturum yönetimi (session management) konusunda da oldukça iyidir. PHP'de bu özellik Netscape Enterprise Server üzerinde kullanılan Server Side JavaScript kadar gelişmiş ve kolay kullanılır değil. NSEnterprise Server, SSJS ile geliştirilmiş uygulamaları çağrılmadan önce hafızaya yüklemekte ve bunların ortak kullanabileceği hiyerarşik session objectleri tanımlamaktadır. Bu ise işinizi gerçekten kolaylaştırmaktadır. PHP'ye de aynı işi yaptırabilirsiniz ancak PHP'de scriptler ancak çağrılınca hafızaya alındığı için bu iş için biraz uğraşmanız gerekmektedir. Paylaşımlı hafıza (shared memory) ve semafor kullanmak durumundasınız. İstemci tarafına cookie kullanarak oturum bilgisini atmanız da PHP ile oldukça kolay. Sürekli Veritabanı Bağlantısı (Persistent Database Connections) özelliği veri tabanı uygulamalarında hızı oldukça arttıran bir faktör. Normalde

bir kullanıcı veri tabanıyla ilgili bir iş yapmak için web sunucuya birden çok istem iletir. Cgi programları aynı kullanıcının her istemi için veri tabanı sunucusuna yeni bir bağlantı kurar. PHP ile persistent connection kullanıldığında sadece ilk request için veri tabanı bağlantısı kurulup sonra aynı bağlantı numarası (connection handle) kullanılarak bağlanılabilmektedir. Authentication ve authorization için gereken gereksiz bir yığın işten kurtulunmaktadır.

6. Ekonomik PHP, General Public Liscense ile ücretsiz dağıtılmaktadır.

2 MATERYAL VE METOD

2.1 TÖTM yapısı

Turgut Özal Tıp Merkezinde özel bir firma tarafından geliştirilmiş olan Hastane Bilgi Yönetim Sistemi yazılımı kullanılmaktadır. Bu yazılım Windows altında ve istemci-sunucu mimarisiyle çalışmaktadır. Veritabanı olarak ise Oracle kullanılmaktadır.

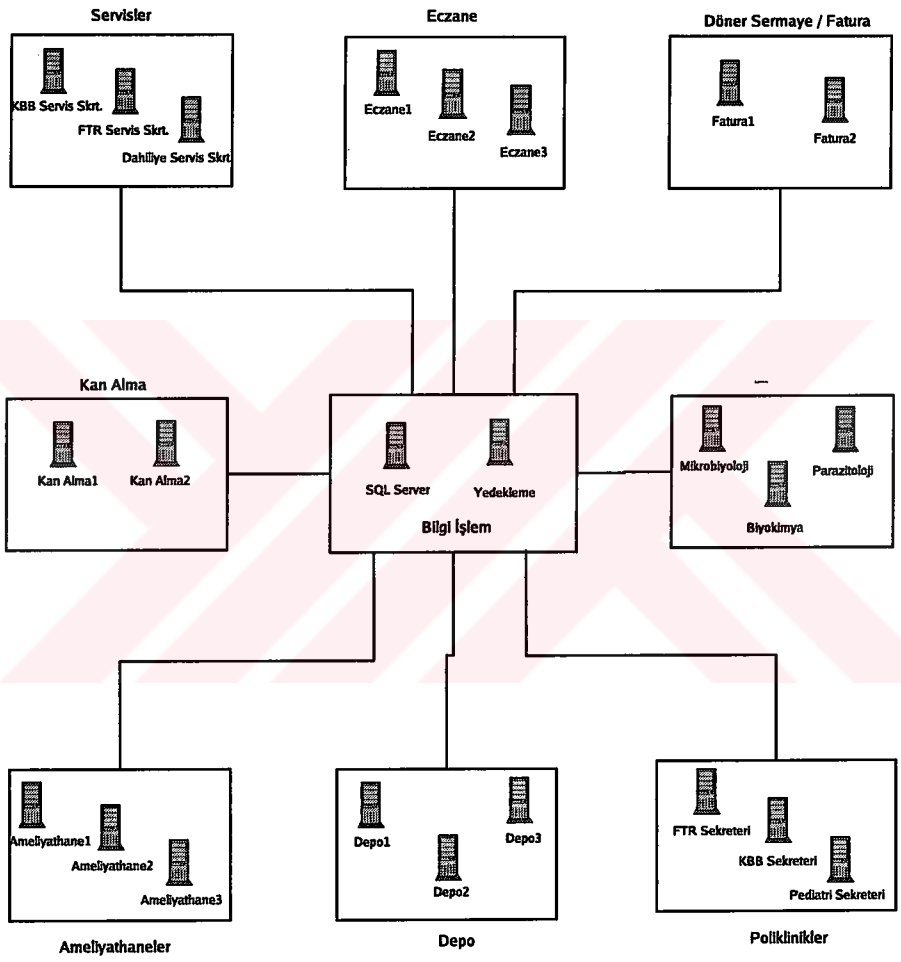
Sunucu tarafında IBM server yer almaktadır. Bu sunucu üzerinde Microsoft 2000 sunucusu ve veritabanı sunucusu olarak ta Oracle çalışmaktadır. Hastane Bilgi Yönetim Sisteminde tüm bilgiler bu Oracle sunucusunda saklanmaktadır.

Bilgisayar ağı alt yapısı Şekil 2.1 da verildiği gibidir.

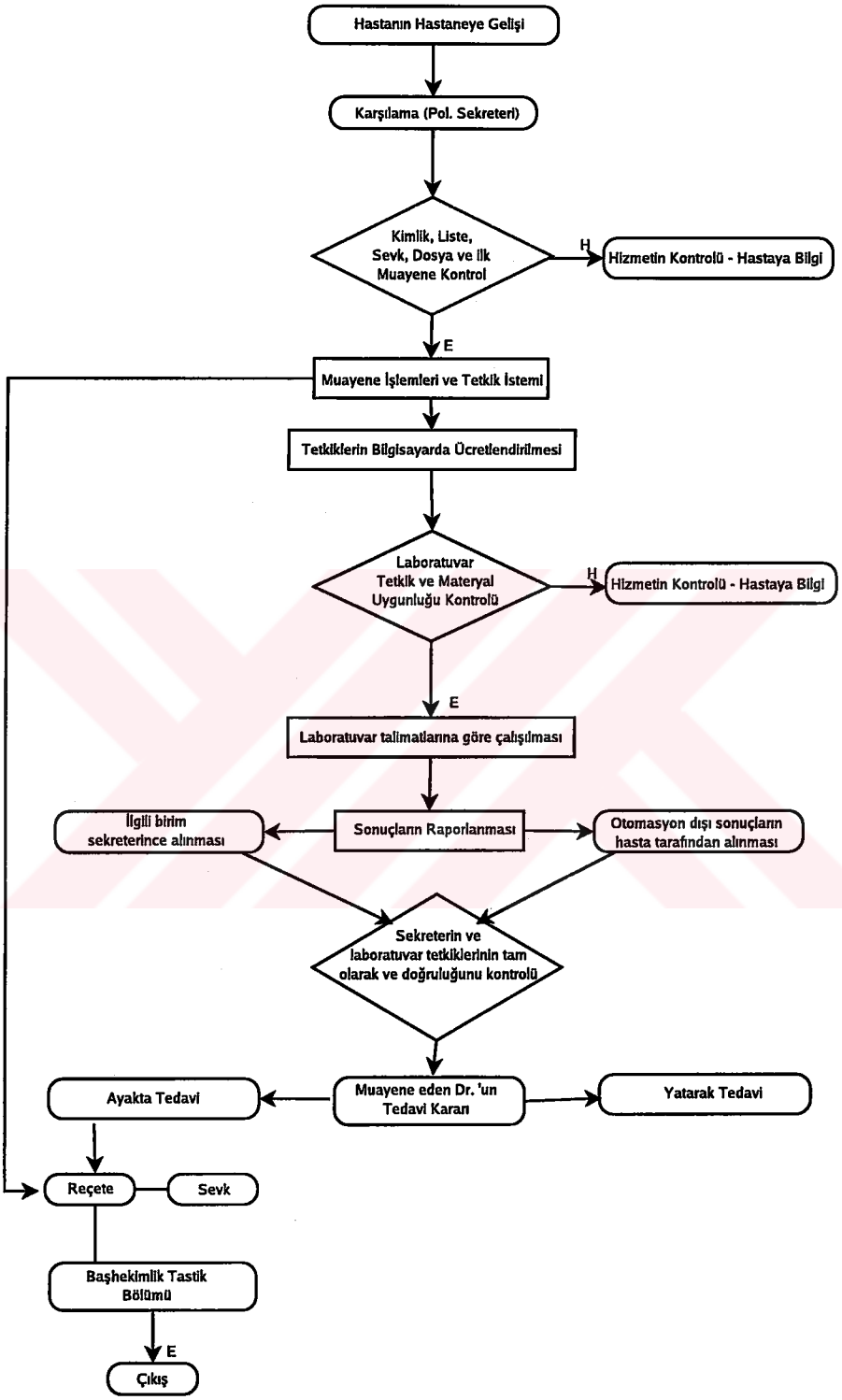
Yazılımın istemci tarafını hastane bünyesinde bu yazılımın yüklü olduğu bilgisayarlar oluşturmaktadır. Poliklinik sekreterliklerinde ki, servis sekreterlerinde ki, laboratuarlarda ki, eczane de ki, depolarda ki, ayniyattaki bilgisayarlar sunucuya bağlanarak oracle veritabanından gerekli verileri alıyor. Bu veriler üzerinde değişiklikler yapılınca veya yeni veriler oluşturulunca sunucuya bağlanarak düzenlenen veya yeni oluşturulan veriler oracle veritabanına aktarılmaktadır.

2.2 Hastane Bilgi Yönetim Sistemi İşleyişi

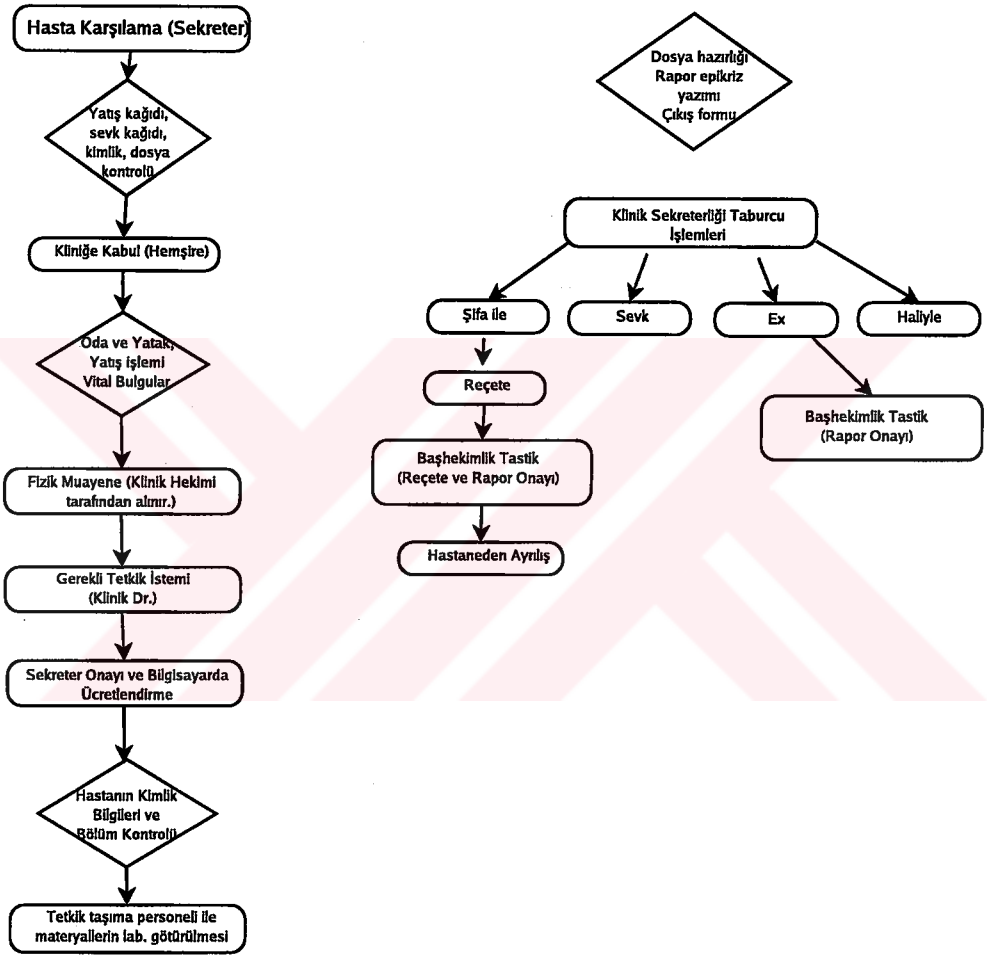
Şekil 2.2 , 2.3, 2.4, 2.5 de Turgut Özal Tıp Merkezinde poliklinik, servis ve ameliyathanelerin işleyiş şekilleri verilmiştir.



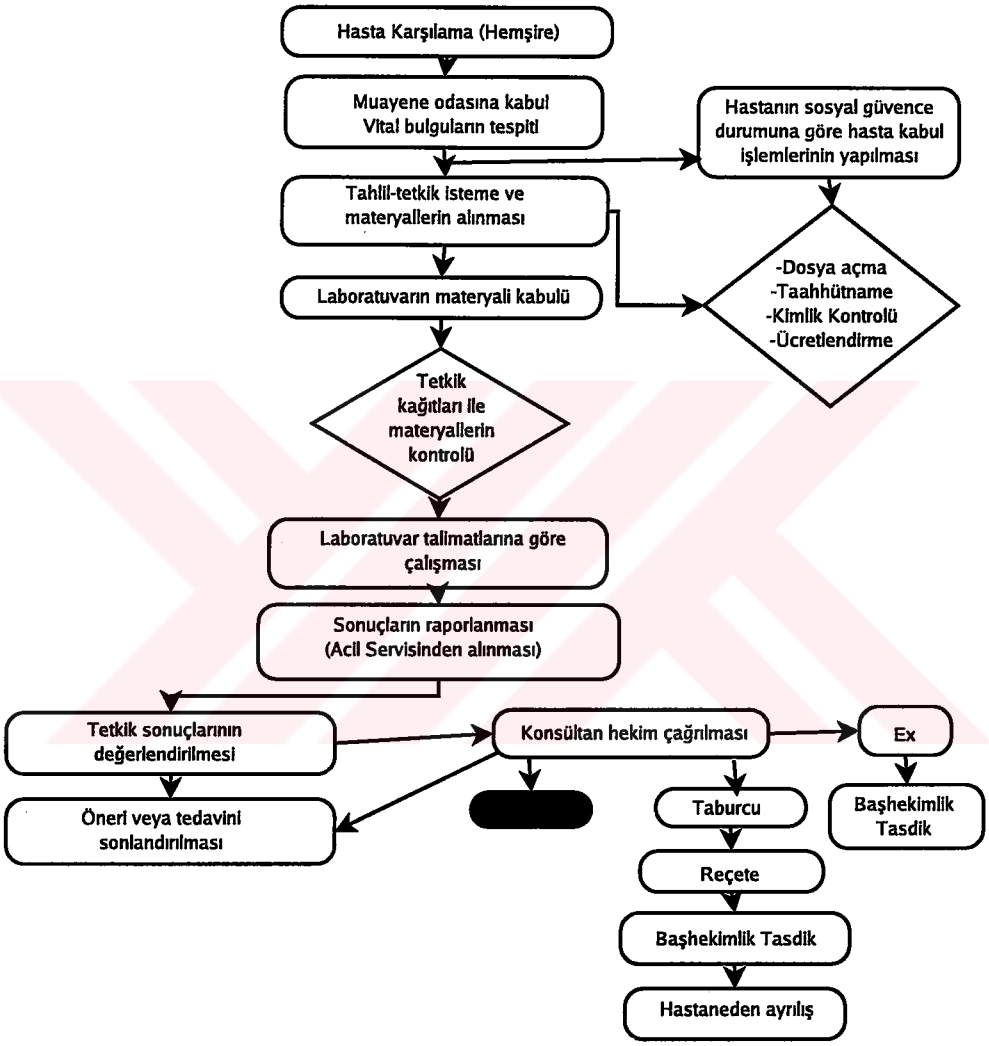
Şekil 2.1: Turgut Özal Tıp Merkezi Otomasyon Şeması



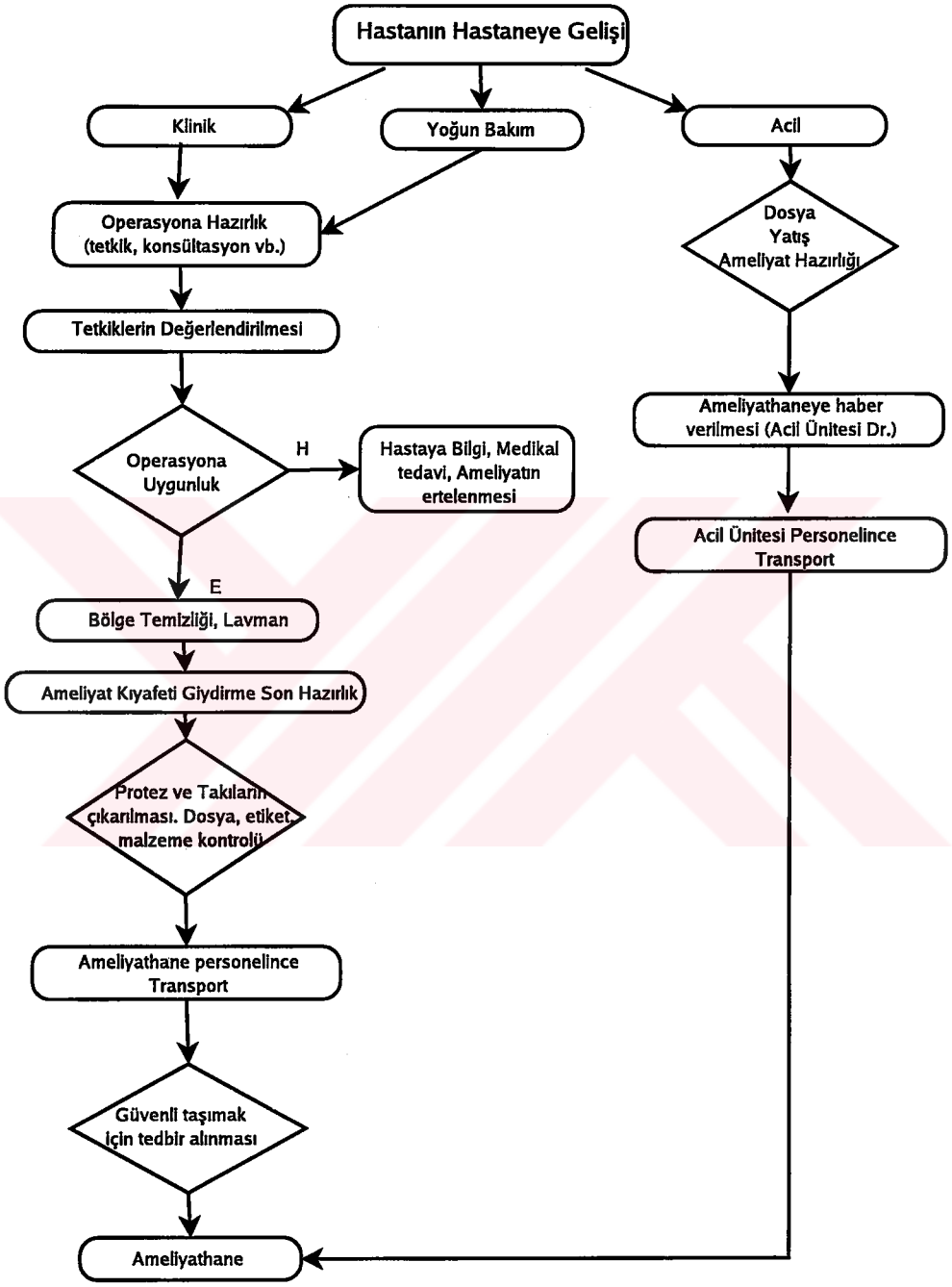
Şekil 2.2: Poliklinik İşlemleri



Şekil 2.3: Klinik İşlemleri



Şekil 2.4: Acil Servis İşlemleri



Şekil 2.5: Ameliyat İşlemleri

2.3 Verilerde sorgulamalar (SQL sorguları)

Hasta kayıt ve laboratuvar sonuç SQL tablolarının genel yapısı Şekil 2.6 de verilen şemadaki gibidir. Bütün hastane bilgi yönetim ve laboratuvar yönetim sistemleri temelde bu şekildedir.

Sorgulama yapabilmek için bize gerekli olan tablolar ve veri tipleri şunlardır.

2.3.1 Hasta Bilgi tablosu

Hasta ile ilgili bilgileri içeren bu tabloda sorgu yaparken kullandığımız alanlar

- hasta_id : Bir hastayı diğerlerinden ayıran sayı
- cinsiyet : Hastanın cinsiyetini
- dogumtarihi : hastanın doğum tarihi

SQL de bu tablolunun oluşturulması aşağıdaki gibidir.

```
CREATE TABLE hasta\_bilgi {
hasta\_id  serial,
cinsiyet varchar (1),
dogumtarihi date
};
```

2.3.2 Analizler Tablosu

Analizler ile ilgi bilgileri içeren bu tabloda sorgu yaparken kullandığımız alanlar

- analiz_id: bir analizi diğerlerinden ayıran sayı
- analiz_adi: analizin genel adı
- max_deger: analizin alabileceği maksimum değer
- min_deger: analizin alabileceği minimum değer

SQL de bu tablolunun oluşturulması aşağıdaki gibidir.

```
CREATE TABLE analizler {
analiz\_id serial,
analiz\_adi varchar(50),
max\_deger float,
min\_deger float
}
```

2.3.3 Analiz Sonuçları Tablosu

analizler ile ilgi bilgileri içeren bu tabloda sorgu yaparken kullandığımız alanlar

- sonuc_id: bir analiz sonucunu diğerlerinden ayıran sayı
- hasta_id: analizi yapılan hastayı belirten sayı.Bu alan hasta tablosu ile ilişkili
- analiz_id: hastaya uygulanan analizin hangi analiz olduğunu belirtir.Bu alan hasta tablosu ile ilişkili
- analiz_tarihi: analizin yapıldığı tarih
- sonuc: analizin sonucu

SQL de bu tablolunun oluşturulması aşağıdaki gibidir.

```
CREATE TABLE analiz\_sonuclari {
sonuc\_id serial,
hasta\_id integer,
analiz\_id integer,
analiz\_tarihi date,
sonuc numeric,
};
```

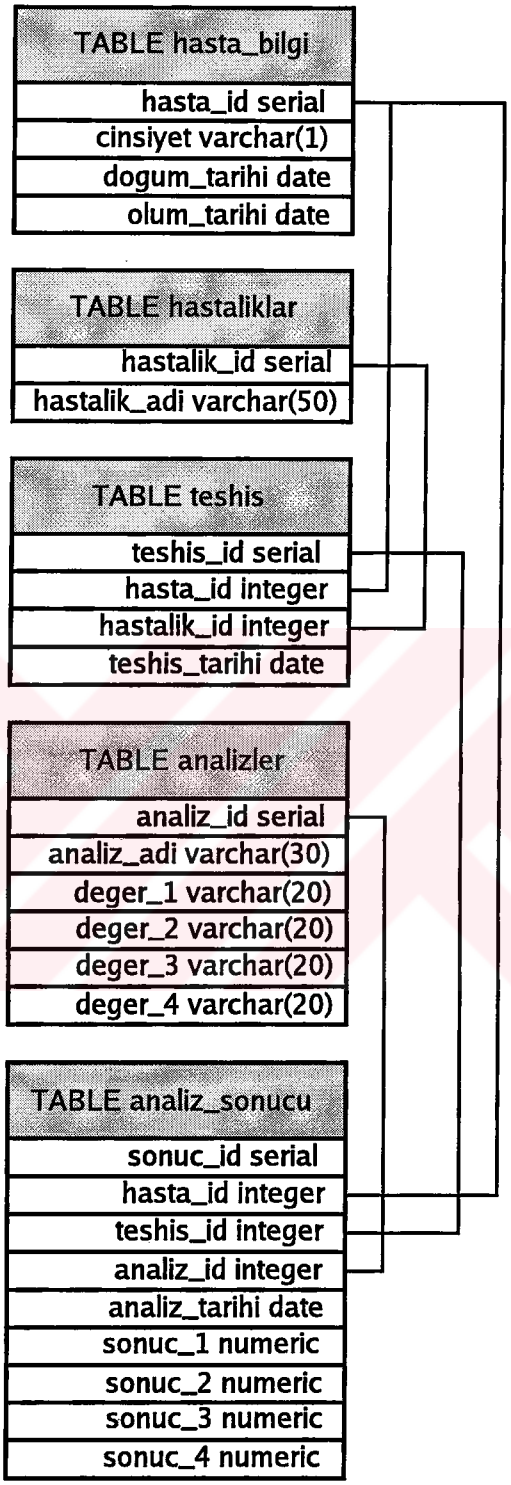
Cinsiyeti erkek olan hastalara uygulanan Lamda analizinin sonuçlarını veren bir sorgu şu şekilde yazılır

```
SELECT s.sonuc
FROM analiz\_sonuclari AS s,
hasta AS h , analiz AS a
WHERE h.cinsiyet='E' AND a.analiz\_adi='Lamda'
AND a.analiz\_id=s.analiz\_id
AND s.hasta\_id=h.hasta\_id
```

Doğum tarihi 01.01.1970 ile 01.01.2005 arasında alan hastalara uygulanan LDH analizinin sonuçlarını veren bir sorgu şu şekilde yazılır

```
SELECT s.sonuc
FROM analiz\_sonuclari AS s,
hasta AS h ,
analiz AS a
WHERE h.dogumtarihi
```

```
BETWEEN '01.01.1970' AND '01.01.2005'
AND  a.analiz\_adi='LDH' AND a.analiz\_id=s.analiz\_id
AND s.hasta\_id=h.hasta\_id
```



Şekil 2.6: SQL tabloları ve bağlantıları

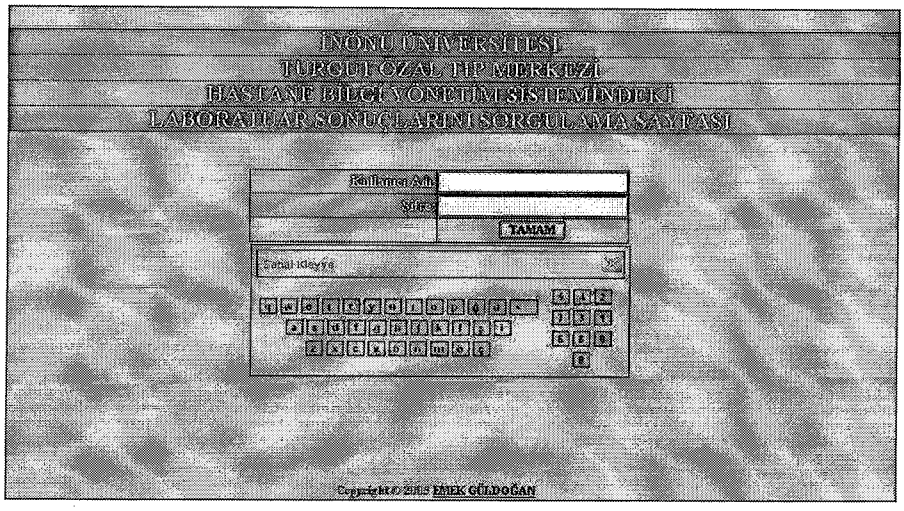
3 BULGULAR

Oluşturulan arayüz iki kısımdan oluşmaktadır.

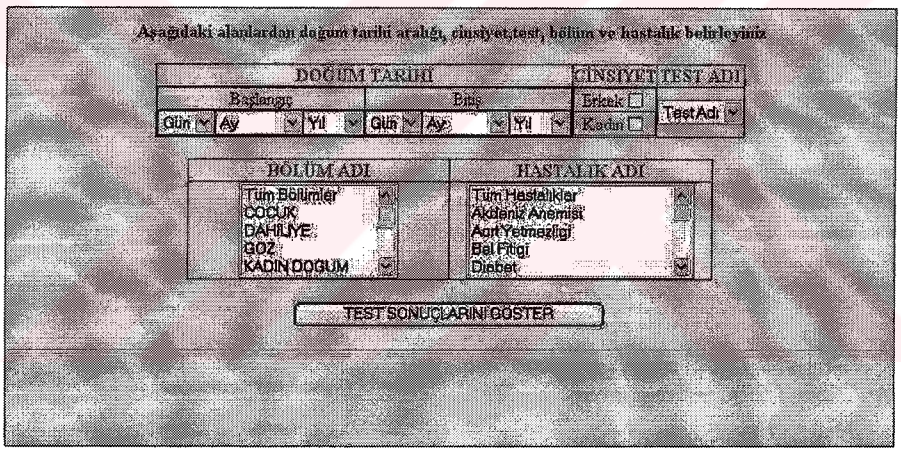
Birinci kısım da kullanıcı adı ve şifre girişi yapılmaktadır. Arayüzü kullanma izni olan kişiler kendilerine verilen kullanıcı adı ve şifre bilgilerini doğru girdiklerinde laboratuar sonuçlarını sorgulayacakları pencere karşılarına çıkmaktadır (Şekil 3.1). Kullanıcı adı ve şifreden birinin yanlış girilmesi durumunda ekrana bilgilerini yanlış girdiğine dair uyarı mesajı alınmaktadır. Üyelere Kullanıcı adı olarak üyenin emekli sicil numarası verilmektedir. Şifre girişi sanal klavye yardımı ile yapılmaktadır. Günümüzde key logger adı verilen programlar klavyeden girilen her karakterin bilgisini bir dosyaya yazabilmektedir. Üyemizin arayüze erişim yapmak istediği bilgisayarda bu programlardan birinin kurulu olma ihtimaline karşı, üyemizin girdiği şifrenin güvenliğini sağlamak için şifre giriş kısmında sanal klavye kullanılmaktadır.

İkinci kısım da laboratuar sonuçlarının sorguları yapılmaktadır (Şekil 3.2). Bu kısımda sorgulama yapacağı beş alan bulunmaktadır. Bunlar;

1. Doğum Tarihi Aralığı
2. Cinsiyet
3. Test Adı
4. Testi İsteyen Bölüm
5. Hastalık Türü



Şekil 3.1: Giriş Ekranı



Şekil 3.2: Sorgu Ekranı

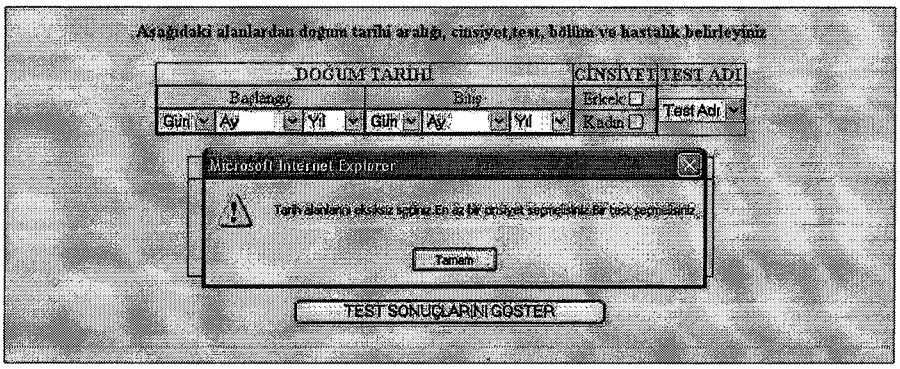
Bu alanlardan birinin seçilmemesi durumunda ekrana uyarı mesajı gelmektedir (Şekil 3.3).

Sorgu için yapılması gerekenler

1. Belli bir yaş gurubu üzerinde inceleme yapmak için iki doğum tarihi belirlenmesi

2. Cinsiyet kısmında kadının,erkeğin veya her ikisini birden seçilmesi

3. Testlerden birini belirlenmesin



Şekil 3.3: Hata Mesajı

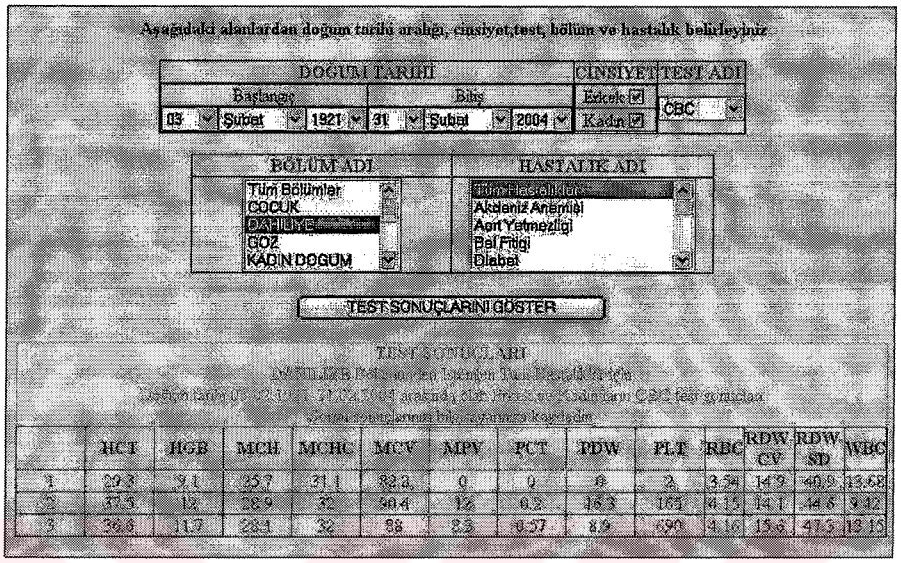
4. Sonucunu incelemek istediğimiz testin hangi bölüm tarafından istenildiğinin belirlenmesi

5. Hastalıklar kısmında tüm hastalıkları veya istenilen hastalıkların seçilmesi

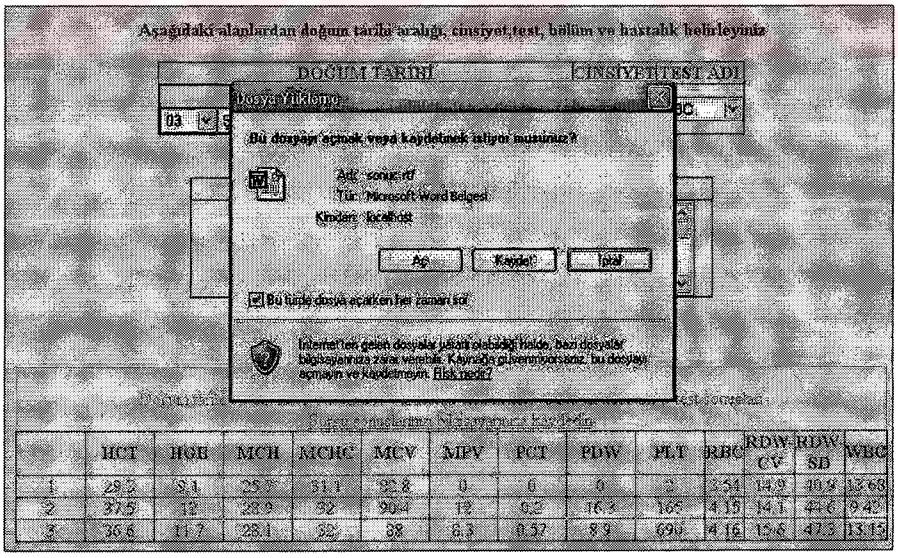
Sorgulama yapmak istediğimiz alanlar seçildikten sonra TEST SONUÇLARINI GÖSTER butonuna bastığımızda belirlediğimiz kriterlere uygun test sonuçları karşımıza gelmektedir (Şekil 3.4) .

Doğum tarihi 03.02.1921 ile 31.02.2004 arasında olan kadın ver erkeklerin dahiliye bölümünden istenmiş tüm hastalık tanılarındaki CBC test sonuçları şu şekilde çıkmaktadır.

Bu sonuçları tablo şeklinde değil de virgüllerle ayrılmış alanlar olarak bilgisayarınıza kaydetmek için Test Sonuçları başlığının altında yer alan Sorgu sonuçlarınızı bilgisayarınıza kaydedin linkine tıklamamız gerekmektedir. Bu linkine tıkladığınızda ekrana bu dosyayı açmak veya kaydetmek istiyor musunuz şeklinde bir uyarı penceresi gelecektir (Şekil 3.5) . Kaydet butonuna basıldığında sonuc.rtf adındaki dosyayı bilgisayarımıza kaydedecektir.



Şekil 3.4: Sorgu Ekranı



Şekil 3.5: Sonuçları Kaydetme

4 TARTIŞMA

Laboratuvar sonuçlarının istatistiksel araştırma için elde edilmesinde çeşitli uygulamalar yapılmaktadır. Sunucu istemci mantığıyla çalışan bu sistemlerde kullanıcı doğrudan veri üzerinde sorgulama yapmaktadır. Bu da veri tabanının performasının düşmesine neden olmaktadır.

Tıp merkezinde sadece aylık veya yıllık poliklinik ve ameliyat sayıları yönünde istatistiksel çalışmalar yapıldığı gözlenmektedir. Oysa veri kayıtlarında bunun yüzlerce katı bilgi gömülü olarak yatmaktadır. Bunun dışında bu verilerin ileriye yönelik olarak doğru bir şeklide korunup korunmadığı da şüphelidir. Sistem üzerindeki veriler başka bir ortama çekilerek, bu ortamda daha hızlı sorgulamalar ve çeşitli veri sorgulamaları optimizasyonları geliştirmek olanaklı hale getirilmiştir. Hastane Bilgi Sisteminin kullanılmadığı zamanlarda üzerindeki veriler kurulan bu sisteme çekilerek kullanıma sunulabilir. Sorgulamaların başka bir veri tabanında yapılması ile orjinal bilginin güvenliği sağlanmış olmakla birlikte, sistem hızının yavaşlatılması sorunu yaşanmayacaktır. Sorgulamalar ve işlemler sunucu üzerinde gerçekleştirildiği için bilgi ağında da bir yavaşlık gözlenmeyecektir. Web arayüzü olarak gerçekleştirilmiş olduğundan yapılacak değişikliklerde kullanıcı tarafında herhangi bir değişiklik veya program güncellemesi yapma gerekliliği ortadan kalkmaktadır.

Bu çalışma ile Turgut Özal Tıp merkezinde laboratuvar sonuçlarına dayalı istatistiksel çalışmalara veri elde etme mümkün olmaktadır. Bu veriler değerlendirilip saklanan verilerde eksiklikler olup olmadığı araştırılmalı ve ileriye yönelik önlemler alınması yolunda çalışmalar yapılması gereklidir. Ele

geçen veriler ile çok değerli sonuçlar çıkarılabilir. Bu sonuçlarla yönetimsel açıdan daha etkili planlama yapılması dışında, önemli bilimsel bugular elde edileceği apaçık ortadır. Yapılacak istatistiksel çalışmalarla, koruyucu sağlık hizmetleri, bu hizmetleri vermede kullanılan gereçlerin üretimi, satın alınmaları, hizmete sunulmaları doğrudan etkilenecektir. Toplumda gelir düzeylerine, sosyo-kültürel düzeylere, insanların alıştıkları yaşam standartlarına bağlı olarak araştırmalar yapılmalı ve sonuçların toplumun geleceği adına çok doğru saptanarak korunması, yaşatılması ve geliştirilmesi gereklidir.

Bu çalışma ile ortaya konulan arayüz, araştırmacıların ve merkezin yönetiminin kullanımına açılmalı ve her türlü bilgiyi elde etmeye yönelik bir yapının oluşturulması gereklidir.

Dünya Sağlık Teşkilatının tanımladığı ve bütün dünyanın kabul ettiği biçimiyle "sağlık, bedensel, ruhsal ve sosyal tam bir iyilik halidir". Bu iyilik halinin sağlanmasında tıp merkezlerine düşen görev, ancak iyi planlama, iyi sağlık araçları donanımı, eksiksiz bir güven ortamı, koruyucu hizmetlerin sağlanmasıdır. Bu amaçla sağlık yatırımları, altyapı ve insan gücü etkin bir biçimde bağdaştırılacaktır.

5 SONUÇ ve ÖNERİLER

Tıp alanında veri madenciliği uygulamaları son on-on beş yıl içerisinde son derece hızlı bir artış göstermiştir. Yazılım şirketleri bu konuya yönelik AR-GE çalışmaları yapmakta ve bu konuda finasmanlar ayırmaktadırlar. Kullanılan veya sorgulanması gereken verinin çok heterojen ve büyük miktarda oluşu nedeniyle standardizasyonun sağlanması oldukça güç görünmektedir. Türkiyede sağlık alanında kullanılan bilgi yönetim sistemleri birbirinden çok farklı ve standart içerdiği söylense de çoğu faturalama, depo stok kontrol özelliğinden öteye geçmemektedir. Laboratuvar sonuçlarının yer aldığı veri tabanları ise yine standartlardan uzak bir şekilde kayıt bulunmaktadır. Bu karışık ve birbiriyle ilişkilendirilmemiş veri yapıları ve tıp merkezlerindeki bilişim personeli azlığı nedeniyle önemli bir miktarda veri işlenmeden öylece durmaktadır. Sağlık bilişimi alanında veri kaydetme ve raporlama için standartlara dayalı bir yapı gerekliliği ortadadır. HL7 ve PACS standardına uyduğu söylenen bir çok HBYS sistemi ayrıntılı olarak incelendiğinde standartlardan uzak bilgi depoladığı gözlenmektedir. Bunun dışında ülkemizde sağlık alanında kullanılan sabir veriler üzerinde bir standart kodlama olmayışı da saklanan verilerin kurumlara özel kalmasını doğurmaktadır. Ulusal çapta bir istatistiksel çalışma bu veriler ile mümkün değildir. Veri karışıklığı sorunu aşmanın birkaç farklı yöntemi vardır. Bunlardan bir tanesi XML(Extensible Markup Language) ile veri tabanlarında bilgi tutma veya sistemler arası veri paylaşımı için kullanılmasıdır. Veri tabanında bir kişiye ait veriyi Emek, Güldoğan, 5534 şeklinde saklamak yerine

```
<isim>Emek</isim>
<soyisim>Güldoğan</soyisim>
<dosyano>5534</dosyano>
```

şeklinde bir standartlaşma ile veri saklama veya raporlamaya gidilirse kaydedilen veri her çeşit ortamda tutulabilir ve kullanılabilir. Bir işaretleme (markup) dili olmasından dolayı XML platformdan bağımsız çalışma olanağı da sağladığı için her işletim sisteminde, her disk ortamında sorunsuz kullanım olanağı sağlar.

ÖZET

TURGUT ÖZAL TIP MERKEZİ HASTANE OTOMASYONUNDAKİ LABORATUAR SONUÇLARININ SPSS PAKET PROGRAMINA AKTARILABİLMESİ İÇİN ARA YÜZ YAZILIMI GELİŞTİRİLMESİ

Bu çalışmada tıbbi veriler üzerinde veri madenciliği çalışılmıştır. Akademisyenlerin belirli kriterlere göre Turgut Özal Tıp Merkezi hastane otomasyonunda yer alan hastaların laboratuar sonuçlarını sorgulamalarını sağlamak ve sorgu sonuçlarını istatistik çalışmaların yapıldığı SPSS paket programında kullanılabilecek bir dosyasa haline getirmek için arayüz geliştirilmesi ve bu arayüz geliştirilirken kullanılan hastane bilişim sistemleri açıklanmıştır. Arayüz geliştirmede php programlama seçilmiş ve sorgulamalar tamamen internet üzerinden kullanıcı yetkilendirme ile yaptırılmıştır.

Anahtar Kelimeler: Veri Madenciliği, Hastane Bilişim Sistemleri, Veritabanları, SQL, Postgresql, PHP

ABSTRACT

DEVELOPMENT OF A GRAPHICAL USER INTERFACE TO EXPORT THE TURGUT OZAL MEDICAL CENTERS LABORATORY RESULTS FOR SPSS PACKAGE PROGRAM

In this work, data mining is worked on medical data. Some user interfaces are developed to use with SPPS package on laboratory results of Turgut Özal Medicine Center. On the other hand Hospital Information Systems are explained. User interfaces were created with PHP programming language and all queries made over Internet.

KEYWORDS: Data mining, Hospital Information System, Database, Postgresql, PHP.

6 Ekler

6.1 Ek 1 (index.php)

```
<?
//index.php
//Kullanıcı adı ve şifre girişinin yapılmasını sağlayan dosya
//Postgresql veritabanına bağlanmak için gerekli host,user,password
//tanımlarını içeren php dosyasının include edilmesi

include("config.php");

//Postgresql veritabanına bağlanmak için gerekli fonksiyonları içeren
//dosyanın include edilmesi

include("baglanti.php");

//Ekranda görüntüyü oluşturacak fonksiyonun tanımlanması
function ekran($mmm){

print "<form action=index.php method=post name=sanal>";

//Sanal klavyenin include edilmesi
include("sanal_klavye.php");
```

```
    print "Kullanıcı Adı:<input type=\"text\" name=\"kullanici_adi\">";
    print "Şifre: <input type=\"password\" name=\"eski\">";
    print "<input type=\"submit\" name=\"tamam\" value=\"TAMAM\">";
    print "</form>";

// Üyenin daha önceden girip girmediğinin kontrolü
if(!session_is_registered("username")){

$kullanici_adi=$HTTP_POST_VARS["kullanici_adi"];
$sifre=$HTTP_POST_VARS["eski"];

  if($HTTP_POST_VARS["tamam"]=="TAMAM")  {
     if($kullanici_adi=="" || $sifre=="") {
      $hata="<center><font color=\"#FF0000\"><b>
       Kullanıcı Adı ve Şifrenizi Giriniz</b></font></center>";
       ekran($hata);
     }
    else  {
          $sifre=md5($sifre);
        //Kullanıcı adı ve şifre kontolü
        $varmi1=@pg_exec("SELECT * FROM ogretim_uyesi_sifresi
           WHERE emekli_sicil_no='$kullanici_adi' and sifre='$sifre' ");

      //Kullanıcı adı ve şifre yanlışlığında yapılacak işlem
      if(@pg_numrows($varmi1)==0){

      $hata="<center><font color=\"#FF0000\">
```

```
        <b>Kullanıcı Adı ve Şifrenizi Doğru Giriniz</b></font></center>";
        ekran($hata);
        }
        //Kullanıcı adı ve şifre doğruluğunda yapılacak işlem
        else {
             //Bilgileri oturum yönetimine aktarma
             session_register("username");
             $username=$kullanici_adi;
            //Sorgulama penceresine yönlendirme
            Header("Location:sonuc.php");
             }
        }
        }
else    {
$hata="<br>";
//Görüntüyü oluşturan fonksiyonun çağrılması
ekran($hata);
}
}
//Üye olan bir kişinin ulaşacağı alana yönlendirme
else  Header("Location:dersler.php");
?>
```

6.2 Ek 2 (sonuc.php)

```
<?
//Sorgulama işleminin yapıldığı dosya

//Oturum yönetiminin başlaması
session_start();
//
set_time_limit(0);
//Postgresql veritabanına bağlanmak için gerekli host,user,password
//tanımlarını içeren php dosyasının include edilmesi
include("config.php");

//Postgresql veritabanına bağlanmak için gerekli fonksiyonları içeren
//dosyanın include edilmesi
include("baglanti.php");

//Üye olan bir kişini olup olmadığının kontrolü
if(session_is_registered("username")){

echo "
<html>
<head>
<meta http-equiv=\"Content-Language\" content=\"tr\">
<meta name=\"GENERATOR\" content=\"Microsoft FrontPage 5.0\">
<meta name=\"ProgId\" content=\"FrontPage.Editor.Document\">
<title>Yeni Sayfa 1</title>
```

```
<script language=javascript>
function kontrol(){
    baslangic_gun=document.formum.baslangic_gun.value;
    baslangic_ay=document.formum.baslangic_ay.value;
    baslangic_yil=document.formum.baslangic_yil.value;
    bitis_gun=document.formum.bitis_gun.value;
    bitis_ay=document.formum.bitis_ay.value;
    bitis_yil=document.formum.bitis_yil.value;

    uyari_mesaji='';

    if(baslangic_gun=='0' || baslangic_ay=='0' || baslangic_yil=='0'
        ||  bitis_gun=='0' ||  bitis_ay=='0' ||  bitis_yil=='0') {
        uyari_mesaji='Tarih alanlarını eksiksiz seçiniz';
        }

    if(document.formum.erkek.checked==false &&
                    document.formum.kiz.checked==false) {
         if(uyari_mesaji=='')
                uyari_mesaji='En az bir cinsiyet seçmelisiniz';
         else uyari_mesaji=uyari_mesaji+'.En az bir cinsiyet seçmelisiniz';
    }

    if(document.formum.test.value=='test'){
           if(uyari_mesaji=='')    uyari_mesaji='Bir test seçmelisiniz';
           else uyari_mesaji=uyari_mesaji+'.Bir test seçmelisiniz';
       }
```

```
    if(uyari_mesaji!='') {
        alert(uyari_mesaji);
        return false;
        }
    else {
        return true;

        }
    }
</script>

</head>
<body bgcolor=\"#C0C0C0\">

<form method=\"POST\" action=\"sonuc.php\"
                    onSubmit=\"return kontrol()\" name=formum>
    <div align=\"center\">
      <center>
      <font color=\"#000080\"><b>Aşağıdaki alanlardan doğum tarihi
aralığı, cinsiyet,test, bölüm ve hastalık belirleyiniz</b><br><br></font>

      <td width=\"41%\">
      <p align=\"center\"><b>DOĞUM TARİHİ</b></td>
      <td width=\"15%\">
      <p align=\"center\"><b>CİNSİYET</b></td>
      <td width=\"34%\" align=\"center\">
```

```
    <p align=\"center\"><b>TEST ADI</b></td>
  </tr>
  <tr>
    <td width=\"40%\">
      <tr align=center>
        <td width=\"40%\">Başlangıç</td>
        <td width=\"60%\">Bitiş</td>
      </tr>
      <tr >
        <td width=\"50%\" align=center>
                 <select size=\"1\" name=\"baslangic_gun\" >
                <option selected value=\"0\">Gün</option> ";

//Başlangıç doğum tarihi alanın gün kısmını oluşturma
         for($i=1;$i<33;$i++) {
          $secili="";
          if($i==$_POST["baslangic_gun"]) $secili="selected";
          if($i<10) $i="0".$i ;
          echo   "<option value=\"$i\" $secili>$i</option>";
          }

         echo "
         </select><select size=\"1\" name=\"baslangic_ay\">
         <option value=\"0\">Ay</option> ";

//Başlangıç doğum tarihi alanın ay kısmını oluşturma
        $aylar=array("Ocak","Şubat","Mart","Nisan",
              "Mayıs","Haziran","Temmuz","Ağustos",
```

```
            "Eylül","Ekim","Kasım","Aralık");
          for($i=1;$i<13;$i++) {
           $secili="";
           if($i==$_POST["baslangic_ay"]) $secili="selected";
           $k=$i-1;
           if($i<10) $i="0".$i ;
           echo   "<option value=\"$i\" $secili>$aylar[$k]</option>";
           }

          echo "
          </select><select size=\"1\" name=\"baslangic_yil\">
                <option value=\"0\">Yıl</option>";

//Başlangıç doğum tarihi alanın yıl kısmını oluşturma

   $bu_yil=date("Y");

    for($i=1920;$i<=$bu_yil;$i++) {
        $secili="";
        if($i==$_POST["baslangic_yil"]) $secili="selected";
            echo   "<option value=\"$i\" $secili>$i</option>";
    }
          echo "
          </select></td>
          <td width=\"60%\" align=center>
                <select size=\"1\" name=\"bitis_gun\">
          <option selected value=\"0\">Gün</option> ";
```

```
//Bitiş doğum tarihi alanın gün kısmını oluşturma
          for($i=1;$i<33;$i++) {
           $secili="";
           if($i==$_POST["bitis_gun"]) $secili="selected";
           if($i<10) $i="0".$i ;
           echo   "<option value=\"$i\" $secili>$i</option>";
           }

          echo "
</select><select size=\"1\" name=\"bitis_ay\">
          <option value=\"0\">Ay</option> ";

//Bitiş doğum tarihi alanın ay kısmını oluşturma

        $aylar=array("Ocak","Şubat","Mart","Nisan","Mayıs",
              "Haziran","Temmuz","Ağustos","Eylül",
              "Ekim","Kasım","Aralık");
          for($i=1;$i<13;$i++) {
           $secili="";
           if($i==$_POST["bitis_ay"]) $secili="selected";
           $k=$i-1;
           if($i<10) $i="0".$i ;
           echo   "<option value=\"$i\" $secili>$aylar[$k]</option>";
           }

          echo "
          </select><select size=\"1\" name=\"bitis_yil\">
          <option value=\"0\">Yıl</option>";
```

```
//Bitiş doğum tarihi alanın yıl kısmını oluşturma
    for($i=1920;$i<=$bu_yil;$i++) {
        $secili="";
        if($i==$_POST["bitis_yil"]) $secili="selected";
            echo   "<option value=\"$i\" $secili>$i</option>";
    }

//Cinsiyet alanının oluşturulması
    if($_POST["erkek"]!="")  $erkek_secili="checked";
    else $erkek_secili="";

    if($_POST["kiz"]!="") $kiz_secili="checked";
    else $kiz_secili="";

          echo "

          </select></td>
        </tr>
      </table>
      </td>
      <td width=\"18%\">
          <p align=\"center\">Erkek
          <input type=\"checkbox\" name=\"erkek\"
              $erkek_secili value=\"ON\"></td>
        </tr>
        <tr>
          <td width=\"100%\">
```

```
        <p align=\"center\">Kadın<input type=\"checkbox\"
             name=\"kiz\" $kiz_secili value=\"ON\"></td>
      </tr>
    </table>
    </td>
    <td width=\"7%\">
    <p align=\"center\"><select size=\"1\" name=\"test\">
    <option value=\"test\">Test Adı</option>";

//Test alanın oluşturulması
    $sonuc=pg_exec("select distinct r_testadi from lis_result_tmp
                    where r_testadi is not null order by r_testadi");
    $satir=pg_numrows($sonuc);

    for($i=0;$i<$satir;$i++){
        $r_testadi=pg_result($sonuc,$i,"r_testadi");
        $test_secili="";
        if($r_testadi==$_POST["test"])   $test_secili="selected";
        echo "<option value=\"$r_testadi\" $test_secili>
                $r_testadi</option>";
        }

    echo "</select></td></tr></table></center><br>

      <tr>
        <td width=\"33%\" align=\"center\">
    <p align=\"center\"><b>BÖLÜM ADI</b></td>
        <td width=\"33%\" align=\"center\">
```

```
<p align=\"center\"><b>HASTALIK ADI</b></td>
  </tr>
  <tr>

    <td width=\"7%\">
<p align=\"center\"><select size=\"5\" name=\"bolum[]\" multiple>";

//Bölümler alanının oluşturulması

$sonuc=pg_exec("select distinct t_bolumadi from lis_result_tmp
       where t_bolumadi is not null order by t_bolumadi");
$satir=pg_numrows($sonuc);

for($i=0;$i<$satir;$i++){
    $t_bolumadi=pg_result($sonuc,$i,"t_bolumadi");
        $bolum_secili="";
        $tum_bolum_secili="";
        for($k=0;$k<$satir;$k++){
            if("Tüm Bölümler"==$bolum[$k])
                        $tum_bolum_secili="selected";
            if($t_bolumadi==$bolum[$k]) {
                $bolum_secili="selected";
                break;
            }
        }

    if($i==0) {
        echo "<option value=\"Tüm Bölümler\"
```

```
                $tum_bolum_secili>Tüm Bölümler</option>";
        }
        echo "<option value=\"$t_bolumadi\" $bolum_secili>
              $t_bolumadi</option>";

        }

    echo "</select></td>
        <td width=\"7%\">
    <p align=\"center\"><select size=\"5\" name=\"hastalik[]\" multiple>";

//Hastalıklar alanın oluşturulması

    $sonuc=pg_exec("select distinct t_tani from lis_result_tmp
                    where t_tani is not null order by t_tani");
    $satir=pg_numrows($sonuc);

    for($i=0;$i<$satir;$i++){
        $t_tani=pg_result($sonuc,$i,"t_tani");

            $tani_secili="";
            $tum_tani_secili="";
            for($k=0;$k<$satir;$k++){
                if("Tüm Hastalıklar"==$hastalik[$k])
                      $tum_tani_secili="selected";
                if($t_tani==$hastalik[$k]) {
                    $tani_secili="selected";
                    break;
```

```
                }
            }
        if($i==0) {
            echo "<option value=\"Tüm Hastalıklar\"
              $tum_tani_secili>Tüm Hastalıklar</option>";
        }

        echo "<option value=\"$t_tani\" $tani_secili>$t_tani</option>";

        }

    echo "</select></td></tr></table>

  <p align=\"center\"><input type=\"submit\"
          value=\"TEST SONUÇLARINI GÖSTER\" name=\"tamam\"></p>
</form>";

echo "</td></tr><tr><td width=\"100%\" bgcolor=\"#999999\">";

//TEST SONUÇLARINI GÖSTER butonuna  basıldığında gereçekleşecek kısım
if($tamam!=""){

//Sonuçları yazdıracağımız dosyanın açılması
$dosya=fopen("sonuc.rtf","w+");

//Seçilen doğum tarihi aralığının belirlenmesi
$baslangic_tarihi=$_POST["baslangic_yil"]."-".
            $_POST["baslangic_ay"]."-".$_POST["baslangic_gun"];
```

```
$bitis_tarihi=$_POST["bitis_yil"]."-".
              $_POST["bitis_ay"]."-".$_POST["bitis_gun"];
$secilen_test_adi=$_POST["test"];

$ekran_baslangic_tarihi=$_POST["baslangic_gun"]."
  .".$_POST["baslangic_ay"].".".$_POST["baslangic_yil"];
$ekran_bitis_tarihi=$_POST["bitis_gun"].".".$_POST["bitis_ay"].
                    ".".$_POST["bitis_yil"];


//Seçilen cinsiyetin belirlenmesi
$cinsiyet_sorgu="";
if($_POST["erkek"]!="" && $_POST["kiz"]!="") {
    $cinsiyet_sorgu=" and (t_cinsiyet='1' or t_cinsiyet='2') ";
    $ekran_cinsiyet='Erkek ve Kadın ların';
    }
else if($_POST["erkek"]!="") {
     $cinsiyet_sorgu=" and t_cinsiyet='1' ";
     $ekran_cinsiyet='Erkek lerin';
}
else {
    $cinsiyet_sorgu=" and t_cinsiyet='2' ";
    $ekran_cinsiyet='Kadın ların';
}

//Seçilen bölümlerin belirlenmesi
$m=0;
while($bolum[$m]!=""){
```

```
            if($bolum[0]=="Tüm Bölümler") {
                $ekran_bolum="Tüm Bölümlerden";
                $secilen_bolum_sorgusu="";
                break;
            }
            else   {
                if($m!=0) {
                    $secilen_bolum_sorgusu.= " or ";
                    $ekran_bolum.=",";
                    }
                $secilen_bolum_sorgusu.=" t_bolumadi='".$bolum[$m]."'";
                $ekran_bolum.=$bolum[$m];
                }
            $m++;
}
if($ekran_bolum!="Tüm Bölümlerden" and $m==1)
                $ekran_bolum.=" Bölümünden";
if($ekran_bolum!="Tüm Bölümlerden" and $m!=1)
                $ekran_bolum.=" Bölümlerinden";
$ekran_bolum.=" İstenilen";

//Seçilen hastalıkların belirlenmesi
$m=0;
while($hastalik[$m]!=""){

            if($hastalik[0]=="Tüm Hastalıklar") {
                $ekran_hastalik="Tüm Hastalıklar";
```

```
            $secilen_hastalik_sorgusu="";
            break;
        }
        else   {
            if($m!=0) {
                $secilen_hastalik_sorgusu.= " or ";
                $ekran_hastalik.=",";
                }
            $secilen_hastalik_sorgusu.=" t_tani='".$hastalik[$m]."'";
            $ekran_hastalik.=$hastalik[$m];
            }
        $m++;
}

if($ekran_hastalik!="Tüm Hastalıklar" and $m==1)
                    $ekran_hastalik.=" hastalığı";
if($ekran_hastalik!="Tüm Hastalıklar" and $m!=1)
                    $ekran_hastalik.=" hastalıkları";
$ekran_hastalik.=" için";

if($secilen_bolum_sorgusu!="")
        $secilen_bolum_sorgusu="and ($secilen_bolum_sorgusu)";
if($secilen_hastalik_sorgusu!="")
        $secilen_hastalik_sorgusu="and ($secilen_hastalik_sorgusu)";

//Sorgu kriterlerine göre veritabanından sorgu yapılması
$sonuc=pg_exec("select distinct o_barkod from lis_result_tmp where
      t_dogtar>='$baslangic_tarihi' and t_dogtar<='$bitis_tarihi'
```

```
        $cinsiyet_sorgu and r_testadi='$secilen_test_adi'
          $secilen_bolum_sorgusu  $secilen_hastalik_sorgusu");
$satir=pg_numrows($sonuc);


//Yapılan sorgulamaya uygun sonuçlarının ekrana yazdırılması
else {
     echo "<center><b>TEST SONUÇLARI</b></center>

<center><font color=\"#FF0000\">$ekran_bolum $ekran_hastalik<br>
Doğum tarihi $ekran_baslangic_tarihi-$ekran_bitis_tarihi
arasında olan  $ekran_cinsiyet $secilen_test_adi test sonuclari </font><br>
<a href=\"sonuc.rtf\">Sorgu sonuçlarınızı bilgisayarınıza kaydedin</a><br>

</center>
";

$sonuc1=pg_exec("select r_sonuc_adi from lis_result_tmp
    where r_testadi='$secilen_test_adi'  group by r_testadi,
        r_sonuc_adi order by r_sonuc_adi");
$satir1=pg_numrows($sonuc1);

      for($i=0;$i<$satir1;$i++){
          $r_sonuc_adi=pg_result($sonuc1,$i,"r_sonuc_adi");
          //if($satir1-1==$i) $dosyaya_yazilacak.=$r_sonuc_adi."\n";
          //else $dosyaya_yazilacak.=$r_sonuc_adi.",";
          echo " <td>$r_sonuc_adi</td>";
          }
```

```
    echo "</tr><tr>";

     for($i=0;$i<$satir;$i++){
         $o_barkod=pg_result($sonuc,$i,"o_barkod");
         $s=$i+1;

         echo " <td width=\"10%\" align=\"center\">$s</td>";

         for($m=0;$m<$satir1;$m++){

        $r_sonuc_adi=pg_result($sonuc1,$m,"r_sonuc_adi");
        $sonuc2=@pg_exec("select * from lis_result_tmp
            where o_barkod='$o_barkod' and r_testadi='$secilen_test_adi'
                        and r_sonuc_adi='$r_sonuc_adi'");
        $r_deger_num=@pg_result($sonuc2,0,"r_deger_num");
        echo " <td> $r_deger_num</td>";
         }

 echo " </tr> ";

}
//Sonucların dosyaya yazılması
//fwrite($dosya,$dosyaya_yazilacak);
//sonuc.rtf dosyasının kapatılması
fclose($dosya);

echo "</table></td></tr></table>";
```

```
    }
}
echo "</body></html> ";
}
//Üye olmayan kişinin bu kısma erişmeye çalıştığında yapılacak işlem
else  echo Header("Location:index.php");
 ?>
```

7 Kaynaklar

[1] M. Kane. *Genomics &. Proteomics*, 4:10, 2005.

[2] E. Pampalka, G. Widmera, and A. Chanc. *Intelligent Data Analysis*, 8:131, 2004.

[3] J. Barrett, A. Kostadinova, and J. A. Raga. *TRENDS in Parasitology*, 21:207, 2005.

[4] V. Sonek. *http://www.pclabs.gen.tr/article.asp?doc=210&page=4*, 2003.

[5] K.Kovar, J. Furnkraz, J. Petrak, and B. Pfahringer. *Cybernetics and Systems: An International*, 31:649, 2000.

[6] D. J. Hand. *Statistical Methods in Medical Research*, 9:305, 2000.

[7] P. Smyth. *Statistical Methods in Medical Research*, 9:309, 2000.

[8] I. N. Lee, S. C. Liao, and M. Embrechts. *med. inform.*, 25:81, 2000.

[9] R. H.L. Chiang, C. E. H. Cecil, and E. P. Lim. *Data & Knowledge Engineering*, 53:311, 2005.

[10] N. Lavrac. *Artificial Intelligence in Medicine*, 16:3, 1999.

[11] K. J. Ciosa and G. W. Mooree. *Artificial Intelligence in Medicine*, 26:1, 2002.

[12] S. C. Liao and I. N. Lee. *Med. Inform.*, 27:59, 2002.

[13] T. P. Nadeau, K. A. Sullivan, T. j. Teorey, and E. L. Feldman. *Journal of Integrative Neuroscience*, 2:201, 2003.

[14] F. Lemke and J. A. A. Mueller. *Systems Analysis Modelling Simulation*, 43:1399, 2003.

[15] L. Sokol, B. Garcia, J. Rodriquez, M. West, and K. Johnson. *Top Health Inform Manage*, 22:1, 2001.

[16] A. Genkin and C. A. Kulikowski. *Journal of Intelligent and Fuzzy Systems*, 12:5, 2002.

[17] S. Doddi, A. Marathe, S. S. Ravioe, and D. C. Torney. *Med. Inform.*, 26:25, 2001.

[18] D. Kreuze. *Technology Review*, 3:32, 2001.

[19] D. Gündüz. *http://www.gunduz.org/seminer/pg*, 2000.

[20] M. H. Dilek. *http://www.ulakbim.gov.tr/dokumanlar/programlama/2000php/*, 2000.

Özgeçmiş

26.04.1978 tarihinde Malatya'da doğdu.İlkokulu Derme İlköğretim Okulunda, Ortaokul Malatya Anadolu Lisesinde,liseyi Malataya Fen Lisesinde okudum. Orta Dogu Teknik Üniversitesi Bilgisayar Mühendisliğinden 26.06.2001 tarihinde mezun oldum. 26.09.2001 tarihinden beri İnönü Üniversitesi Enformatik Bölümünde okutman olarak görev yapmaktayım. 2002-2003 eğitim yılı güz yarıyılda İnönü Üniversitesi Sağlık Bilimleri Enstitüsü Biyoistatistik Anabilimdalında yüksek lisansa başladım.

Sürekli Adres: İnönü Üniversitesi, Enformatik Bölümü, 44280,
Malatya, eguldogan@inonu.edu.tr