T.C.
AKDENİZ ÜNİVERSİTESİ
SAĞLIK BİLİMLERİ ENSTİTÜSÜ
BİYOİSTATİSTİK ANABİLİM DALI

AKDENİZ ÜNİVERSİTESİ TIP FAKÜLTESİ HASTANESİ İÇİN GELİŞTİRİLEN HASTANE OTOMASYON SİSTEMİNİN DAĞITIK VERİTABANI MODELİ İLE TASARLANMASI

YÜKSEK LİSANS TEZİ

ŞEYHMUS ALTUN

DANIŞMAN ÖĞRETİM ÜYESİ
YRD.DOÇ.DR.MEHMET YARDIMSEVER

ANTALYA,1997

891

İÇİNDEKİLER *Sayfa*

Şekiller Listesi

Tablolar Listesi

1.GİRİŞ VE AMAÇ

Geleneksel sistem geliştirme stratejisinde veriler ve uygulama yazılımları ana bilgisayarda (mainframe) yönetilir ve bir çok kullanıcı tek merkezden uygulamalarını işletirler. Son kullanıcılar, akılsız terminal (Dummy Terminal) denilen işlem yapma yeteneğinden yoksun araçlar vasıtasıyla uygulamalarını işletirler. Bu sistem, donanım ve yazılım bağlılığını beraberinde getirmiş olup kuruluşlara kısıtlı imkanlar sunmuştur.

Küçük ve orta boy bilgisayarlarda kullanılabilen hızlı ve birim sürede birden çok mikro komutu işletebilen RISC tabanlı mikro işleyicilerin ortaya çıkması ve platform/donanım bağlılığını ortadan kaldıran açık sistemlerin yaygınlaşmasıyla merkezi sistemler yerini dağıtık modüler sistemlere bırakmaya başlamıştır. Buna SMP (Symetrical Mutiple Processing) denilen, bir bilgisayarda birden çok RISC tabanlı mikroişleyicinin bulunması ve bunların paralel çalışmasını sağlayan teknoloji eklenince düşük maliyetli, yüksek performanslı sistemler geliştirmek mümkün olmuştur.

Bilgisayar ağı teknolojisindeki ve yukarıdaki sözü edilen gelişmelere istemci/sunucu (Client/Server) mimarisi de eklenince veri, işlem ve yordam (Remote Procedure Call-RPC) paylaşımı da gündeme gelmiştir. Dolayısıyla birim sürede birden çok işlemin gerçekleşmesi mümkün olmuştur.

Tüm bu yenilikler veritabanı sistemlerini de etkilemiştir. Tür ve yapılarına göre veri ve işlemlerin dağıtılmasına dayanan dağıtık veri tabanı modeli yaygınlaşmaya başlamıştır. Bu modelde veri tabanının fiziksel görünümü dağıtık yapıda olmasına karşılık mantıksal açıdan veritabanı, geleneksel merkezi veritabanı gibi görünmektedir. Veritabanının dağıtık yapıya kavuşturulması ve istemci/sunucu mimarisinin olanaklarından yararlanılarak uygulamaların da dağıtık yapıya kavuşturulmasıyla performansı daha arttırmak mümkün görünmektedir. Dolayısıyla yazılımların modüler yapıda geliştirilmesi gerekmektedir. Tüm bu gelişmelerin kuşkusuz birçok avantajı ve dezavantajı vardır. Ancak en çok göze çarpan özelliğin sistemin ve yazılımların sonradan ortaya çıkan yeni gereksinimlerin, sistem iş akışını bozmadan ilgili modülü etkilemesiyle karşılanmasıdır. İyi bir sistem çözümleme stratejisiyle, bakım ve genişleyebilirlik maliyeti önemli derecede düşmektedir. Ancak iyi olmayan bir sistem çözümleme çalışması bu maliyeti merkezi sisteme göre daha da

yükseltmektedir. Diğer bir deyişle dağıtık sistem farklı uzmanlıkları (Veritabanı yöneticisi, uygulama geliştirme uzmanları, bilgisayar ağı uzmanı, sistem mühendisi, sistem çözümleyici,vb.) gerektiren iyi organize olmuş bir bilgi işlem personeli gereksinimini doğurmaktadır.

Bu çalışmada, Akdeniz Üniveristesi Tıp Fakültesi Hastanesi için geliştirilen Hastane Bilişim Sisteminin istemci/sunucu mimarisi çerçesinde dağıtık veritabanı modeli ile gerçekleştirilmesi için bir tasarım modeli önerilmektedir. Bu sistem, Windows NT sunucuları üstünde yüklü dağıtık bir ORACLE veritabanı ile DOS ve Windows (3.1 ve 95) temelli istemciler üstünde çalışan Clarion ve Visual Basic ile geliştirilmiş istemci arayüzlerinden oluşmaktadır. Sunucular ve istemciler birbirleriyle TCP/IP temelli ORACLE SQL*NET ile haberleşmektedirler. Ancak Hastane Bilişim Sistemi bir açık sistem ve modüler olarak düşünüldüğünden sözü edilen platformun dışındaki yapılar ve şu anda işleyen veritabanı ve uygulamalar sisteme uyarlanabilir.

Bu yapıyı uygulayabilmek için Akdeniz Üniversitesi Hastanesinin sistem çözümlemesi bir Hastane Bilişim Sisteminin(HBS) özellikleri doğrultusunda gerçekleştirilmiştir ve donanım/yazılım ihtiyacının optimum olması için bir çalışma yapılmıştır. Bu çalışmalar ışığında oluşan veritabanı, işlem ve verilerin tür, yapı ve yoğunluklarına göre belirlenen donanımlara dağıtılması gerçekleştirilmiştir. Ayrıca uygulama arayüzlerinin dağıtık veritabanlarıyla iletişimini kurmak için gerekli olan en uygun yazılımın hangisinin olabileceği konusunda bir tartışma yapılmaktadır.

2. GENEL BİLGİLER

2.1. Veritabanı (Database)

Veritabanı(VT) için günümüzde herkesin kabul ettiği genel bir tanım yoktur. Yapılan tanımlar içinde en çok kabul görenleri şu şekilde sıralayabiliriz:

1. Date'e göre, "Veritabanı bir işletmenin uygulama dizgelerince kullanılan bellekte saklı ve işler durumdaki veriler bütünüdür".

2. Martin'e göre,"Veritabanı bir yada birçok uygulamada kullanılmak için,gereksiz yinelemelerden arınmış olarak bellekte saklanan ve birbirleriyle ilişkili veriler topluluğu olarak tanımlanabilir".

3. Ashany'e göre,"Veritabanın dizgelerinin çalışma alanı verilerin çözümlenmesi, yorumlanması, düzenlenmesi, sınıflandırılması, kaydedilmesi, günlenmesi, taranması ve bulunmasıdır"[1].

Bu tanımlarla VT'nın birçok özelliği belirtilmektedir:

1. Buna göre bir veritabanı belirli bir işletmenin (hastane, banka, üniversite, fabrika,...vb. herhangi bir kamu ya da özel kuruluşu) verilerinden oluşur. Günümüzde en çok kullanılan veritabanı modeli olan ilişkisel veritabanı modelinde her veri, alanların (Field) oluşturduğu kayıtları (Record) barındıran talolardan oluşur.

2. VT birbirleriyle ilişkili işletimsel veriler ve veritabanı hakkındaki verilerden (Metadata) oluşur, giriş/çıkış verileri veya geçici ara veriler içermez. VT'nı oluşturan veriler birbirleriyle ilişkili verilerdir; başka deyişle VT'da yalnız veriler değil aynı zamanda iş akışını belirleyen veriler arasındaki ilişkilerde saklanmaktadır. Bu da işletmenin bilgi gereksinimlerini karşılar[1]. İlişkilerine göre veriler bir araya gelerek arayüzleri(view) dediğimiz kayıtlı sorguları(stored query) oluştururlar. Bunlar bir veya birden çok tablonun birleştirilmesiyle(join) yaratılır.

3. Veritabanı işletme kurallarını, kısıtları ve bazı işlevleri bir veri olarak saklar.

2.1.1. Veri Tabanının Sunduğu Avantajlar

Bir tam işlevli VTYS'ne (full-FUNCTION DBMS) bağlı bir sistem aşağıdaki avantajlara sahip olacaktır:

1. Minimum Veri Tekrarı(Minimum Data Redundancy) Veri tabanı yaklaşımında, birbirinden ayrı olarak ve tekrarlı verileri içeren dosyalar birbirleriyle mantıksal bir bütün oluşturur ve yinelemelerden kurtarılır. Her veri VT'da sadece bir yerde kayıt olarak tutulur. Bazı durumlarda (örneğin veriye erişimi hızlandırmak ve etkinleştirmek için) yinelemelere maruz kalınır, ancak VT'da bu yinelemeler denetim altında tutulur.[2] Sistemin genel performansı arttırılır ve sistem yinelemelere karşı korunmuş olur. Örneğin bir hastanın hastanedeki kaydı sadece bir yerde tutulur ve aynı dosya numarasına karşılık bir kayıt tutulur. Sistemin daha hızlı çalışması için VT yöneticisi bu kayıtların kopyalarını işletmenin çeşitli makinalarda tutabilir.

2. Verinin Tutarlığı(Data Consistency) Verilerdeki tekrarları kaldırarak verinin bütünlüğü sağlanmış olur. Bir kayıt üzerinde yapılan her türlü değişiklik tüm veri tabanına yansır.[2]Verinin tutarlılığını sağlamak için VTYS şu işlemleri gerçekleştirir:

- Veri üzerinde belirli bir anda yapılan işlemler bütünü(transaction) sonuçlanmadan veri üzerinde değişiklik yapılmamasını(tutarlılık) garantiler.
- Aynı veri üzerinde sorgulama(okuma) yapanlar, yazma(günleme,ekleme,silme) işlemini gerçekleştirenleri veya diğer sorgulama yapanları beklemez.
- Aynı veri üzerinde yazma yapanlar, sorgulama yapanları beklemez.
- Aynı veri üzerinde yazma yapanlar yine aynı veri üzerinde yazma yapanların işlemlerini bitirmelerini bekler.[3]

Örneğin hastane içinde öğretim üyelerine muayene için bir kişi rezervasyon yaptırdıysa aynı anda diğer bir kişinin de aynı saate rezervasyon yapması engellenmelidir.

3 Verinin Bütünlüğü(Integrity of Data) Kullanıcıların tanımladığı işletim kurallarının, VTYS ve uygulamalar tarafından denetlenmesi ve garanti edilmesi.[3] VT'daki veriler, kısıtlarıyla (birbirleri arasındaki ilişkiler,anahtarlar,...) birlikte mantıksal bir bütün oluştururlar, bu ise kullanıcılara veriler arasındaki ilişkileri tanımlama ve kısıtları kullanma olanağı sunar[2]. Örneğin hastane içindeki öğretim üyelerinin kişisel bilgileriyle birlikte hastane içinde ürettikleri hizmetler bu özellik sayesinde sorgulanabilir veya depodan 1500 adetten fazla malzemelerin birimlere dağıtımının

engellenmesi gibi kurallar veri tabanı yönetim sistemi tarafından denetlenir. Aşağıda ORACLE VTYS tarafından sunulan ve hemen her VTYS'de bulunan kısıtlar görülmektedir:

- NOT NULL: Bu kısıtla tanımlı bir veri alanına mutlaka bir değer girilmelidir. Örneğin bir hastanın, kayıt dosyasında adı ve soyadı mutlaka olmalıdır,boş geçilmesi engellenmelidir.
- UNIQUE : Unique olan bir veri alanı(veya bir dizi veri alanı) tanımlı olduğu nesne içindeki kayıtlardan en çok bir kayda karşılık gelir. Örneğin hastane içinde kayıtlı bir hastanın nüfus cüzdan numarası sorgulandığında (UNIQUE veri alanı) tüm hasta dosyaları içinde en çok bir kayda rastlanır,çünkü aynı nüfus cüzdan numarasına sahip birden fazla kişi bulunamaz. Bu veri alanı, NOT NULL tanımlanmadığı sürece boş değer taşıyabilir,yani bir hastanın nüfus cüzdan numarası bilinemeyebilir. UNIQUE tanımlı bir veri alanı(veya bir dizi veri alanı) VT'nında indeksin yaratılmasına neden olur.
- PRIMARY KEY: Primary Key olarak tanımlı veri alanı (veya bir dizi veri alanı) tanımlı olduğu nesnenin kayıtları için tekliği tanımlar, dolayısıyla birbirinin aynı iki kayıt bulundurmazlar. Tıpkı UNIQUE olma gibi primary Key sorgulamalarda tek bir kayıt döndürür. Örneğin hizmet kodu L1301 olan hizmet sorgulandığında buna karşılık en fazla bir kayda rastlanır. Primary key olarak tanımlı olan alan(lar) boş geçilemezler, bu alanlar aynı zamanda NOT NULL kısıtına da sahiptirler ve bir indeksin yaratılmasına neden olurlar.
- FOREIGN KEY: Veriler arasındaki ilişkilerin tanımlanmasını sağlayan kısıtlardır.Örneğin hasta faturaları ile o faturayı oluşturan hizmetler arasındaki ilişkiler. Her faturanın bir veya birden çok hizmeti vardır.
- CHECK : Yukarıda anlatılan kısıtların dışında bazı veri alanları özel kurallara tabii tutulabilirler Örneğin hastaların cinsiyeti ya erkektir ya da kadın, bunun dışında bir cinsiyet yoktur. İlgili veri alanına bir CHECK kısıtı konulursa bu denetim VTYS tarafından denetlenir.

4 Veri Paylaşımı (Sharing of Data). Bir veri tabanındaki her türlü veri kullanıcıların erişim haklarına göre eşanlı olarak erişimlidir. Her birim kendisi için gerekli olan verilere arayüzler(view) yoluyla aynı anda erişebilir üzerinde gerekli işlemleri yukarıda sözü edilen olanaklarla gerçekleştirebilirler[2]

5.Uygulama Geliştirmede Kolaylık(Ease of Application Development): VT kullanımının en önemli avantajlarından bir tanesi de VT'na bağlı kullanıcılar tarafından çalıştırılacak olan uygulama programlarının 4.Kuşak programlama dillerinin (4^{th}Generation Language) olanakları sayesinde maliyetlerinin düşük olması ve kısa sürede geliştirilebilmeleridir.[2]

6.Güvenlik(Security): Çok kullanıcılı VT sistemleri, VT erişimi ve kullanımı ile ilgili güvenliğe ilişkin olanaklara sahiptirler. Güvenlik mekanizması genellikle şu temel işlemleri yerine getirir:

- Yetkisiz VT erişiminin engellenmesi.
- VT nesnelerine yetkisiz erişimin engellenmesi.
- Disk kullanımının denetlenmesi.
- Sistem kaynaklarının(Merkezi İşlem Birim) kullanımının denetlenmesi.
- Kullanıcıların işlemlerinin izlenmesi.
- Kullanıcıların yarattığı VT nesnelerinin boyutlarının sınırlandırılması(quota) ve denetlenmesi.
- Her VT nesnesi için gerekli erişimlerin tanımlanması.Örneğin hastanedeki bir veznedar bir hastanın sadece kaydını görebilir,izleyebilir,ancak üzerinde değişiklik yapamaz.
- Kullanıcılara rollerin atanması.Roller belirli işlemlerin yerine getirilmesi için gereken yetkiler bütünüdür[3].

7.Çöküntülere Karşı Korunma ve Yedek Alma(Recovery and Backup): Her VT'da sistem veya donanımsal çöküntülerin oluşması olasıdır(ani elektrik kesilmesi,disk arızası,....v.b.). Bir çöküntü oluştuğunda ve bundan VT etkilendiyse VT'nın tekrar eski haline dönüştürülmesi gerekmektedir[3].Aksi taktirde yukarıda bahsedilen VT'nın bütünlük ve tutarlılık gibi önemli olanakları karşılanmamış olur. Tamamlanmayıp da kesilen işlemler(transaction) kaldıklara noktaya geri getirilmeli ve VT kaldığı yerden güvenli bir şekilde devam ettirilmelidir. Kesilen işlemler geri getirilemiyorsa bunlar VT'na işlenmeden geri alınmalıdır(Rollback).Büyük verilerin bulunduğu sistemlerde bu özellik son derece önemlidir.

Disk kapasitelerinin sınırlı olması ve donanımsal çöküntülerin VT'daki verilerin kaybolmasına yol açacağından yedek alma gereksinimini doğurmaktadır.VT sistemlerinde yedek alma kolaylığı önemli bir unsur olarak karşımıza çıkmaktadır.

Çöküntülerden sonra veri kullanılamıyorsa VT'nın bu olanağını kullanarak son yedeği alınan veriler sisteme kolay bir şekilde tekrar yüklenebilir.

8.Veri Erişiminde Kolaylık(Ease Of Data Access). Bir VT sisteminde kullanıcı gereksinimlerinin sık sık değiştiği gözönüne alınırsa veriye erişimin kolay bir şekilde gerçekleştirilmesi isteği doğmaktadır. Yapısal Sorgulama Dili (Structured Query Language-SQL) denilen 4.Kuşak bir sorgulama diliyle her türlü veriye kısa zamanda kolay bir şekilde erişme imkanı doğmaktadır. Örneğin bir hastanedeki laboratuarda üretilen hizmetlerin adlarını ve fiyatlarını görmek için şu SQL cümleciğinin yazılması yeterli olacaktır.

SELECT ad,tutar FROM hizmet WHERE hiztur='LAB',

9.Veri Bağımsızlığı(Data Independence). Verilerinin tanımlarıyla, verilere ilişkin olan uygulama programlarının birbirinden ayrılması olayına veri bağımsızlığı denir. Veri tanımlarının zamanla değişmesi ve gelişmesi veriyi kullanan uygulamaların değişmesine gerek kalmamaktadır. VT'da veriler uygulamalardan bağımsız bir şekilde VTYS ve VT yöneticisi denilen kişi(ler) tarafından yönetilmektedir. Bu aynı zamanda uygulama programlarının bakım maliyetini düşürmekte ve zaman kaybını minimuma indirgemektedir[2].

2.1.2. Veri Tabanı Geliştirmenin Maliyetleri

Yukarıda sözü edilen avantajların dışında bir VT sisteminin maliyetleri ve riskleri vardır:

1. Bir VTYS diğer uygulama geliştirme araçlarına göre daha pahallıdır.
2. Veritabanı sistemleri, öncelikle VT yönetimi için gerekli özel eğitimli personel ihtiyacını yaratmaktadır. Bu, kuruluş için ek bir personel alımını ve eğitimini gerektirmektedir. VTYS kullanmayan bir kuruluş, birkaç programcı ile sistem geliştirebilir, ancak VTYS için farklı konularda uzmanlaşmış bir ekibe (hatta bilgi işlem birimine) gereksinim duymaktadır.
3. Veri tabanının donanımsal çöküntülere karşı korunması için öncelikle verilerin dış ortamda yedeklerinin alınması(backup) gerekmektedir. Bu, kuruluş için ek bir iş yükü demektir. Ancak istenmeyen nedenlerden ötürü oluşacak olan veri kayıplarının ortadan kaldırılması ancak bu yolla engellenebilir.

4. Verilerin paylaşılması sırasında çakışmalar oluşabilir. Çakışma bazen istenmeyen kilitlenmelere(deadlock) yol açabilir[3]. Bu durum farklı kişilerin aynı anda aynı kayıt üzerinde günleme işlemlerini yaptıkları sıralarda oluşur ve işlemlerin takılmasına ve sonuçlanmamasına neden olur. Tablo-1'de iki transaction sırasında, 1 nolu zaman noktasında herhangi sorun yok, ancak 2 nolu zaman diliminde 1 nolu transaction işlemini tamamlamadan 2 nolu transaction'nın kilitlediği(lock) veri üzerinde günleme yapmak istemektedir, aynı durum 2 nolu transaction için de geçerlidir. Sonuç olarak her iki transaction birbirlerinin işlemlerini tamamlamasını bekler ve kilitlenirler. 3 nolu zaman noktasında VTYS bir hata mesajı yollar ve her iki işlemi keser. Bu durumları önlemek için daha farklı yöntemlerle işlemlerin yapılması gerekmektedir.

Transaction 1	Zaman Noktası	Transaction 2
UPDATE stok SET giris=giris+200 WHERE hizkod=100	1	UPDATE stok SET cikis=cikis+500 WHERE hizkod=21
UPDATE stok SET giris=giris+100 WHERE hizkod=21	2	UPDATE stok SET cikis=cikis+300 WHERE hizkod=100
ERROR:Deadlack while waiting for resource	3	

Tablo 1: İki Transaction'nın kilitlenmesi(Deadlock)

5. Bir VT sistemi farklı işlemler yapan birçok kullanıcıya aynı anda hizmet verdiğinden, kullanıcılara ilişkin yetkilerin ve rollerin iyi tanımlanması gerekmektedir. Bunun sağlıklı bir şekilde yapılması için öncelikle iş tanımlarının ve kuruluşun organizasyon şemasının iyice incelenmesi gerekmektedir. Örneğin hastanedeki bir bölüm sekreterinin personellerin döner sermaye verilerine erişmesi engellenmelidir. Bu özellik ek bir iş yüküne neden olmaktadır.

6.Uygulama Programlarında Donanım Maliyeti: Bir VT sisteminin avantajlarından biri uygulama geliştirmedeki kolaylıktı, ancak bunu sağlayan ürünler çok sayıda ek kütüphanelere ve yordamlara sahip ve büyük olduklarından bu olanağın sağlanması için gerekli olan donanım birimleri, daha alt düzey programlama ürünlerine göre daha fazladır. Örneğin bir hastanedeki son kullanıcıların sayısının 100'ün üzerinde olduğunu varsayarsak, bu maliyetin önemli boyutlara çıktığını göreceğiz Bir ORACLE VT'nına SQL*FORMS ürünüyle erişmek için bir PC'nin bellek gereksinimi en az 16MB'tır(Windows 3 x ortamında). Oysa aynı VT'na daha kısıtlı

imkanlarla ve ek modüller yoluyla 3.kuşak programlama (CLARION veya Visual Basic)dilleri 4MB'lık bir bellek istemektedir. 100 kullanıcı için gerekli olan toplam bellek maliyeti 1200 MB'tır.

2.1.3.Veri Tabanı Yönetim Sistemi(Database Management System)

Erişim haklarına göre büyük çaptaki verilerin bulunduğu, bir VT'nın yaratılmasını, korunmasını, denetimli sorgulama işlemlerini ve yıkılmalara karşı korunmasını sağlayan yazılıma veri tabanı yönetim sistemi (VTYS) denir.[3]

Bir VTYS işlevlerine ve karmaşıklığına göre; kişisel bilgisayarlardan (PC) çok kullanıcılı ana bilgisayarlara(mainframe) kadar çeşitli ortamlarda çalışırlar. Fiyatları birkaç yüz dolar(dBASE IV) ile birkaç yüzbin dolar(IBM'den DB2) arasında değişirler. Tam işlevli bir VTYS'nin (full-FUNCTION DBMS) temel bileşenleri(Şekil_1) şunlardır:

1.VTYS çekirdeği(DBMS Engine): Bu bileşen bir VTYS'nin ana bileşenidir. Temel işlevi VTYS'nin tüm bileşenlerini koordine etmek ve VT'na ilişkin tüm isteklerin yerine getirilmesini sağlamaktır. İşletim sistemiyle birlikte ana bellek, tampon bellek,indeks ve disk yönetimini üstlenir.

2.Arayüz Altsistemi: Kullanıcıların ve uygulama programlarının VT'na erişmesini sağlayan bir geçittir. Taleplerin gerçekleşmesi için gerekli olan işlemlerin yükümlülüklerine göre diğer ara bileşenlere dağıtımını sağlar.

3.Bilgi Kaynağı Sözlük Altsistemi -BKSA(Information Resource Dictionary Subsystem-IRDS): Bir VT'nı oluşturan nesnelerin ve işlemlerin tanımları sözlük denilen bir kaynakta saklıdır. Bir sözlükte şu bilgiler tutulur:

- Kullanıcıların adları.
- Kullanıcıların her türlü erişim yetkileri ve rolleri.
- VT nesnelerinin adları (tablo, arayüz, indeks, sequence, altyordam, paket, tetikleyici, ... v.b.)
- Bütünlük kısıtlayıcılar (integrity constraint) hakkında her türlü bilgi
- Kolonların default bilgileri.
- Bir VT nesnesinin VT'da nekadar yer kapladığı ve toplam nekadar yerin rezerv edildiği bilgisi.

- Gözlem bilgileri (auditing), örneğin bir kullanıcının hangi objeleri sorguladığı veya güncelleştirdiği bilgisi.
- Diğer genel VT bilgileri.

Sözlükler, sadece sistem tarafından VT üzerinde yapılan işlemler doğrultusunda günlenir. BKSA VTYS çekirdeğinin de işlevlerini kullanarak sözlüğün yönetim ve denetimini üstlenen bir VTYS bileşenidir.

4.Performans Yönetcisi: Bu bileşen VTYS'nin performansını artırmak için gerekli olan optimizasyon işlemlerini yerine getirir[2]. Genel olarak iki önemli işlevi vardır:

- Sorgu Optimizasyonu(Query Optimization): SQL sorgularının (veya diğer sorguların) sonuçlandırılmasını hızlandırmak. Örneğin bir tablo özerinde sorgu yapılırken hangi indeksin kullanılacağına karar vermek gerekir, çünkü o tablo üzerinde tanımlı birçok indeks olabilir İçiçe SQL sorgularında hangi sorgunun öncelikle yapılacağının tespit edilmesi önemlidir. Örneğin: Aşağıdaki SQL cümleciğinde resmi olmayan hastaların hizmetleri sorgulanmaktadır. HAREKET tablosunun verilerinin FATURA verilerinden fazla olduğunu düşünürsek, optimizasyon sırasında önce FATURA tablosunun sorgusunun yapılması önemlidir. Aksi taktirde, HAREKET0 tablosunun tüm kayıtları incelenecek sonra FATURA üzerindeki sorguyla altküme operasyonu yapılacaktır. Bu önemli bir zaman kaybı demektir. Bu yüzden önce HFATURA tablosu ardından HAREKET0 tablosu üzerinden sorgulama yapılmalıdır.

```
SELECT hizkod FROM hareket WHERE kabnum not in (SELECT kabnum from fatura)
```

- VTYS'nin yeniden yapılandırılması: Sözlük yoluyla VT üzerinde tutulan istatistikler VT yöneticisine yeniden yapılandırmanın gerekliliği konusunda bilgiler verir. Örneğin verilerin disk üzerinde dağınık değil (fragmantation) de birarada tutulması disk hareketini azaltır ve sistemin hızlı çalışmasına yol açar. Bu yüzden VT'nın yeniden yapılandırılmasına gerek vardır. Bu yapılandırma belirli bir nesne üzerinde indeks yaratmak da olabilir.

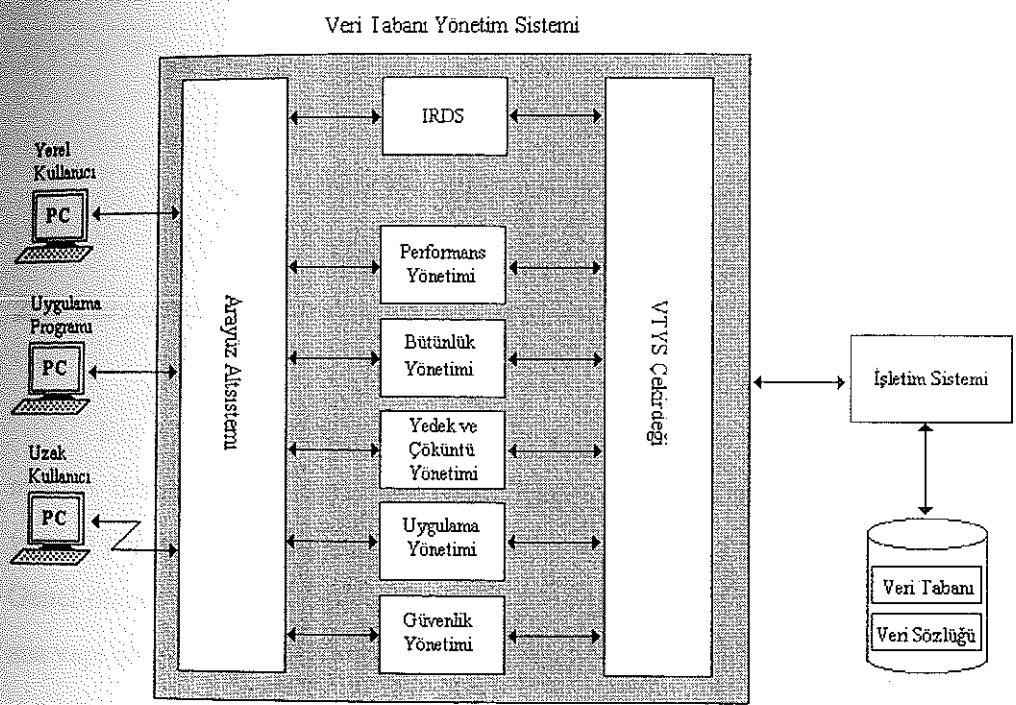
5.Bütünlük Yönetimi: VT ve sözlükteki verilerin bütünlüklerinin sağlanmasını gerçekleştirir. Bütünlük veri alanları üzerinde tanımlanan kısıtların sağlanmasıyla gerçekleşir. Bu bileşen aynı zamanda VT'nın çoklu ortamda tutarlığının sağlanmasını üstlenmektedir. Örneğin bir hastaneye başvuran hastalar ya ayakta ya da yatarak tedavi

görürler. Bunun dışında bir hasta kabül tipi olmadığından bunu bütünlük yönetecisi her kayıt için denetler. Bu kısıtı sağlamak için uygulama programlarında ek program satırlarının yazılmasına gerek yoktur.

6.Yedek ve Çöküntü Yönetimi(Backup And Recovery Management): Bu bileşen VT'nın yedeğinin alınması ve çöküntülerde VT ve transaction'ların kurtarılmasını sağlar.

7.Uygulama Yönetimi(Application Management): Programcıların ve uç kullanıcıların Veri Tabanı uygulama programlarını geliştirmesini sağlayan birimdir.

8.Güvenlik Yönetimi(Security Management): Yukarıda sözü edilen VT'nın güvenlik işlevini yerine getirir.

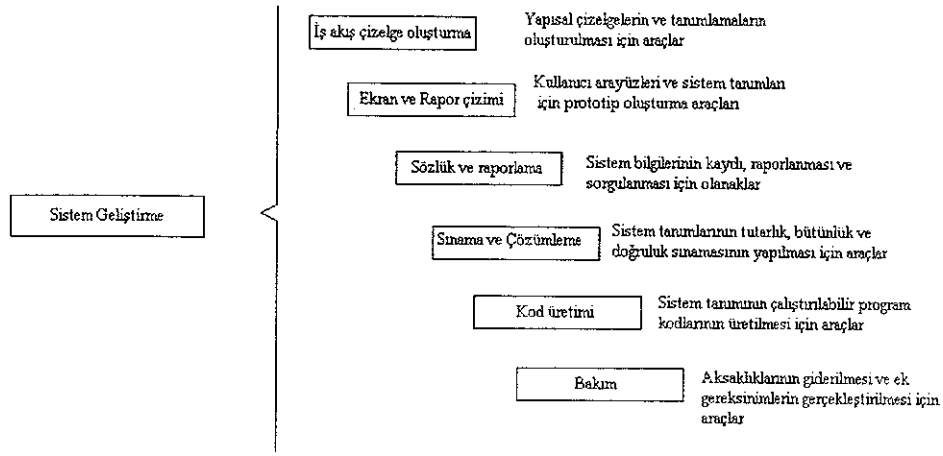


Şekil-1: Bir Veritabanı Yönetim Sisteminin Bileşenleri

2.2. Bilgisayar Destekli Yazılım Mühendisliği Araçları

Günümüzde yazılım geliştirme ile ilgili çalışmalar Bilgisayar Destekli Yazılım Mühendisliği (BDYM) araçlarına (Computer Aided Software Engineering-CASE tools) ulaşmış bulunmaktadır. BDYM araçları bir sistemin uygulama yazılımlarının ve VT'nın geliştirilmesini ve bakımını otomatize edilmesine imkan veren bir sistem geliştirme aracıdır.

İdeal bir BDYM aracı sistem geliştirenlere, yazılım geliştirme sırasında sistematik bir şekilde takip edilmesi gereken bütün aşamalarını (Şekil-2) ve yazılım mühendisliği ile ilgili hertürlü olanağı (kavram, teknik, yöntem, yol, yordam, değerlendirme) içeren bir desteği vermeyi hedeflemektedir.[4]



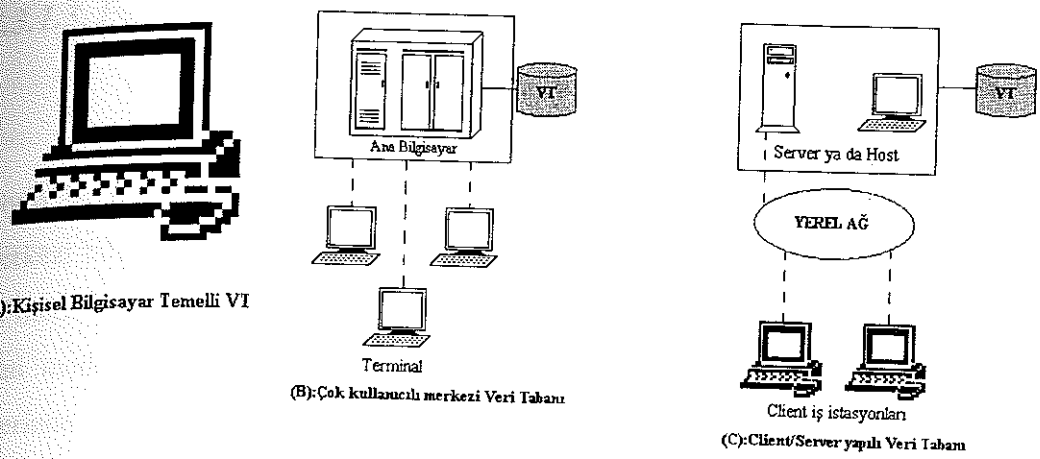
Şekil 2: BDYM Araçlarının gerçekleştirdiği Görevler

2.3. Yerleşim Biçimlerine Göre Veri Tabanı Türleri

Bir VT sisteminin geliştirilmesi sırasında, iş akışının ve verinin boyutuna, maliyet hesabına, kuruluşun büyüklüğü ve dağılımına göre, personel kapasitesine, yazılım parkına ve performans kriterlerine göre farklı VT sistemi modelleri uygulanır. Genel olarak iki ana yapıdan söz edilebilir; merkezi VT ve dağıtık VT, ancak bu iki ana yapı da kendi içinde farklı şekillerde modellenebilir.

2.3.1.Merkezi Veritabanı

Bu modeldeki veri tabanı, merkezi olma niteliği taşıyan bir bilgisayarda bulunur (iş istasyonu, ana bilgisayar, kişisel bilgisayar). Son kullanıcılar taleplerini bir iletişim protokolü yoluyla bu ana bilgisayara yollarlar. Ana bilgisayarda meydana gelecek herhangi bir aksaklık tüm sistemin işlememesine yol açar. Şekil-3'te görüldüğü gibi merkezi VT modelinin üç alt modeli vardır.



(A):Kişisel Bilgisayar Temelli VT

(B):Çok kullanıcılı merkezi Veri Tabanı

(C):Client/Server yapılı Veri Tabanı

Şekil 3: Merkezi Veritabanı

2.3.1.1. Kişisel Bilgisayar Temelli Veritabanı

Şekil-3a'da görüldüğü gibi bu yapıda VT (dBASE, Clarrion, Clipper, Paradox, ...v.b.) kişisel bir bilgisayarda yüklüdür. Tek kullanıcılı bu sistemde VT'nın yaratılması, sorgulanması, günlenmesi ve uygulama yazılımlarının işletilmesi bir tek kişi tarafından yapılır,dolayısıyla VT üzerindeki veriler paylaşımlı değildir. Bu modele sahip veritabanları kısıtlı ve ucuzdurlar.

2.3.1.2. Merkezi Bilgisayara Bağlı Çokkullanıcılı Veritabanı

Veriler ve uygulamalar, birçok kullanıcıya hizmet veren, güçlü ana bilgisayar(mainframe) ve işistasyonları (workstation) tarafından yönetilir(Şekil-3b). Erişimde kullanılan uçlar (terminaller) genel olarak akılsız (işlem yapma ve bilgi saklama yeteneğinden yoksun) üniteler veya bir yerel ağ (Local Area Network-LAN) yoluyla bağlı kişisel bilgisayarlardır (Macintosh, genelde IBM uyumlu PC). Uçlar tarafından talep edilen her türlü veri veya işlem, ana bilgisayar tarafından öncelik sırası ve kuyruktaki sıralarına göre zaman paylaşımlı (timesharing) olarak ele alınır. Zaman paylaşımlı mantığı her işlemin önemine göre belirli zaman aralıklarıyla işletilmesine dayanan bir görev işletim yöntemidir. Bu sistemler büyük çapta verinin ve çok sayıda uygulamanın aynı anda işletilmesine olanak verir.

2.3.1.3. İstemci/Sunucu (Client/Server) Mimarisindeki Veritabanı

Önceleri bir bilişim fantazisi olarak kabul edilen bu yapı artık günümüzde büyük bir hızla yaygınlaşan bir mimaridir. İstemci/Sunucu mimarisi, çok kullanıcılı merkezi bilgisayara bağlı olan VT sisteminin pahallı olması ve işyükünün tamamen bir merkezde yığılmış olmasına alternatif olarak çıkmış ve birçok yönden üstünlükleri olan modern bir modeldir. İstemci/Sunucu modeli, bir bilgisayar ağı (Network) üzerinde bulunan, talepleri yerine getiren ve bunları talep edenlere sunan bir sunucu(Server) ve taleplerini sunucuya ileten istemcilerden(client) oluşur.

Veri transferinin bilgisayarlar arasında yapılmasına olanak sağlayan bilgisayar ağı, bir bina içinde bulunan yerel ağ (Local Area Netwok-LAN) olduğu gibi örneğin internete bağlı bir geniş alan bilgisayar ağı da olabilir(Wide Area Network-WAN).

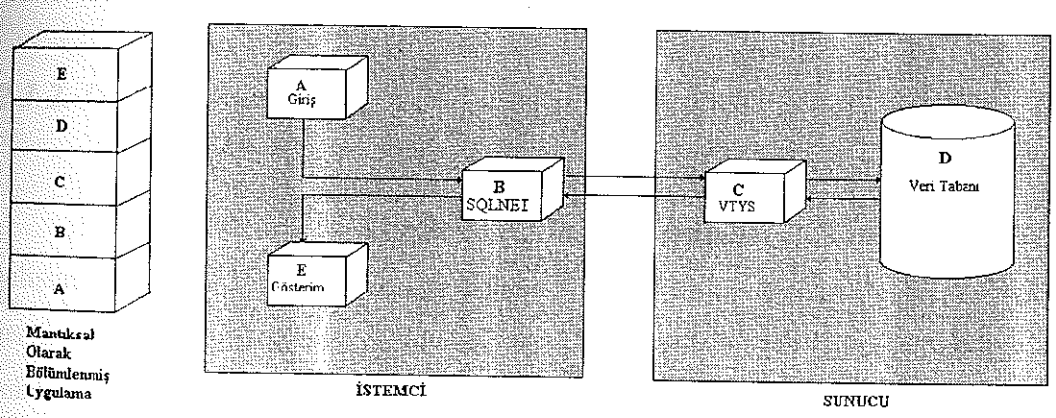
Sunucular işyüküne göre, işlem yapma hızı yüksek RISC(reduced İnstrunction Set Code) temelli merkezi işlem ünitesine(CPU) sahip işistasyonları (Workstation) veya kişisel bilgisayarlar (Macintosh, PC) olabilir. Sunucular ağ protokollerine (TCP/IP,DecNet, ...v.b) açık yazılım ve donanım kaynaklarına (Open System) sahiptir. Diğer bir deyişle bir sunucu, aynı ağ üzerinde bulunan istemciler hangi platformda çalışırlarsa çalışsınlar o istemcilerin taleplerini karşılayacak kaynaklara sahiptir. Veritabanındaki veriler sunucu üzerinde bulunan VTYS tarafından yönetilir ve denetlenir.

İstemciler kendi başına uygulama yazılımlarını işleten bir PC, Macintosh veya işistasyonu olabilir. Bir istemci çalışma esnasında sırasıyla şu işlemleri yerine getirir[6]:

- İstemci üzerindeki kullanıcı çalıştırdığı uygulama programı yoluyla bir işlem talebinde bulunur (girdi işlemleri).
- İstemci ilgili mesajı sunucu için düzenler.
- İstek bilgisayar ağı yoluyla sunucuya iletilir.
- Sunucu mesajı alır, işler ve istemciye sonucu yollar
- Sunucunun yolladığı mesaj bilgisayar ağı yoluyla ilgili istemci(ler) tarafından alınır.
- Veriler istemci üzerindeki uygulama tarafından formatlanır ve işlenir.

Bir istemci/sunucu modelinde tüm işlemler, tek bir merkezden değil de Şekil_4'te görüldüğü gibi dağıtılmış durumdadır. Örneğin bir hastanedeki bir veznedar PC'de çalışan uygulama yazılımıyla hizmetin kodunu girdiğinde (A işlemi) hizmetin adını ve ücretini uzakta bulunan bir VT'dan getirmesi gerekecektir. SQLNET denen uzaktaki VT'na erişimi sağlayan bir yazılım yoluyla (B işlemi), istek SQL cümleciğine formatlanır ve bilgisayar ağından ilgili sunucuya iletilir. Sunucunun VTYS, SQL cümleciğini (C işlemi) algılar ve VT'dan hizmete ilişkin bilgilere erişir (D işlemi) ve yine bilgisayar ağı yoluyla istemciye iletir. İstemci sonucu alır ve ekrandan veznedara ücretini ve adını gösterir (E işlemi). Bu işlem birkaç kez tekrarlandıktan sonra sıra makbuzu kaydetmeye geldiğinde, makbuza ilişkin tüm bilgiler yine aynı yöntemlerle sunucuya iletilir ve VT'na kayıt gerçekleşir. Sonuç olarak veznedar kaydın gerçekten tamamlandığını algıladığında makbuz bilgilerini kendine bağlı yazıcıdan döker, sunucuyu meşgul etmez.

Örnekteki veznedarların sayısını arttırdığımızda bunların hepsinin bağımsız çalıştığını ve aynı anda birden çok işlemin gerçekleştiğini ve dolayısıyla sistemin genel performansının arttığını göreceğiz. Ancak bilgisayar ağının hızının ve yapısının, bu modelin etkinliğini etkileyen önemli parametrelerden biri olduğunu unutmamak gerekir.



Şekil 4: İstemci/Sunucu Modelinde Uygulamaların Bölümlenmesi

İstemci/sunucu mimarisindeki VT'nın avantajlarını şu şekilde sıralayabiliriz:

1.Kişisel bilgisayar teknolojilerindeki hızlı gelişmelerin sisteme minimum efor ve maliyetle adapte edilebilmesi. Bilişim sektöründeki yeniliklerin büyük kısımlarının kısa zamanda kişisel bilgisayarlara yansıdığından bu yeniliklerden faydalanmak

mümkündür. Oysa bir ana bilgisayar temelli sistemde bunun yapılması zordur, çünkü bu tip sistemler yeniliklere son derece az açıktırlar ve maliyetleri yüksektir.

2.Uygulama arayüzleri bağımsız olduğundan, grafik tabanlı kullanıcı arabirimlerinin (Graphic User Interface-GUI) geliştirilebilmesi mümkündür. Böylece kullanılması, öğrenilmesi, geliştirilmesi kolay uygulamalar yazılabilir.

3. Aynı anda birçok işlem yapıldığından ve işlemlerin dağıtılmış olmasından dolayı sistemin genel performansı artar.

4. İstemci/Sunucu mimarisi açık sistem olduğundan farklı platforma sahip sistemlerin (ağ protokolu, işletim sistemi, VTYS) entegrasyonu yapılabilir. Bu yapıya heterojen sistem de denir.

5. Sistemin geliştirilmesi ana bilgisayar (mainframe) temelli sisteme göre ucuzdur.

Buna karşılık istemci/sunucu mimarisinin dezavantajları ise şunlardır:

1. Sistemde farklı platformların bulunması denetimi zorlaştırmaktadır.

2. Sistemde tek tip yazılım, donanım ve uygulama olmadığında ve bunların çok çeşitli olması dolayısıyla kurulması (installation) daha fazla efor gerektirmektedir.

3. Veri ve işlem paylaşımı bilgisayar ağı üzerinden yapıldığından sistemin performansı ağ yapısından etkilenmektedir.

2.3.2. Dağıtık Veritabanı

Dağıtık sistemler (Distributed Systems), sadece bilgisayar teknolojisinin gelişimiyle değil aynı zamanda daha fazla kullanıcıların giderek gelişen gereksinimlerinin doğal sonucu olarak ortaya çıkmışlardır. Bu süreçteki etkenleri şu şekilde sıralayabiliriz:

1. Merkezi sistemlerin yatırım ve büyüme (Donanım, yazılım, iletişim, uzman personel,işletim,...) maliyetinin yüksek olması.

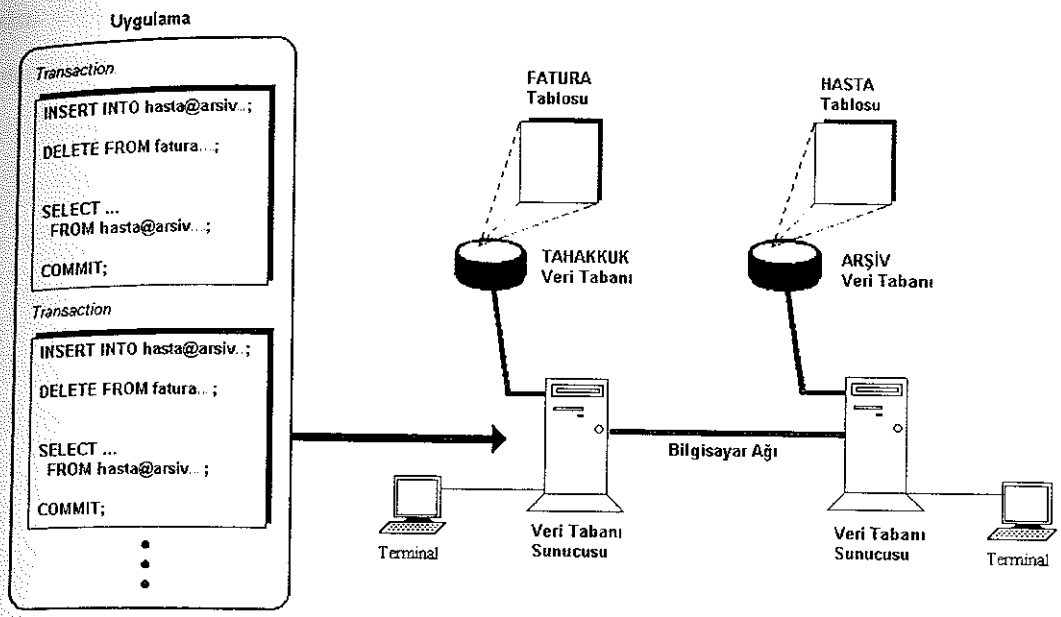
2. Bilgi işlem gereksinimlerinin örgütlerin kendi gelişimlerine paralel olarak artma eğilimi göstermesi ve merkezi sistemlerin başlangıç yatırım maliyetlerinin yüksek olması nedeniyle her işi gören merkezler yerine amaca özel hizmet birimlerini destekleyen sistemlerin daha verimli hale getirilmesi.

3. Merkezi sistemlerin örgütün tüm bilgi ihtiyaçlarını istenilen sürede ve kalitede karşılayamaması, buna karşılık bu tür yazılımların "küçük sistemlerde" kullanımı kolay esnek ve ucuz bir şekilde hazır program arternatiflerinin fazlalığı.

4. Örgütün alt birimleriyle üst birimlerinin bilgi gereksinimleri ile zamanlamanın aynı olmaması sayılabilir, bu da sistemin bakım maliyetinin artması anlamına gelmektedir.

Dağıtık sistemler, ilk ortaya çıktığı dönemlerde genellikle birbirinden bağımsız çalışan sistemlerden oluşurdu. Bu yönetim açısından bir sorun teşkil ederdi, çünkü bağımsız birimlerin verilerinin toplanması sonucunda insan faktöründen kaynaklanan hataların çokluğu denetim, yürütme ve karar destek açısından yanlış sonuçların ortaya çıkmasına, yol ve zaman kaybına yol açabilir, ayrıca çağımızda bilişim sistemlerinin en önemli işlemlerinden biri, piyasa rekabet gücünün arttırılması ve işletim maliyetlerinin düşürülmesi olduğu kadar yönetim etkinliğinin arttırılması da sayılmaktadır. Dağıtık Sistemleri Merkezi Sistemlere göre farklı kılan olgu onların bağımsızlığı değil, tam tersine, yerel ve aynı zamanda gerçek zamanlı olarak ilişkide bulundukları Global Veri Tabanına (GVT) da sahip olmalarıdır. Bu nedenle dağıtık olan sistemlerin entegrasyonuna ihtiyaç vardır. Günümüzde gelişen Açık Sistem (Open System) teknolojisi bizi bu sorunun çözümüne götürmektedir. Dağıtık sistemlerdeki standartlaşmış farklı iletişim protokolleri (TCP/IP,Novell,DecNet,...) ek yazılımlarla fiziksel olarak farklı olmalarına rağmen mantıksal olarak bütünlükleri sağlanmaktadır. [5]

Dağıtık veritabanı modeliyle tasarlanan bir sistemde veritabanları fiziksel olarak farklı bölgelerde bulunmalarına rağmen mantıksal olarak bir bütün gibi görülürler ve bir VTYS'nin üstlendiği tüm görev ve özellikleri taşırlar. Şekil-5'te örnek bir dağıtık veritabanı modeli görülmektedir.Bu modelde iki işistasyonunda dağıtılmış iki farklı veritabanı vardır. Veritabanlarından birtanesi hasta arşiv bilgilerini (kimlik bilgileri, teşhisler, tedaviler, özgeçmiş,...v.b.) diğeri ise hastaların fatura bilgilerini (hastanın her gelişi için uygulanan hizmetler, fatura ödeme bilgileri,...v.b.) tutmaktadır. Hastane içindeki herhangi bir kullanıcı, uygulama programlarında her iki veritabanını aynı anda kullanabilmektedir.



Şekil 5: Örnek Bir Dağıtık Veritabanı Modeli

Dağıtık veritabanlarının iki alt modeli vardır; homojen dağıtık veritabanı, heterojen dağıtık veritabanı.

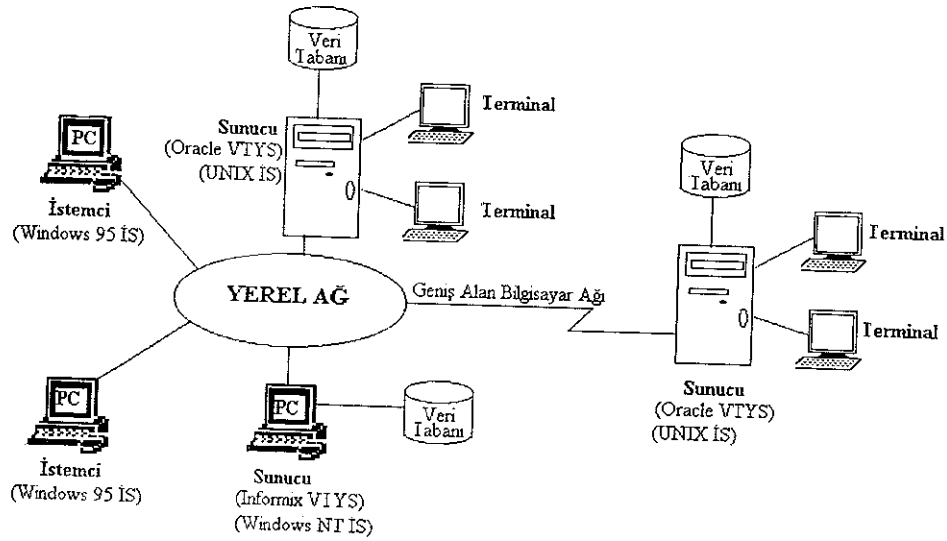
2.3.2.1. Homojen Dağıtık Veritabanı

Eğer bir sistemde dağıtılmış veri tabanlarının hepsinin teknolojileri(işletim sistemi,VTYS) aynı veya uyumlu ise o veritabanı sistemi Homojen Dağıtılmış Veritabanı Sistemidir. Şekil-5'teki modelde iki ayrı işistasyonun işletim sistemleri (UNIX) ve VTYS (ORACLE) aynı olsa o sistemin homojen bir sistem olduğu söylenir.

2.3.2.2. Heterojen Dağıtık Veritabanı

Heterojen dağıtık veritabanlarında her veritabanı sisteminin aynı olması gerekmemektedir. İstemci/sunucu mimarisine ve açık sistemlere paralel olarak ortaya çıkmış olan bu mimari gelişmelere açık, ucuz ve hızlı bir sistem olduğundan gittikçe daha çok kuruluş tarafından uygulanmaktadır. Şekil-6'da örnek bir heterojen dağıtık veritabanı mimarisi görülmektedir. Sistemde birbirinden farklı işletim sistemi ve VTYS' ne sahip istemci ve sunucular bulunmaktadır. Örnekte ayrıca uzak bölgede yer

alan diğer bir veritabanı görülmektedir. Dağıtık olarak değişik bölgelere dağılmış olarak bulunan veritabanları birbirleriyle mantıksal bir bütün oluşturmaktadır.



Şekil 6: Örnek Bir Heterojen Dağıtık Veritabanı Modeli

2.3.2.3. Dağıtık Veritabanı Mimarisinin Avantajları Ve Dezavantajları

Dağıtık VT'larının merkezi veri tabanlarına göre birçok avantajları vardır:[1]

1.Yüksek Güvenirlik. Merkezi bir veritabanında meydana gelecek herhangi bir aksaklık veya arıza tüm sistemin işlemez duruma gelmesine yol açacaktır. Oysa dağıtık veritabanlarında sadece aksaklığın meydana geldiği VT ve buna bağlı uygulamalar çalışmaz, diğer VT ve ilişkin uygulamalar işlemlerini sürdürürler.

2.Yerel Denetim. Dağıtılmış durumdaki VT'larının herbiri fiziksel olarak ayrı olduğundan her birim "kendi" verilerini işletir, denetler ve yönetir. Bu da her VT'nın bütünlüğünü artırır ve yönetimini kolaylaştırır. Gerekli durumlarda diğer birimlerin VT'larındaki verilere yetkileri ölçüsünde erişebilirler.

3.Modüler Büyüme. VT'larının ayrı olmasından dolayı oluşacak yeni gereksinimler ilgili veritabanı ve ilgili uygulama yazılımlarını etkiler bunun dışındaki VT'ları üzerinde değişiklik yapma gereksinimi yoktur. Bu da tüm sistemin bakımının daha kolay yapılması demektir.

4.Düşük İletişim Maliyeti. VT'ları ilişkili oldukları birimlere yakın olduklarından bilgisayar ağındaki trafiği merkezi sisteme göre azaltırlar.

5.Hızlı İşlem: Aynı anda birçok istemci işlem yaptığından ve buna bağlı olarak aynı anda birçok sunucu (VTYS) istemlere karşılık verdiğinden belli bir sürede birden çok uygulama işletilir. Oysa merkezi sistemde işlemler zaman paylaşımlı olduğundan belli bir sürede bir işlem tamamlanır.

Dağıtık veritabanları iyi bir önhazırlık ve çözümleme gerektirmektedir, aksi takdirde yukarıda sözü edilen avantajlar dezavantajlara dönüşebilir. Bu dezavantajları şu şekilde sıralamak mümkündür:

1. Yavaş işlem: VT'ları ve buna bağlı olarak işlemler sağlıklı dağıtılamazsa ve ağ trafiği bu yükü taşıyacak modele ve hıza sahip değilse işlemler merkezi VT sistemine göre daha yavaş tamamlanırlar.

2.Az güvenirlik: Bilgisayar ağındaki, işlemlerdeki,VTYS'lerindeki ve bilgisayarlardaki iştrafiğinin iyi çözümlenmemesi durumunda hata artar ve buna bağlı olarak sistemin güvenilirliği azalır.

3.Yazılım maliyeti ve karmaşıklık: Dağıtık veritabanı mimarisi ve buna ilişkin uygulama programları daha karmaşıktır. Ayrıca işlemlere göre VT'ları farklı bölgelere dağıtıldığından dolayı iyi çözümlenmemiş sistemde karmaşıklığa yol açar ve denetimi zorlaştırır.

2.4. Hastane Bilişim Sistemi

Bir Hastane Bilişim Sisteminin(HBS) amacı hastanedeki uzmanların (idari,akademik), işlerini daha etkin bir şekilde yürütmeleri için gerekli olan klinik, finans ve yönetim bilgilerini yönetmektir.[8]

Bir HBS, hastane içinde hasta takip işlemleri sırasında üretilen, tıbbi (akademik, klinik) ve idari bilgilerin bilgisayara dayalı sistemlerle depolanmasını ve işlenmesini sağlar. Tıbbi bilgiler açısından bir HBS'nin en önemli işlevi hasta takip işlemleri sırasında oluşan her türlü tıbbi bilginin birimler arasındaki akışını ve işlem paylaşımı için iletişim ortamını hazırlamaktır. Eski bilgilerin de tutulmuş olması, teşhis ve tedavi işlemleri sırasında karar destek hizmeti sunulmasına olanak sağlar. İdari bilgiler açısından bir HBS, hastanenin kısa ve uzun dönemli planlarının gerçekleşmesi için gerekli olan karar alımlarına destek olacak bilgilerin tutulması ve

işlenmesi sağlar. Bunun dışında günlük fatura, tahsilat, ödeme ve malzeme alımlarının daha planlı, hızlı ve etkin yapılması için gerekli bilgileri sunar.

Bir hastanenin bilgi gereksinimleri üç ana grupta toplanabilir:

1.İşletimsel Gereksinimler: Bir hastane içindeki her birimde çalışanların (eczane, poliklinik, muhasebe, başhekimlik, hemşirelik,...v.b.) günlük yaptıkları işlemler için belirli bilgilere olan ihtiyaçları bir HBS'nin işletimsel gereksinimlerini oluşturur. Bu gereksinimler sistem çözümleme aşamasında yapılır ve hastanenin genel iş akışı ve bilgi gereksinimlerini ortaya koyar. Bu bir HBS'nin çekirdek kısmını oluşturur.

2.Planlama Gereksinimleri: Hastane çalışanları, hasta takip ve hastane yönetimi için kısa ve uzun dönemli planlar için belirli bilgilere gereksinim duyarlar. Bu bilgiler hastanenin daha planlı ve etkin şekilde işlemesini sağlar ve yöneticilerin karar alımlarına destek olacak bilgilerden oluşur. Planlama için gerekli olan bilgiler hastanenin her biriminde üretilen bigilere dayalı olduğundan bunların yöneticilere bir filitreden geçilerek (çözümlenerek, özetlenerek ve raporlanarak) sunulması gerekir.

3.Belgeleme Gereksinimleri: Hastane içinde üretilen her türlü kaydın, gelecekte kullanılması veya analiz edilmesi amacıyla korunma ihtiyacı belgeleme gereksinimini doğurmaktadır. Böylece bir hastanın tıbbi olarak yapılan müdahaleleri her zaman sorgulanabilir ve uygun tadavinin uygulanıp uygulanmadığı araştırılabilir. İlgili birimler tarafından üretilen hizmetlerin belirli formatta faturalanması bir belge niteliği taşıdığından bir belgeleme gereksinimi doğurur ve gelecekte kullanılması amacıyla arşivlenmesi gerekmektedir.

İdeal bir HBS'nin işlevleri altı temel grupta toplanmaktadır;Çekirdek uygulamalar, finansal işlevler, iletişim ve bilgisayar ağı, birimlerin yönetimi, tıbbi belgeleme ve tıbbi destek.

2.4.1. Çekirdek Sistem

Bir hastanenin çekirdek sistemi hasta kabul, hasta çıkış ve hastanın sevk işlemlerinden oluşur. Hasta kabul işlemleri sırasında hastanın kimlik bilgileri ve ücretlendirme tipi kaydedilir ve ilgili bölüme ve yatağa sevki gerçekleşir. Hastanın hastane içindeki aldığı hizmetler kaydedilir ve yönetilir. Çıkış işlemleri sırasında hastanın aldığı tüm hizmetler sevk biçimine göre düzenlenir, dökülür ve oda bakım

işlemini (housekeeping) başlatır. Hastanın bir yataktan veya bir bölümden diğer yatak veya bölüme sevki gerçekleştirir.

2.4.2. Finans Sistemi

Hasta bakım ve malzeme alım işlemleri sırasında oluşan mali bilgilerin yönetildiği bir sistemdir. Bu sistemin temel işlevi faturaların ilgili kurum veya kişilere dağıtımı, icmal yönetimi, muhasebe, döner sermaye takibi, tahsilat yönetimi ve hastane bütçesinin takibidir. Bu sistem ödeme ve alacakların takibi sonucunda yönetimin malzeme alımları ve yeni yatırımlar için karar alımlarını kolaylaştırır.

2.4.3. İletişim Ve Bilgisayar Ağı Sistemi

Bir HBS'nin bileşenleri arasındaki iletişime olanak sağlar. Kuşkusuz bir hastane bir bütün olduğundan bunu oluşturan birimler arasında veri iletişimi ve işlem paylaşımı sözkonusudur. Bu iletişimin sağlanması hastane içindeki bir bilgisayar ağının oluşturulması ve veri trafiğini kaldıracak düzeyde bir modele sahip olması gerekmektedir. Örneğin doktor tarafından merkezi laboratuarlardan istenen tetkikler laboratuarlar tarafından izlenebilir ve tetkikleri ilgili aygıtlardan numunelerle gerçekleştirilir.

2.4.4. Birim Yönetim Sistemi

Bu yönetim sistemi hastane içindeki her birimin kendine özgü olup tüm hastane bilişim sistemini ilgilendirmeyen özel işlemleri ve ilişkin verileri yöneten bir sistemdir. Örneğin dializ hastalarının kayıtlarının işlenmesi ve tutulması tüm hastanenin iş akışı ile direkt ilişkide olmayan bir işlemdir dolayısıyla birime özgü bir uygulama olmaktadır. Bu sistem global veritabanıyla ilişkili olmadığından dağıtık veritabanı modeline uygun bir sistemi gerektirmektedir. Ancak yukarıda sözü edilen iletişim ve bilgisayar ağı sisteminin tüm hastanede uygulanmasından ötürü birimlerin kendine has verileri gerektiğinde, özellikle eğitim amaçlı olarak hastanenin her terminalinden erişilebilir olmaktadır.

2.4.5. Tıbbi Belgeleme Sistemi

Bu sistem, hastane içinde hasta bakımı sırasında üreyen her tıbbı (akademik) bilginin standartlaştırılması, kaydedilmesi, organize edilmesi ve sunumuna ilişkin işlevleri yerine getirmektedir. Tıbbi bilgiler, hasta bakımı sırasından yapılan her türlü istemlerin, istemlere ilişkin sonuçların, teşhislerin ve tedavilerin standardize edilerek tutulmaktadırlar ve raporlanmaktadırlar. Böylece hasta bakımı sırasında yapılan her türlü müdahaleler arşivlenmekte, eğitim amaçlı kullanılabilmekte ve yönetimin ileriye dönük kararları sağlıklı almasını sağlamaktadır

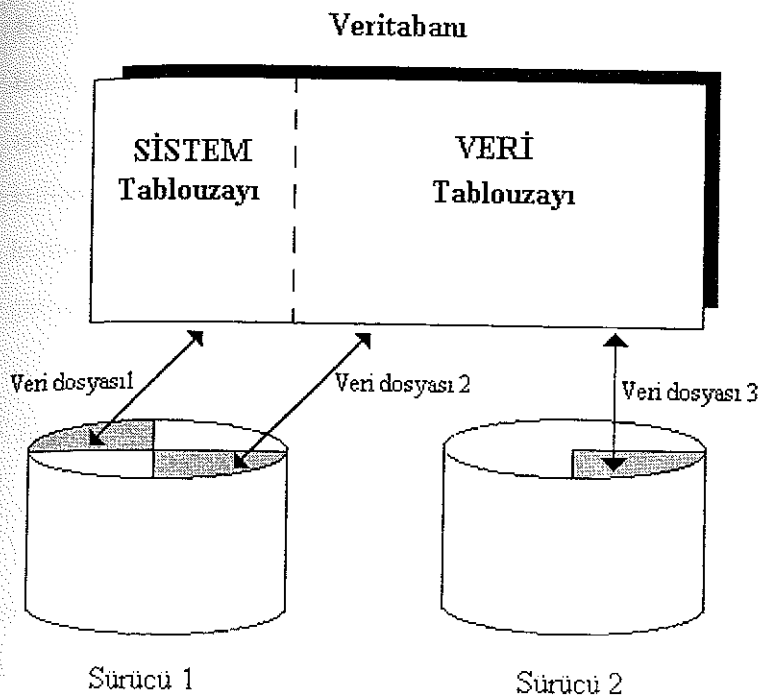
Tıbbi Belgeleme Sistemi aynı zamanda hastane içindeki aktiviteleri (enfeksiyon denetimi, kalite kontrol, vb.) yönlendirmekte ve çalışanların performanslarının izlenmesine olanak sağlamaktadır. Bu bilgiler ışığında yöneticiler hastanenin işleyişi hakkında daha net verilere sahip olmakta ve dolayısıyla karar alımları kolaylaşmaktadır.

2.4.6. Tıbbi Destek Sistemi

Akademik personelin verileri yorumlamasını ve hastaya uygulanacak mudahalelerde karar alımını kolaylaştırmayı amaçlayan bir destek sistemdir. HBS'nin bileşenleri iyi organize edildiğinde tıbbi destek sistemi hastaların izlenmesini ve tedaviler sırasındaki kararlara yön vermeyi amaçlayan bir sistemdir.

2.5. ORACLE VTYS

ORACLE, VTYS ve ek paketleriyle birlikte daha önce sözü edilen veri tabanı özelliklerini içinde barından bir VTYS'dir. Bir ORACLE veri tabanı mantıksal ve fiziksel olmak üzere Şekil 7'de görüldüğü gibi iki temel bileşenden oluşur.[3]



Şekil 7: ORACLE VT, Tablouzayı ve veri dosyası

1. *Fiziksel Veritabanı Yapısı*. ORACLE veri tabanlarının fiziksel yapısı veri tabanının verilerini tutan işletim sistemi dosyalarından oluşur. Bu dosyalar depolama birimlerinde tutulur ve ORACLE tarafından talep edilip işletim sistemi tarafından yönetilen dosyalardır.

2 *Mantıksal veritabanı Yapısı*. Her veri en küçük veri depolama birimi olan veri bloklarına yerleşir. Örneğin bu blok değeri DG_UX işletim sistemi için 2048 Byte'tır. Veriler bu bloklar üzerinden veri dosyalarına yerleşirler. Veri dosyasında arka arkaya gelen veri blokları extent adı verilen belli bir nesne için kullanılan veri alanlarını oluştururlar. Veriler her defasında extentleri doldurduğunda yerine yeni extentleri kendileri için ayırırlar. Bu alan veri dosyalarındaki boş alanlara karşılık gelmektedir. Extenler bir araya gelerek belli bir veritabanı nesnesine karşılık gelen (Örneğin bir tablonun verileri) segmentleri oluştururlar. Segmentler ise bir araya gelerek birden çok veritabanı nesnesini yöneten tablouzaylarını oluştururlar. Bir ORACLE veritabanı tablouzayı denen mantıksal parçalara ayrılmıştır. Bu parçalar veritabanını oluşturan tüm nesneleri içinde maktıksal bir bütün çerçevesinde VTYS tarafından yönetilir. Şekil-7'de de görüldüğü gibi bir veri tabanı bir veya birçok tablouzayından oluşur. Tablouzayında bulunan her nesne sürücüler

üzerinde işletim sistemi dosyalarında fiziksel olarak tutulurlar ve bu dosyaların boyutlarının toplamı o tablouzayının boyutunu oluşturur. Sistem tablouzayında bulunan nesneler ORACLE veritabanının sözlük bilgilerini tutar. Tüm bu birimler işletim sistemi dosyasındaki alanlara karşılık geldiğinden disk alanlarının kullanımı ile yakından ilişkili parametrelerle yönetilirler. Dolayısıyla bu nesnelerin kontrollü kullanılması ve veritabanı nesnelerinin davranışlarının iyi çözümlenmesi gerekmektedir.

2.5.1. ORACLE Veritabanı Nesneleri

Bir ORACLE veritabanı işlevlerine göre mantıksal nesnelerden oluşur. Bu nesnelerin herbiri farklı işlevlerine göre veri tabanı yöneticisi ve uygulama programcıları tarafından yaratılırlar. ORACLE veritabanlarındaki nesneler şunlardır:

1. Tablolar(Tables): Bir veritabanının temel veri depolama birimi olan tablolar tüm kullanıcıların yetkilere göre erişeceği verileri tutar. Tablolardaki veriler kolon ve satırlardan oluşur. Satırlar o tabloya ilişkin kayıtları, kolonlar ise kayıtların özelliklerini tutar.

2. Arayüzler(Views): Arayüzler birden çok tablonun ilişkilerine göre birleştirilmesinden (join) oluşan kayıtlı sorgulardır. Bu nesneler veritabanını kullanan her birimin erişeceği verileri oluşturur. Bu nesneler içinde veri tutmazlar sadece hangi tablolardan hangi işleve göre veri alınacağını belirten sorguların tanımlarını tutarlar.

3 Sequence'lar: Kullanıcılar çoğu zaman birbirini takip eden sıralı numaralara ihtiyaç duyarlar. Bunların birçok kullanıcı tarafından aynı anda erişildiği düşünülürse her istendiğinde birbirinden farklı sıralı numaralar üretilmesi ve çakışmanın oluşmaması gerekmektedir. Bu nesneler ORACLE VTYS tarafından denetlenmektedir. Örneğin bir hastanedeki fatura numaralarının veya istem formlarındaki numaraların tekliği bu nesneyle kolaylıkla sağlanabilir.

4.Program Parçaları: Uygulama programları tarafından kullanılan her türlü program parçaları (yordamlar, paketler, tetikleyiciler) veri tabanına kaydedilir ve birçok kullanıcının kullanımına sunulur. Bu nesnelerin veri tabanına kaydı yetkisiz kullanımı da denetler. Bu program parçaları ORACLE'ın PL/SQL diye adlandırılan bir programlama dili ile geliştirilir.

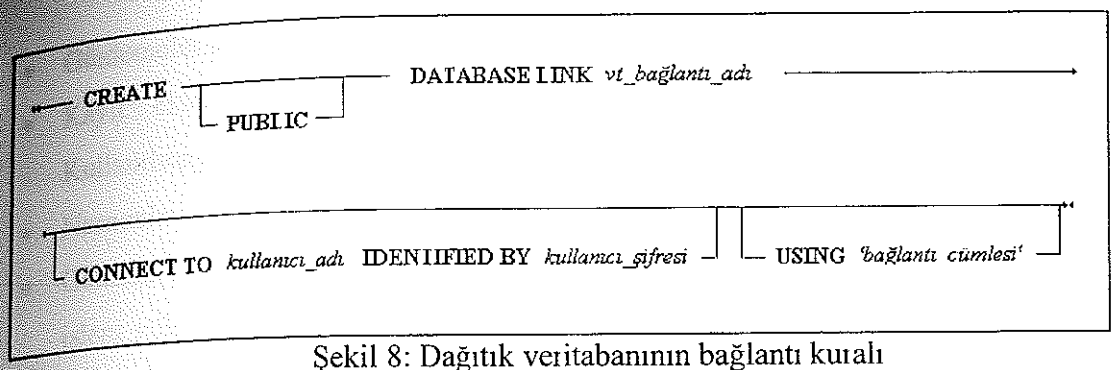
5.Synonym'lar Veritabanındaki nesnelere farklı isimlendirilmesine olanak veren bu nesneler veri tutmazlar sadece nesneye referans gösterilirler. Bu nesnelerin anlamlı isimlendirilmesini sağlar ve veritabanının okunabilirliğini arttırır, gerçek ismi ve yaratıcı kullanıcıyı maskeler, ilgili nesneye her kullanıcının erişimi sağlar, SQL cümleciklerinin veritabanı kullanıcılarına kullanımını kolaylaştırır.

6.Indeks, Cluster, Hash Cluster Indeksler veritabanındaki verilere erişimi hızlandıran nesnelerdir. Tablolardaki veri alan(lar)ı üzerinde uygulama programcısı ve veritabanı yöneticisi tarafından yaratılan ve veri tabanında yer kaplayan nesnelerdir. Bu indeksler uygulama programcıları tarafından belirtilmediği sürece sistem tarafından optimize edilerek kullanılır. Ancak uygulama programcıları SQL cümleciklerinde optmizasyonu ipuçlarıyla (hint) yönlendirebilirler ve erişimi daha da hızlandırabilirler. İndeksler, fiziksel ve mantıksal olarak üzerinde tanımlı oldukları tablolardan bağımsızdırlar. Tablolar üzerinde yapılan ekleme, silme ve günleme işlemleri indekslere de yansır.

Cluster'lar birbirleriyle sıkı ilişkide bulunan iki veya daha çok tablonun diskte yakın bölgelerde yerleştirilmesini sağlayan nesnelerdir. Bunlar disk hareketlerini azalttığı için cluster üzerinde tanımlı tablolara erişimin daha hızlı yapılmasına olanak sağlarlar. Örneğin hastalar için doktorlar tarafından istenen istem formları ile (istem numarası, istemi talep eden doktor, istemi gerçekleştiren birim,istem tarihi,...v.b.) isteme ilişkin gerçekleştirilen hizmetler birbirleriyle sıkı ilişki içinde bulunan iki tablodur. Bunların bir cluster üzerinde tanımlanması hem istemlere hem de hizmetlere hızlı erişimi sağlar.

Hash cluster'lar ise cluster üzerinde tanımlanan indekslerin bir matematiksel formülle (hash işlevi) verilere erişimi sağlayan indekslerdir. Aynı hash anahtarına sahip verilerin birarada tutulduğundan erişim hızlanır.

7. Veritabanı bağlantıları (Database Links) Dağıtık veritabanı sistemiyle tasarlanmış veritabanlarının mantıksal olarak bütünlüklerini sağlamak için veritabanları arasında bağlantılar tanımlanır. Böylece yerel veritabanlarının biraraya gelmesiyle mantıksal olarak genel bir veritabanı oluşturulmuş olur. ORACLE'da bu bağlantılar Şekil-8'deki gibi tanımlanır:



Şekil 8: Dağıtık veritabanının bağlantı kuralı

Örneğin: Tahakkuk servisinde bulunan sunucudaki veritabanında **CREATE PUBLIC DATABASE LINK *db_arsiv*** denildiğinde arşivde bulunan sunucu üzerindeki *db_arsiv* veritabanı, tahakkuk bölümündeki sunucunun tüm kullanıcılarına açık olarak olarak bağlanmış olur, sanki kendi üzerinde yaratılmış bir ORACLE veritabanı gibi. Böylece tahakkuktaki bir kullanıcı arşivde bulunan hastaları listelerken şu SQL cümleciğini yazması yeterli olacaktır:

SELECT * FROM *hos.hasta@db_arsiv*;

2.5.2. ORACLE VTYS Nesnelerinin Yönetimi

Yukarıda sözü edilen nesnelerin veritabanında uygun şekillerde yerleştirilmesi sistemin performansı ile orantılıdır. Veritabanı yöneticisinin sistemi tasarlarken verilerin davranışlarını (kayıtların genişleme oranı, ortalama kayıt boyları, ekleme, silme, sorgulama ağırlıkları, ...vb.) ne kadar iyi çözümlerse verilerin yerleşimi, bakımı ve disk üzerinde yayılması o kadar kontrollü ve sağlıklı işler. Bunun sağlanması ORACLE VTYS nesnelerindeki parametrelere bağlıdır.

2.5.2.1. PCTFREE Parametresi

Verilerin yerleştiği en küçük veri alanı olan veri bloklarının içinde bulunan kayıtların günlemeler sonuncunda genişlemelerinin ilgili bloğa yerleşmesini sağlayan veri alanlarının yüzdelerini gösterir. Örneğin bir UNIX işletim sistemindeki veri bloğu 2048 KB'tır. Bunun %10'luk PCTFREE değeri 204 KB'tır. Yani ilgili nesne için 204 KB günlemeler için ayrılmıştır ve her veri bloğundaki kayıtların genişleyen kısımları bu alana yerleşir[9]. Düşük bir PCFREE değeri ilgili nesne için şu etkileri doğurur:

- Günlemeler sonucunda kayıtların genişleme alanları azaltılır.
- Kayıt eklemelerinde verilerin veri bloklarına tam yerleşmelerini sağlar. Kullanılmayan veri alanı azalır.
- Nesnenin kayıtları daha az bloğa yerleşir.
- Günlemeler için ayrılan alan az gelirse bir kaydın verileri farklı bloklara dağılır ve işlem yükü artar sistem yavaşlar. Dolayısıyla nesnenin yeniden organize edilmesi gerekir bu da ek işlem demektir.

PCTFREE değeri eğer nesne fazla günlenmeyen kayıtlar gerektiriyorsa kullanılır.

Yüksek bir PCTFREE değeri ise şu etkileri yaratır:

- Günlemeler için daha çok veri bloğu alanı rezerv edilir.
- Nesnenin kayıtları daha fazla bloğa yerleşeceğinden daha fazla disk ve bellek alanı harcanmış olur.
- Günlemeler neticesinde bir kayıt farklı bloklara yayılmaz dolayısıyla performans artar.
- Bir kayıt farklı veri bloklarına yayılmadığından tekrar organizasyon gerekmez.

Yüksek PCTFREE değeri eğer kayıtlar sıksın ekleniyorsa ve günleniyorsa kullanılır.

Görüldüğü gibi kayıtların davranışları iyi çözümlenmelidir. Verileri sıksın günlenen bir nesneye düşük PCTFREE değeri verilirse veriler disk üzerinde çok farklı bölgelere dağılır, kilitlenmeler artar ve erişim hızı iyice yavaşlar. Tersi durumda ise disk alanı israfı söz konusu olur ve kullanılan veri alanı (disk doluluk oranı) düşer.

2.5.2.2.PCTUSED Parametresi

Bir veri bloğundaki boş alan PCTFREE değerine ulaştığında yeni kayıtların eklenmesi nesne için tanımlı PCTUSED değerinin altına inmeden gerçekleşmez. Böylece ORACLE veri bloğunun en az PCTUSED değeri kadar dolmasını garantiler. PCTUSED değeri default olarak %40'tır ve alabileceği değerler 0 ile 90 arasındadır.[9]

Düşük bir PCTUSED değerinin nesne üzerindeki etkileri şu şekildedir:

- Bir veri bloğunun doluluk oranı yüksek bir PCTUSED değerine göre daha azdır.
- Günleme ve silme işlemlerinde işlem maliyetini azaltır.

- Veri tabanının kullanılmayan veri alanları artar.

Yüksek bir PCTUSED değerinin nesne üzerindeki etkileri ise şunlardır:

- Bir veri bloğunun doluluk oranı düşük bir PCTUSED değerine göre daha azdır.
- Alan etkinliğini arttırır.
- Ekleme ve günleme işlemlerinde işlem maliyetini arttırır.

2.5.2.3. Segment Depolama Parametreleri

Her veri tabanı default depolama parametresine sahiptir ve işletim sistemi kuralları çerçevesinde değerler taşırlar. Ancak nesnelerin daha sağlıklı gelişmeleri ve veri tabanının bakım maliyetinin azaltılması için depolama parametreleri ayrıca hesaplanmalıdır. ORACLE VTYS'nin parametreleri şunlardır:

1.INITIAL: Segment için yaratılan ilk extentin kaplayacağı veri alanı boyunu belirler. Eğer nesne gelişmeyen ya da az gelişen bir yapıya sahipse bu nesnenin verileri sabit sayıda ve yakın bloklara yerleşir bu da erişim hızını arttırır

2.NEXT: Segmentler, INITIAL değerini geçtiklerinde NEXT parametresiyle tanımlı olan extent alanlarına yerleşir.

3.MAXEXTENTS: Segment için en fazla ayrılacak extent sayısını belirler. Segmentin extent sayısı bu değeri geçtiğinde yeni extent yaratılmaz ve hata oluşur.

4.MINEXTENT: Segment ilk yaratıldığında ayrılacak olan extent sayısıdır.

5.PCTINCREASE: Segment NEXT parametresiyle tanımlı alanları doldurduktan sonraki alanların miktarının bir öncekine göre arttırma yüzdesidir.

6.INITRANS: Segment üzerinde yapılacak her transaction için önceden ayrılacak olan alan sayısı.

7.MAXTRANS: Nesne üzerinde aynı anda aynı veri bloğuna yapılan transaction'lar için ORACLE tarafından veri bloklarında bir alan rezerv edilir. Eğer bu alan INITRANS değerini geçerse MAXTRANS değerine kadar yeni alanlar ayrılır.

2.6. CLARION Uygulama Geliştirme Aracı

Clarion, programların kolay okunması, geliştirilmesi ve gereksiz yazım kurallarından arındırılması düşüncesiyle ortaya çıkmış bir yazılım geliştirme aracıdır. Clarion'u, Visual Basic, Delphi ve Power Builder gibi rakiplerinden ayıran en önemli özellik, bunların veritabanı geliştirme araçlarıyla genişletilmiş birer programlama dili olmalarına karşılık, Clarion'un programlama araçlarıyla genişletilmiş bir veritabanı geliştirme sistemi olmasıdır. [1]

Clarion, Windows 3.x, Windows95, Windows NT ve Dos ortamlarında alt düzey ve veritabanı uygulamaları geliştirmek için Topspeed tarafından geliştirilmiş bir uygulama geliştirme aracıdır. 1996 sonlarında piyasaya sürülen Clarion 2.0 sürümü, Windows ortamlarındaki RAD (Rapid Application Development), ActiveX, OLE(Object Linking and Embedding), DLL(Dynamic Link Library), DDE(Dynamic Data Exchange), ODBC(Open Database Connectivity), SQL ve Client/Server desteği vermektedir. Bir Clarion programının genel yapısı şu şekildedir:

```
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 tanımları
          CODE                                      ! Program satırları
          program_cümlecikleri                      ! İşletilebilir program komutları
          [RETURN]
etiket  ROUTINE                                     ! Yerel altprogram
        program_kodları
        END
etiket  PROCEDURE(parametre_listesi)                ! Yordam tanımı
        Yerel_veriler
        CODE
        Program_cümlecikleri
        [RETURN]
etiket  ROUTINE                                     ! Yerel altprogram
        program_kodları
        END
etiket  FUNCTION(parametre_listesi)                 ! İşlev tanımı
        Yerel_veriler
        CODE
        Program_cümlecikleri
        [RETURN(değer)]
etiket  ROUTINE                                     ! Yerel altprogram
        program_kodları
        END
```

[1] **BONNER, Paul,**"Good Advice:When in Doubt, Get a Trout", http://www zdnet com/wsources/content/960910/regrev16.html,1996

C programlama dilinin hızlılığını ve etkinliğini uygulamalarında kullanıcılara sunan bu programlama aracı minimum donanım kaynaklarında hızlı çalışması ve standartlara uyumlu olması nedeniyle, bu yazılım Akdeniz Üniversitesi Hastanesi için geliştirilmekte olan HBS'nde kullanıcı uygulamalarının gerçekleştirilmesinde kullanılacaktır. Clarion'un ORACLE Connect ürünü ile, yerel ağa bağlı tüm ORACLE veritabanı nesneleri, ve yordamları minimum düzeyde kaynak ile Clarion uygulamalarında kullanılabilmektedir.

3.METOD

3.1. Akdeniz Üniversitesi Hastanesinin Genel Yapısı

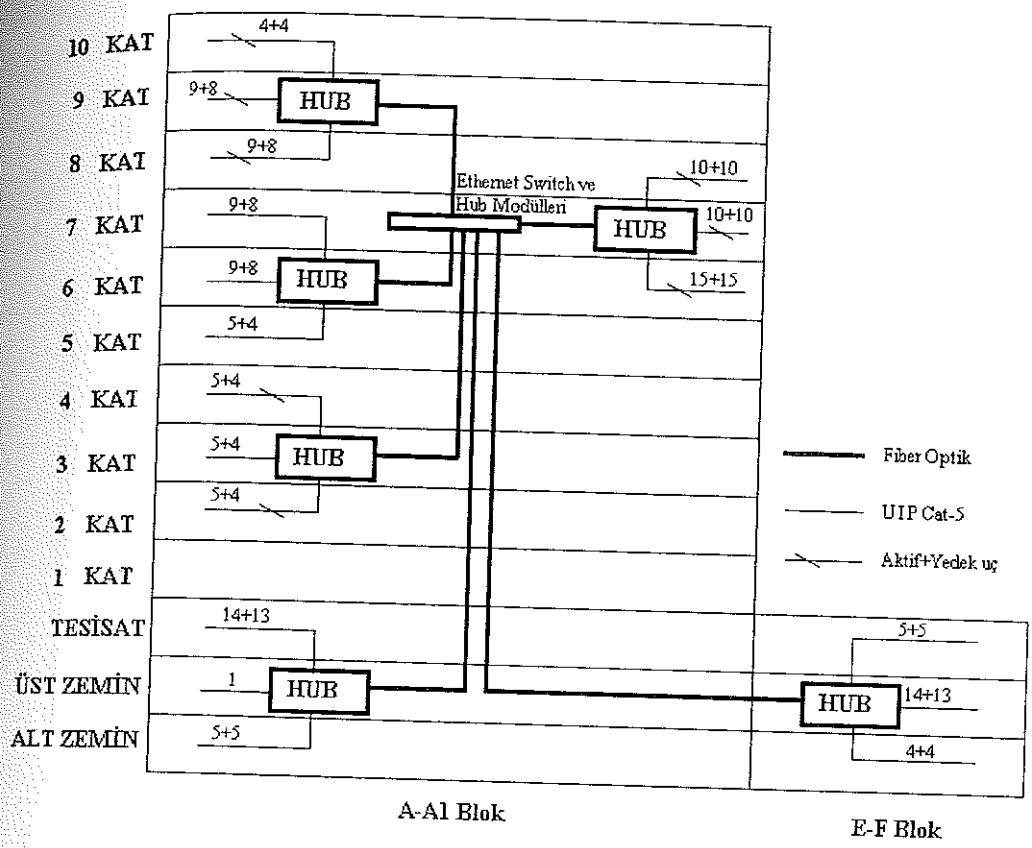
17 Klinik, 18 Laboratuar, 27 Poliklinik, 27 adet idari ve teknik birimin bulunduğu Akdeniz Üniversitesi Hastanesi, temel olarak iki bloktan oluşmaktadır. 11 katlı birinci blokta, araştırma merkezleri, muhasebe, tahakkuk, tahsilat, idari birimler, servisler, eczane ve merkezi laboratuar bulunmaktadır ve toplam 300 yatan hastaya hizmet vermektedir. İki katlı ikinci blok ise poliklinikler, arşiv, acil servis, yoğun bakım üniteleri ve ameliyathanelerden oluşmaktadır. Hastane birimlerine göre üç ana grupta ele alınabilir:

- *Klinik ve Poliklinikler*: Genel Cerrahi, Kulak-Burun-Boğaz, Göğüs Cerrahisi, Gögüs Hastalıkları, Kalp Damar Cerrahisi, Çocuk Cerrahisi, Nöroşirurji, Kadın Hastalıkları ve Doğum, Ortopedi, Nöroloji, Üroloji, Göz Hastalıları, Çocuk Sağlığı ve Hastalıkları, Çocuk Psikyatrisi, Enfeksiyon ve Bakteriyoloji Hastalıkları, Psikyatri, Dermatoloji, Nefroloji, Endokrinoloji ve diabet, Gastraenteroloji, Transplantasyon, Periton Daliz(CAPD), Hipertansiyon, İç Hastalıklar, Kardiyoloji, Enfeksiyon Hastalıkları, Fiziksel Tıp ve Rehabilitasyon ve Acil Servis.
- *Laboratuarlar*: Biyokimya, Hematoloji, Mikrobiyoloji+Eliza, Nükleer Tıp+RİA, Radyoloji, Anjiografi, Bilgisayarlı Beyin Tomografisi, Koroner Anjiografi, Efor Testi, Treadmil, Odyoloji, Kan Merkezi, Endoskobi, Ümmünoloji, Patoloji, EKG,EMG,EEG ve ENG
- *İdari, Teknik ve Diğer Birimler*: Başhekimlik, Bilgi İşlem Merkezi, APK, Hemodializ, Ameliyathane, Sterilizasyon, Hemşirelik Müdürlüğü, Eczane Mesul Müdürlüğü, Diyet ve Beslenme Bölüm Şefliği, Sosyal Servis, Satınalma Müdürlüğü, Tahakkuk Şefliği, Ayniyat Saymanlığı, Hasta Kabul Şefliği, Personel Şefliği, Senetler Şefliği, Malzeme Planlama Şefliği, Evrak Servisi, Ev İdaresi Şefliği, Tıbbi Dokumantasyon ve Arşiv Şefliği, Ders Araçları Merkezi, İmamlık, Santral, Elektrik Atölyesi, Sıhhi Tesisat Atölyesi, Kalorifer Atölyesi, Araç İşletme Merkezi, Marangozhane, Terzihane, Boyahane ve Kaynak Atölyesi.

Akdeniz Üniversitesi Hastanesinde 1996 yılı sonunda 353400 hasta kayıtlıdır ve günde ortalama 1200 hasta tedavi görmektedir. Her hasta yine ortalama 5 hizmet almaktadır. Yapılan araştırmalara göre bir hasta yılda üç kez hastanede tedavi görmektedir. Bu veriler ışığında veritabanında oluşturulacak verilerin ortalama yıllık disk gereksinimleri hesaplanmaktadır.

3.2.Akdeniz Üniversitesi Hastanesi Bilgisayar Ağının Yapısı

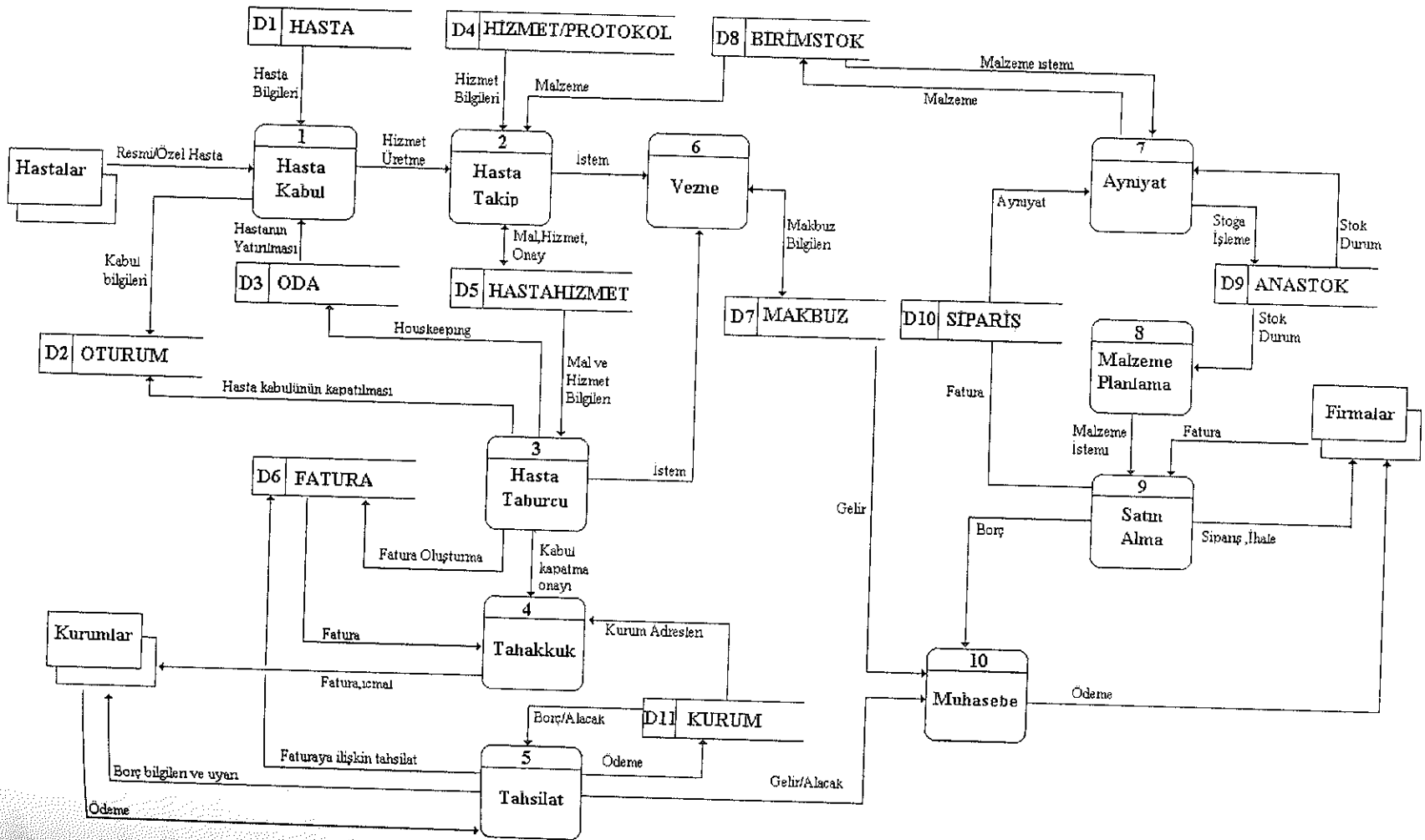
Yapılan olurluk ve sistem çözümleme çalışmalarında Akdeniz Üniversitesinin Bilgi Alt Yapısının Homojen Dağıtık veritabanı ve Client/Server mimarisine uygun olarak tasarlanmasına karar verilmiştir. Bu mimariler hastanedeki tüm uç kullanıcıların her türlü veriye erişebilmesine olanak vermektedir. Ancak daha önce de sözü edildiği gibi, bu mimarilerin etkinliği bilgisayar ağının topoloji ve modeline bağlıdır. Yapılan çalışmalarda ve sonuçlanan ihaleler neticesinde ethernet network yapısının kullanılması uygun görülmüştür. 11 Katlı hastanenin her katı ile yoğun bakım ve ameliyathanelerin bulunduğu bölümler, Fiber Optik kablolarla (62.5/125 mikrometre, 4 damar multimode, graded index) ağ merkezine bağlanmakta ve katlar Hub'lar (10/100BaseT çıkışlı, 24*5 10/100BaseT porta kadar büyüyebilir)ve UTP Category-5 kablolarıyla uçlara (toplam 168 adet) ulaşmaktadır. Ağ iletişim hızı 10M bps olup kısa süre içinde bu hız 100M bps'e ulaştırılacaktır[10]. Ayrıca ortaya çıkacak olan yeni ihtiyaçlar doğrultusunda kablo çekiminin zor olacağı ve veri iletişimine az ihtiyaç duyan yerler için, hem maliyet düşüklüğü hem de kurma kolaylığı açısından kablosuz (Wireless) iletişimin kullanması uygun görülmektedir. Hastane için yapılan bilgisayar ağı omurgası Şekil-9'da görüldüğü gibi gerçekleştirilmiştir.



Şekil 9: Akdeniz Üniversitesi Hastanesi A-A1 ve E-F Bloklarının Bilgisayar Ağı Kablolama Şeması

3.3.Çekirdek Sistem İş Akışı

Hastane içindeki işlem ve bilgi paylaşımının önemi daha önceki bölümlerde belirtilmişti. Yapılan sistem çözümleme çalışmalarında Akdeniz Üniversitesinin iş akışı tamamlanmıştır (Şekil-10). Buna göre veri kaynakları ve yordamlar arasındaki veri ve işlem akışı aşağıdaki bölümlerde anlatılmıştır.



Şekil-10:Akdeniz Üniversitesi Hastanesi Bilişim Sistemi Çekirdek Sistem Akış Şeması

3.3.1.Arşiv

Arşiv, hastane süreçleri sonucunda hastalar ile ilişkili tüm tıbbi ve kimlik kayıtlarının kaydedildiği havuzdur, diğer bir deyişle HBS'nin tıbbi dokumantasyon ihtiyacı karşılanmaktadır. Burada her hasta bir dosya numarası ile karakterize edilir. Bu havuzda istenildiği anda hastanın önceki gelişlerine ilişkin tıbbi kayıtlarına (istemler, sonuçlar, teşhisler, tedaviler, ilaçlar, radyolojik bilgiler, raporlar, laboratuar bilgileri,vb.) erişilmektedir.

3.3.2.Randevu ve Rezervasyon

Randevu ve Rezervasyon merkezleri belli bir tarihte,

- Poliklinik hizmeti,
- Hasta Yatağı,
- Özel öğretim üyesi muayenesi,
- Ameliyat,
- Radyoloji,
- Nükleer tıp,

hizmetlerini almak isteyen hastalara ilişkin hizmetleri yürütür. Randevu ve rezervasyonlar yönetimce belirlenen çalışma ve iş gücü planları doğrultusunda gerçekleştirilmektedir.

3.3.3. Hasta Kabul

Akdeniz Üniversitesi'nde temel olarak resmi, özel ve yarı resmi diyebileceğimiz türden üç tipte hasta tedavi görmektedir. Resmi olan hastalar belli bir kamu kuruluşuna bağlı olan ve kurumlarından aldıkları sevklerle hizmet alan hastalardır. Sözü edilen kamu kuruluşlarıyla hastane arasında her yılın başında bir ücretlendirme protokolü belirlenir ve hastaların hizmetleri bu protokole göre ücretlendirilir. İmzalanan protokollere göre hastaların faturaları ilgili kurumlara yollanmaktadır. Özel hastalar ise bir kuruma tabii olmayıp da kendi bütçelerinden ödemeyi talep eden tipte hastalardır. Bunlar resmi fiyata ve özel tarifeye göre hizmet

alırlar. Yarı resmi tipte hastalar ise tedavilerinin bir kısmını resmi tarifeye (bu kuruma bağlı bir hasta da olabilir) bir kısmını ise özel tarifeye göre ödeme yapmak isteyen hastalardır. Hizmetlerin karşılıkları, hasta kuruma bağlı ise, resmi tarifeye göre olan tedavi hizmetleri kurumlardan, farkları ise hasta tarafından yatırılan makbuzlardan alınmaktadır, hasta kuruma bağlı değilse yatırılan makbuz ve faturalardan alınmaktadır.

Hasta Kabul merkezlerine gelen hastaların öncelikle dosyalarının olup olmadığına bakılır, dosyası olmayan hastalara arşiv tarafından bir dosya açılır. Hastanın kimlik bilgilerinin yanısıra, özel bilgileri (kurumu, kurum sicil numaraları, sevk, protokol,adres,referans,vb.) girilir. Hasta yatan türden hasta ise, isteğine ve randevu/rezervasyon bilgilerine göre, 1.sınıf, 2.sınıf veya suit odalardan biri rezerv edilir, hastanın hastanedeki oturumunu tanımlayan bir kabul numarası verilir ve artık hasta hastaneden hizmet almaya hazırdır. Hastanın transfer (yatak,birim) işlemleri yine bu merkezler tarafından ele alınmaktadır.

3.3.4.Hasta Taburcu Süreci

Hasta kabul bankolarından yürütülen bu işlem, hastanın aktif hastane oturumunu kapatır ve bu oturum esnasında oluşan tüm mal ve hizmet bilgilerini ücretlendirme protokolüne göre gerekli bilgileri tahakkuk birimine, veznelere ve muhasebe tahsilat merkezlerine aktarır ve yatan hasta için housekeeping işlemini başlatır. Son olarak hastaya ait tüm tıbbi kayıtlara göre arşiv dosyasını günleyerek hastanın taburcu sürecini sonlandırır.

3.3.5.Tahakkuk,Tahsilat Süreci ve Vezneler

Tahakkuk işlemi, Hasta taburcu işlemlerinden sonra oturumu kapatılan hastaların hizmetlerinin kurum bilgilerine göre faturalanmasını ve kurumlara postalanmasını, zimmetlerin oluşturulmasını, tahakkuk edilen faturaların aylık, yıllık raporlarının dökülmesini gerçekleştirir.

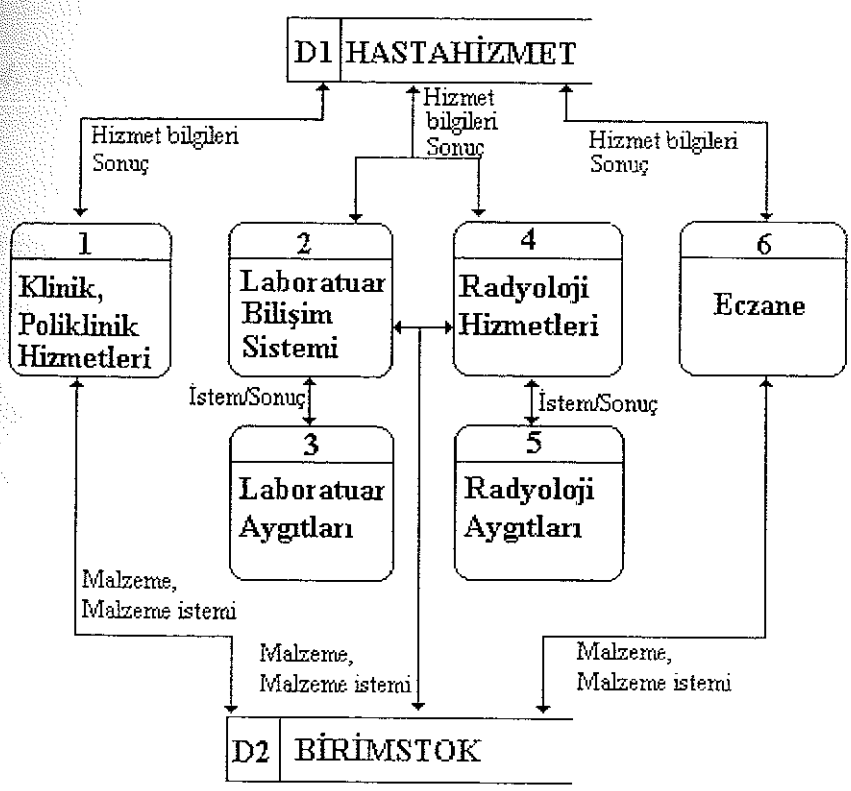
Tahsilat işlemi ise, tahakkuk edilen faturaların kurumlar tarafından yapılan ödemelerinin işlenmesini, izlenmesini, takibini, sorgularını ve gelir defterlerinin

oluşturulmasını gerçekleştirir. Alacaklar konusunda ilgili kurumların uyarı mektuplarını döker ve mali bilgileri tekdüze muhasebe kayıtlarına aktarır.

Vezneler, klinik/poliklinik sekreterlikleri tarafından işlenen hizmetlerin ücretlerinin alınmasını düzenler, kasa defterini döker, öğretim üyeleri döner sermaye miktarlarını düzenler ve bunların kayıtlarını tekdüze muhasebe kayıtlarına aktarır.

3.3.6. Hasta Takip

Hasta takip süreci hastaların tedavi gördükleri süre içerisinde aldıkları tıbbi hizmetleri yöneten bir işlemler bütünüdür. İstemde bulunan ile istemi gerçekleştiren birimler arasındaki bilgi alışverişi ve gerçekleştirilen hizmetlerin tıbbi ve idari niteliklerin gerçek zamanlı (real-time) olarak ilgili birimler tarafından işlenmesi bu süreç içerisinde gerçekleştirilir (Şekil-11).



Şekil-11: Hasta Takip Sürecinin Veri Akış Şeması

3.3.7. Ayniyat, Malzeme Planlama, Satın Alma Hizmetleri

Ayniyat, hastanedeki malzemelerin firmalardan geliş kayıtlarını, stok durumlarını ve birimlere dağıtımını yöneten ve arşivleyen bir süreci kapsamaktadır. İstenen ve dağıtılan malzemelerin birebir eşlenmesi ve veri toplama sürecinin hızlandırılması amacıyla bu modül barkod yazıcı ve okuyucularla desteklenmektedir.

Malzeme planlama ise birimlerin malzeme istemlerine göre hangi malzemeden ne kadar alınacağını veya alınan malzemelerden ne kadarının firmaya sipariş verileceğini yönetir. Yıl içinde her birimin ürettiği hizmetlerin izlenmesi yoluyla bu veriler direkt olarak veritabanından alınabilmektedir.

Satın alma ise malzeme planlama tarafından alınmasına karar verilen malzemelerin ihale yoluyla alınmasını gerçekleştiren bir modüldür. İhale sonucunda üreyen mali bilgiler muhasebe modülünün tekdüze kayıtlarına aktarılır.

3.4. Hastane Bilgisayar Sistemleri Yerleşim Planı ve Yapılandırması

Dağıtık veritabanı modeli olarak tasarlanan HBS iş yükleri göz önüne alınarak işlemler ve veritabanları uygun şekilde dağıtılmıştır. Şekil-12'de HBS'nin ağ omurga yapısı ve bilgisayar sistemlerinin yerleşimleri görülmektedir.

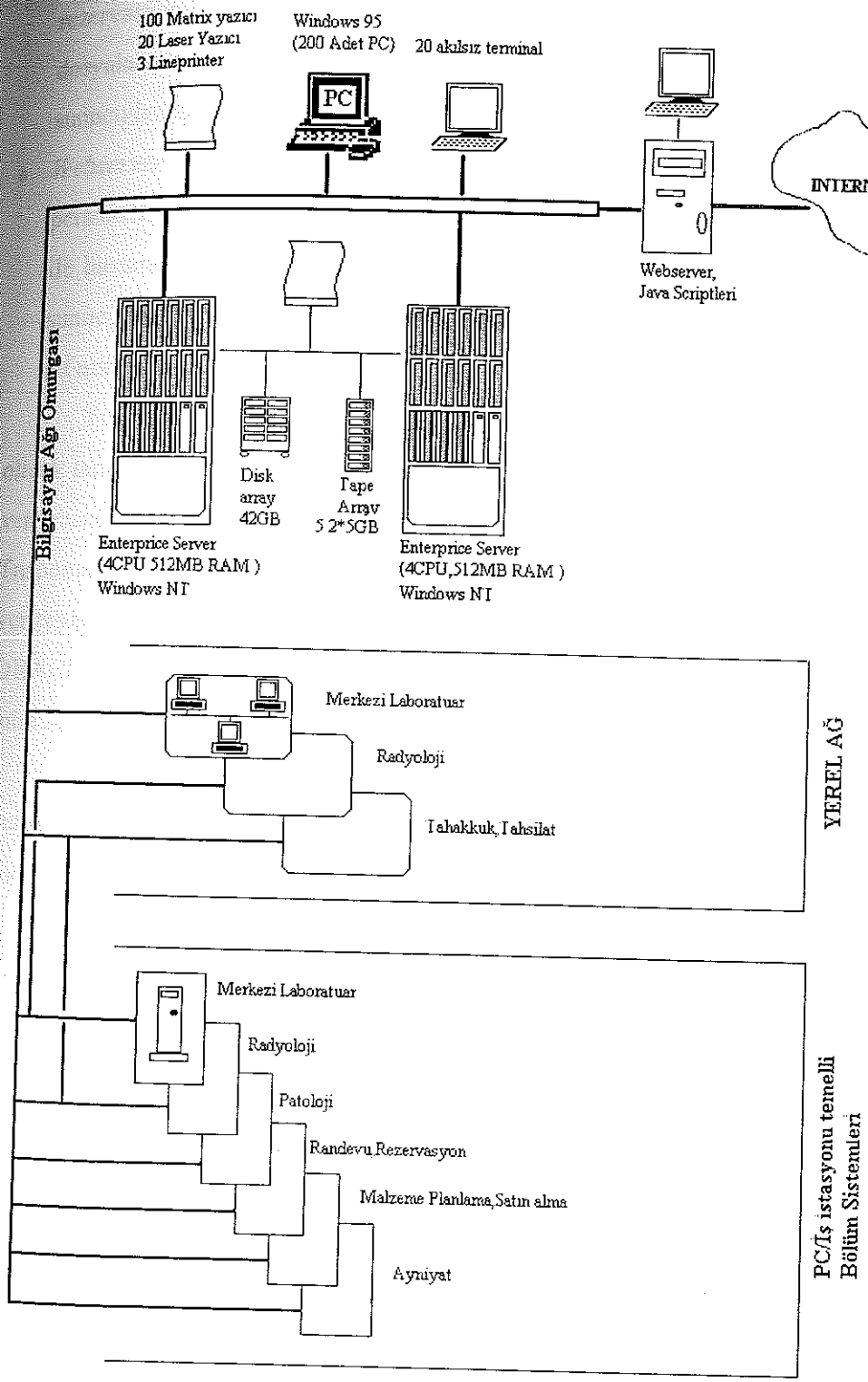
Ana omurganın yükünü arttırmamak ve hastane genel işleyişine etki etmeyen bölümlere özel hizmetler kendi içlerinde bir ağ yapısı kurularak gerçekleştirilmektedir. Bunlar Merkezi Laboratuar, Radyoloji Merkezleri, Bilgi İşlem Merkezi ve tahakuk/tahsilat hizmetleri için tasarlanmaktadır. Her ağ için bir sunucu bulunmaktadır (ORACLE, Windows NT, Workstation, Clarion temelli). Bunun dışında omurgaya bağlı dağıtık veritabanlarını yönetecek olan sunucular bulunmaktadır:

- *Enterprice Server*: İki adet olup, bunlar HBS'nin dağıtık yapısının omurgasını oluştururlar. Diğer bir deyişle dağıtık ORACLE veri tabanlarını Global Veritabanı oluşturmak üzere kurulan veritabanı bağlantılarını (Database Link) yönetirler. Bunun dışında bu sunucular, hastanenin geçmişe ait verilerini arşivlemek, hasta kabul ve hasta takip hizmetlerinde oluşan verileri yönetmek ve yedekleme üniteleri olarak kullanılmaktadırlar. Her biri dört Pentium PRO 200 MHz

işlemcili, 512MB Ana bellekli sunuculardır. Ayrıca her ikisine de bağlı RAID Disk Array'ler ve Tape Array'ler bulunmaktadır. Bu donanımlar disk arızalarında veri kaybını minimum düzeye indirmek, disk alanı sıkıntısını ortadan kaldırmak ve kolay arşivleme amaçları için kullanılmaktadır.

- *Merkezi Laboratuar Sunucusu*: Bu sunucu sekreterlikler tarafından girilen, hekimler tarafından optik formlara işlenen, laboratuar testlerinin yapılması için kullanılmaktadır. Sunuya bağlı ethernet ağı üzerindeki bilgisayarlar Laboratuar aygıtlarına bağlı olup istemleri sunucularından almakla, aygıtlara bu istemleri tanımlamakla, testlerin sonuçlarını alarak hastanın istem bilgilerine işlemekle yükümlüdürler. Bu sistemde ayrıca optik formlara doldurulan tetkiklerin optik okuyularla okunarak testlerin gerçekleştirilmesi de sağlanmaktadır. Sözü edilen bu hizmetin amacı yoğun iş yüküne sahip sekreterliklerin iş yükünü hafifletmek amacıyla kullanılmaktadır.
- *Tahakkuk, Tahsilat Sunucusu*: Hasta taburcu süreçlerinden sonra oluşturulan faturalar bu sunucularda oluşturulmaktadır ve tahakkuk tahsilat işlemlerine temel teşkil eden veriler burada işlemektedir. Ağ iş yükünü hafifletmek amacıyla kurum bilgilerinin birebir kopyası enterprice sunucularının dışında burada da bulunmaktadır. Bu kopyanın birebir tutarlığı ORACLE veritabanı tetikleyicilerle (Database Trigger) sağlanmaktadır.
- *Radyoloji Sunucusu*: Sekreterliklerden işlenen radyoloji istemlerinin alınması ve aygıtlar tarafından gerçekleştirilmesi ve istem kayıtlarına sonuçlarının aktarılması bu sunucu ve sunucuya bağlı ağ üzerindeki PC'ler tarafından gerçekleştirilmektedir. Ayrıca oluşan görüntüler, hekimler tarafından işlenmesi dijital ortamda işlenmesi bu suncu tarafından sağlanmaktadır (Image Processing). Aygıtlarda oluşan görüntülerin hastanın dosyasına gerçek zamanlı aktarılması ve depolanması hedeflenmektedir. Bunun sağlanması için DICOM standardının (yazılım,donanım) hastanemize uygulanması konusunda ayrıca bir olurluk çalışması yapılacaktır.
- *Muhasebe Sunucusu*: Bu sunucu hastanede üreyen mali bilgilerin tekdüze muhasebe kayıtlarına aktarılması, depolanması ve işlenmesi amacıyla kullanılacaktır.

- *Ayniyat, Malzeme Planlama,Satın Alma Sunucusu*: Bu sunucular malzemelerin ortalama tüketim bilgilerini, ana stok durum bilgilerini, ana stoğa giriş ve çıkış bilgilerini, birim stok ve istemleri, ayniyat tarafından birimlere çıkışı yapılamayan



Şekil 12: Hastane Bilgisayar Sistemleri Yerleşim Şeması

malzeme bilgilerini, firmaları, satın alma bilgileri ve siparişleri yöneten verileri tutar. Satın alma sonucunda üreyen mali veriler muhasebe sunucusuna aktarılır.

- *Randevu Rezervasyon sunucusu* Çalışma ve iş gücü planlarını, yapılan rezervasyon ve randuvu kayıtlarını işleyen verileri yönetir. Bu sunucu Clarion dosyalarıyla yerel olarak çalışır ve oluşan bilgiler hasta kabul sırasında bu sunucudan alınır.
- *Webserver*. Hastaneyi internete bağlayan bu sunucu, interneti hastane personellerinin kullanımına sunulmasının dışında Java Scriptleri ve Webserver'lar ile hastanedeki bilgilerin dış kullanıcılar tarafından sorgulanmasını sağlayan bir sunucudur. Ayrıca evlerden randevu/rezervasyon yapma imkanı vermek de mümkündür

Yukarıda sözü edilen sunucular üzerindeki veritabanlarının dağıtık veritabanı olarak tanımlanması için sunucularda ORACLE SQL*NET yazılımına ve kullanılacak veritabanlarının tanımlandığı yapılandırma dosyalarına (Listener.ora, Tnsnames.ora, Tnsnav.ora) gereksinim vardır. SQL*NET ORACLE'ın, uzaktaki veritabanlara erişime olanak sağlayan özel bir iletişim yazılımıdır. Bu yazılım yoluyla ORACLE veritabanlarının her türlü RDBMS işlev ve nesnelerine erişim ve kullanım mümkündür. Bu ürün hem sunucularda hem istemcilerde bulunmak zorundadır.

SQL*Net üzerinden hizmet talep istemcilere, sunucunun hizmet vermesi için mutlaka bir dinleyicinin (Listener) bulunması gerekmektedir.[12] Dinleyicinin konfigurasyonu Listener.ora dosyası ile sağlanmaktadır. Bu dosya şu bölümleri içerir:

- Dinleyicinin adı
- Dinleyicinin adres tanımları.
- Dinleyicinin bulunduğu sunucu üzerindeki veritabanının tanımları.
- Denetim parametreleri.

Aşağıda enterprice sunucusunun örnek dinleyici yapılandırması görülmektedir (Detaylı bilgi için EK_1'e bakınız).

```
LISTENER =                                        !Dinleyici_adı
(ADDRESS_LIST =                                   !Dinleyicinin Servis Tanımı
   (ADDRESS=
    (PROTOCOL=IPC)
    (KEY= Db_hos1)                                ! Servis Adı
   )
   (ADDRESS=
    (PROTOCOL=IPC)
    (KEY= hos1)                                   ! Sunucunun veri tabanı sistem tanımlayıcısı
   )
   (ADDRESS =                                     ! Sunucun ağ adres bilgileri
```

```
    (PROTOCOL = TCP)                          ! Sunucunun üzerinde bulunduğu ağın protokolü
    (HOST = enterprice1)                      ! Sunucunun ağ üzerindeki adı veya IP numarası
    (PORT = 1521)                             ! Dinleyicinin hizmet verdiği port numarası
   )
  )
SID_LIST_LISTENER =                           ! Sunucunun Veritaban(lar)ı Tanımları
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = hos1)                          ! Veritabanı sistem tanımlayıcısı
   (ORACLE_HOME=/d01/home/oracle/product/7 0 16 4)        ! ORACLE ın bulunduğu dizin
  )
 )
STARTUP_WAIT_TIME_LISTENER = 0                'Denetim Parametreleri
CONNECT_TIMEOUT_LISTENER = 10
LOG_DIRECTORY_LISTENER = /listener/tmp
LOG_FILE_LISTENER = sqlnet log
TRACE_LEVEL_LISTENER = ADMIN
TRACE_DIRECTORY_LISTENER= /listener/tmp
TRACE_FILE_LISTENER= listener_trace.trc
```

TNSNames.ora dosyası, dağıtık veritabanı sunucularını, istemcilerin tanımasını sağlayan dosyadır. Bu dosya dağıtık veritabanlarını kullanan tüm istemci ve sunucular için ortaktır. Bu dosya şu bölümleri içermektedir:

- Servis Adı: Hizmeti veren sunucunun servis adı.
- Bağlantı Tanımlayıcıları: Hizmeti veren sunucuların dinleyici adresleri ve veritabanı sistem bilgileri.

Aşağıda, enterprice sunucusunun örnek bir TNSNames.ora dosyası görülmektedir (Detay için EK_1'e bakınız):

```
Db_Hosl =                                     ! Enterprice sunucusunun servis adı
 (DESCRIPTION =                               ! Bağlantı Tanımlayıcısı
  (ADDRESS_LIST =
    (ADDRESS =                                ! Sunucunun adresi
     (COMMUNITY = TCPNET)                     ! Sunucunun bağlı olduğu topluluk adı
     (PROTOCOL = TCP)                         ! TCPIP Protokolü
     (HOST = enterprice1)                     ! Sunucunun ağdaki adı
     (PORT = 1521)                            ! Sunucunun hizmet verdiği port numarası
    )
  )
  (CONNECT_DATA =                             ! Sunucunun veritaban(lar)ı bilgileri
   (SID = hos1)                               ! Veritabanı sistem tanımlayıcısı
  )
 )
```

Dağıtık veri tabanlarının mantıksal olarak bir bütün oluşturulduğu anlatılmıştı. Bu bütünlük veritabanı bağlantıları ile sağlanmaktadır (Database Link). Ancak sistem üzerinde bulunan hangi sunucu için bu bütünlükten söz edebileceğimiz sorusunun yanıtı aramak gerekir. Her sunucu belli bir iş türünü karşılamaktadır, ancak diğer sunucularla aralarında bir ilişki vardır. Her sunucunun işyükü faklı olduğundan bu iş yükünü dağıtmak için her dağıtık veritabanının aynı zamanda global veritabanı

olmasını sağlamak gerekir. Dolayısıyla sistemin genel bütünlüğünü sağlamak ve iş yükünü dengelemek için her sunucu her sunucuyla bağlı olmasını sağlamak gerekir. Dolayısıyla her istemci her sunucudan aynı işi talep edebilir ve sonuçlandırabilir.

3.5.Uygulama Arayüzleri

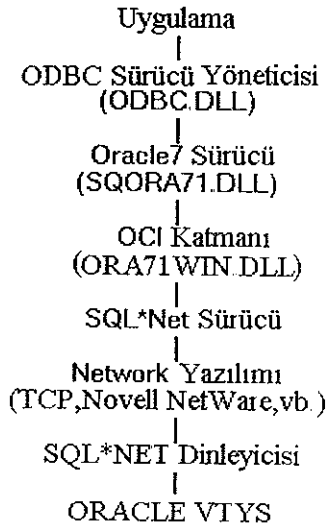
Yukarıda sözü edilen tüm arayüzler, istemci/sunucu mimarisine uygun şekilde Grafik Kullanıcı Arayüz'lerinin(GKA) olanaklarıyla Windows 3.1, Windows 95 ve Windows NT ortamlarında Clarion uygulama yazılım geliştirme aracı kullanılarak geliştirilmektedir. Paylaşımına gerek duyulmayan veriler yerel olarak uygulamanın işletildiği bilgisayarda Clarion dosyalarında tutulacaktır. Paylaşımı gerekli olan veriler ise, yerel ağ üzerinde bulunan bölüm sunucuları (Departmental Server) ve veri sunucularındaki (Enterprice Server) ORACLE veritabanlarında tutulmaktadır. Paylaşımına gereksinim duyulmayan Clarion dosyalarının gelecekte paylaşıma sunulması, Windows NT işletim sisteminin sunucu olma özelliklerinden faydalanılarak yerine getirilecektir. Bu özellik aynı zamanda HBS'nin bakım maliyetini düşüren bir unsur olarak karşımıza çıkmaktadır. HBS istemci/sunucu ve dağıtık veritabanı modelleriyle geliştirildiğinden ötürü uygulama arayüzleri, uzaktaki ORACLE Veritabanlarıyla iletişim kullanmalıdır. Sözü edilen iletişim dört şekilde gerçekleştirilebilmektedir:

- ODBC (Open DataBase Connectivity) yoluyla
- Akdeniz Üniversitesi Hastane Bilişim Sistemi Grubu tarafından geliştirilen Oracle istemci/sunucu arayüzü (OraSrv) yoluyla
- Clarion ORACLE Connect ürünü ile
- OLE (Object Linking & Embedding) yoluyla

Şimdi bu iletişim olanakları tek tek incelenecek, donanım ve yazılım gereksinimlerine ve Oracle nesnelerini kullandırma durumuna göre karşılaştırmalı olarak, HBS'ne en uygun olanı şeçilecektir.

3.5.1. ODBC Yoluyla İstemci/Sunucu İletişimi

ODBC(Open DataBase Connectivity) Windows ortamlarındaki veritabanı dosyalarının (Oracle,Clarion,Dbase,Access,vb.), ODBC desteği veren yazılımlar tarafından kullanılmasını sağlayan bir standarttır. Bir yazılımın, ORACLE tablolarını (uzak veya yerel) ODBC yoluyla kullanması Şekil-14'teki gibi gerçekleşmektedir. Dolayısıyla Şekil-13'teki yazlımların tümünün istemcide bulunması gerekmektedir.



Şekil-13: ODBC yoluyla ORACLE veritabanlarına erişim mimarisi

ODBC yoluyla uzakta bulunan bir ORACLE tablosunun kullanılması için ODBC.ini dosyasının şu şekilde düzenlenmektedir:

```
[ODBC 32 bit Data Sources]          ! ODBC sürücü Yöneticisi
sqlserver=SQL Server (32 bit)

[sqlserver]
Driver32=C:\WINDOWS\SYSTEM\sqlsrv32.dll

[ODBC Data Sources]                 'ODBC dosyaları
db_hos1=Oracle71                    !Enterprice1 sunucusundaki veritabanın servis adı
db_Lis =Oracle71                    'Laboratuar sunucusundaki veritabanın servis adı

[db_hos1]
Driver=C:\WINDOWS\SYSTEM\sqora71.dll      !Veritabanını kullanacak olan ODBC sürücüsü
Server=t:enterprice1:hos1           ! Ağ üzerindeki sunucunun protokol tipi adı ve SID adı

[db_Lis]
Driver=C:\WINDOWS\SYSTEM\sqora71.dll      'Veritabanını kullanacak olan ODBCS sürücüsü
Server=t:lissrv:lis       ! Laboratuar sunucusunun ağ üzerindeki sunucunun protokol tipi,adı ve SID adı
```

ODBC sürücülerinin tanımlanmasında sonra uygulama arayüzleri ilgili veritabanlarındaki nesneleri tanımlanmaları mümkümdür. Aşağıda hastaların hastane içindeki oturumlarının bulunduğu db_hos1 veritabanındaki hkabul tablosunun tanımlanması görülmektedir.

```
Hkabul      FILE,DRIVER('ODBC'),OWNER('db_hos1,hos,torhos'),|
            NAME('hkabul'),PRE(kab),BINDABLE,CREATE,THREAD
kabulkey    KEY(-KabHar:Kabnum),NAME(PK_Kabul),PRIMARY
Rec         RECORD,PRE()
DOSNUM          LONG            !Dosya Numarası
KABNUM          LONG            !Kabul numarası
KABTAR          DATE            !Kabul tarihi
KABTUR          STRING(1)       !Ayakta , Yatan
ISLTUR          STRING(1)       !Resmi , Özel (Acil , Normal)
SEVKNUM         CSTRING(10)     !Kurum sevk numarası
SEVKTAR         DATE            !Sevk tarihi
KURKOD          N(8)            !Kurum Kodu
KURNUM          CSTRING(20)     !Kurum içi sicil no
PRONUM          LONG            !Ücretlendirme protokol numarası
TABTAR          DATE            !Taburcu tarihi veya Çıkış tarihi
            END
        END
```

Sonuç olarak ODBC, fazla sayıda yazılıma gereksinim duyması, SQL desteğinin kısıtlı olması, PL/SQL desteğinin olmaması ve uzaktaki yordamların kullanılaması gibi özelliklerin olamaması dolayısıyla kısıtlı bir araçtır. Bu olanakların olmamasından ötürü ODBC, Akdeniz Üniversitesi Hastanesi Bilişim Sisteminde istemci/sunucu arayüzü olarak kullanılmamaktadır.

3.5.2. Akdeniz Üniv. HBS Grubu Tarafından Geliştirilen Yazılım İle İstemci/Sunucu İletişimi

Akdeniz Univesitesi Hastane Bilişim Sistemini geliştiren grup tarafından 1995 yılında geliştirilen OraSrv iletişim yazılımı, uzaktaki herhangi bir ORACLE veritabanına TCP protokolünü kullanan ağ yazılımı(Winsock.DLL) dışında herhangi bir ek yazılıma gereksinim bırakmamaktadır. Sunucular üzerinde bulunan ve C programlama dili ile geliştirilen bu araç doğrudan soketler üzerinden istemcilerle iletişim kurabilmektedir. İstemci Orasrv sunucusuyla Tablo_3'teki protokole uygun olan komutlarla iletişim kurmaktadır[7]. İletişim sırasında kullanılan ORACLE nesnelerinin istemci arayüzünde tekrar tanımlanması gerekmemektedir. Bu özellik istemcilerin bakım kolaylılığına olanak veren önemli bir özelliktir. OraSrv yazılımı,

PL/SQL, SQL, uzak yordam çağırma (IN, OUT, INOUT parametreleriyle birlikte), arayüz (view), sequence gibi ORACLE VTYS'nin sunduğu tüm nesne ve olanakları kullanabilmektedir.

Mesaj Grubu	Mesaj Tipi	Açıklama	Kullanan Arayüz	Yazım Kuralı	Örnek
Bağlantı	CID	İstemci ID'sini göndererek bağlantı ister	İstemci	CID bağlantı_bilgisi	"CID004"
	CON	bağlantı istemi	Sunucu	CON bağlantı_bilgisi	"CON000" Bağlantı hatası veya "CON001" Bağlantı iletişim#
Denetim	OKE	mesaj alındığına dair onaylama	Sunucu & İstemci	OKE	"OKE"
	REP	son mesajın tekrarlanması	Sunucu & İstemci	REP	"REP"
	CAN	kayıt satırlarının gönderilmesinin kesilmesi	Sunucu & İstemci	CAN	"CAN"
	BYE	İstemin sonuçlanması	Sunucu & İstemci	END	"END"
	COM	taransaction commit	İstemci	COM	"COM"
	ROL	taransaction rollback	İstemci	ROL	"ROL"
Kayıt	RID	sorgu sonucunda donecek olan kayıt sayısı	Sunucu	RID alan_sayısı,hatakodu	"RID00100000000000" veya "RID00001440 "
İstem, Sonuç	PRC	hazır paket yordam isteği	İstemci	PRC kayıt_no, yordam	"PRC0000000000adbul(dosya#) "
	SQL	sql sorgu istemi	İstemci	SQL kayıt_no, sqlcümleciği	"SQL0000000000select "
	DML	insert,update, delete istemi	İstemci	DML kayıt_no text	"DML0000000000delete "
	RES	sonuç satırlarından biri	Sunucu	RES kayıt_sırano, veri alanları	"RES 1 1223|Ali|"

Tablo-2:Orasıv İletişim Protokolü

Orasrv yazılımı Winsock.DLL dosyasındaki yordamları kullanmayı destekleyen tüm program geliştirme araçlarında kullanılabilmektedir. Aşağıda Clarion istemci arayüzünün iletişim yordamı görülmektedir:

```
CODE
...
sock = netopen(host,service,'tcp')                    !Host socket istemi
IF sock < 0 THEN
 msj('Network Hatasi...!')                            ! İletişim hatası
 RETURN
END
sbuf= 'CID001'
adr = ADDRESS(sbuf)
num = netput(adr,SIZE(sbuf),sock)                     !ORACLE bağlantı istemi
adr = ADDRESS(rbuf)
num = netget(adr,1024,sock)
IF CLIP(rbuf) = 'CON000' THEN
 msj('ORACLE acik degil!...')                         !Bağlantı hatası
ELSE
 adr = ADDRESS(txt)
 num = netput(adr,1024,sock)                          !Sorgu iatemi
 LOOP
  adr = ADDRESS(rbuf)
  num = netget(adr,SIZE(rbuf),sock)
  recordsprocessed += 1
  tip = SUB(rbuf,1,3)
  IF tip = 'RID' THEN
   fcnt = SUB(rbuf,4,4)
   sqlcode = SUB(rbuf,8,10)
   IF sqlcode<>'0' AND sqlcode<>'-1405' THEN BREAK. END                    !Sorgu Hatası!
  END
  IF tip = 'END' OR tip = 'BYE' THEN BREAK.                                !İstem tamamlandı
  IF tip = 'RES' THEN                                                      !Kayıt
   txt=''
   s'' = SUB(rbuf,1,14)
   txt = SUB(rbuf,14,LEN(CLIP(rbuf)))
   IF ~OMITTED(1) THEN
    replaceall(txt,'|','')
    que = CLIP(txt)
    ADD(que)
   ELSIF ~OMITTED(2) THEN
    replaceall(txt,'|','')
    grp = txt
    RecordsToProcess = RecordsProcessed
   END
  END
  sbuf = 'OKE'
  adr  = ADDRESS(sbuf)
  num  = netput(adr,SIZE(sbuf),sock)
 END
 sbuf = 'BYE'
 adr  = ADDRESS(sbuf)
 num  = netput(adr,SIZE(sbuf),sock)
END
z# = netclose(sock)
...
```

Sonuç olarak OraSrv yazılımı az donanım ve yazılım kaynakları, bakım kolaylığı sağlaması ve her türlü Oracle nesnelerini kullanma olanağı ile iyi bir tercih olarak görülmektedir. Ancak kurma ve kullanım zorluğu ve buffer yönetiminin aksaması bu yazılımın dezavantajlarıdır.

3.5.3. OLE Yoluyla İstemci/Sunucu İletişimi

OLE, günümüz yazılım mühendisliği teknolojisi olan Nesneye Dayalı Programlama(Object Oriented Programming) mantığını kullanan ve Windows işletim sistemine ait bir standarttır. Bu standardın temel amacı, windows ortamındaki herhangi bir nesnenin herhangi bir ortamda tanımlanması (Class), yaratılması (instance), kullanılmasıdır ve kaldırılmasıdır. Nesneye dayalı yazılım mühendisliğinin tüm özelliklerini taşıyan bu standart OLE desteğini veren tüm yazılımlarda kullanılabilir. Bu tekniğin önemli avantajlarından bir tanesi geliştirildiği yazılımla bir ilişkisinin olmamasıdır. Diğer bir deyişle OLE yoluyla tanımlanan bir nesne sadece üzerinde tanımlandığı yazılımın kaynaklarını kullanır. Bu özellik OLE'nin daha az kaynak kullanmasını gerektirir.

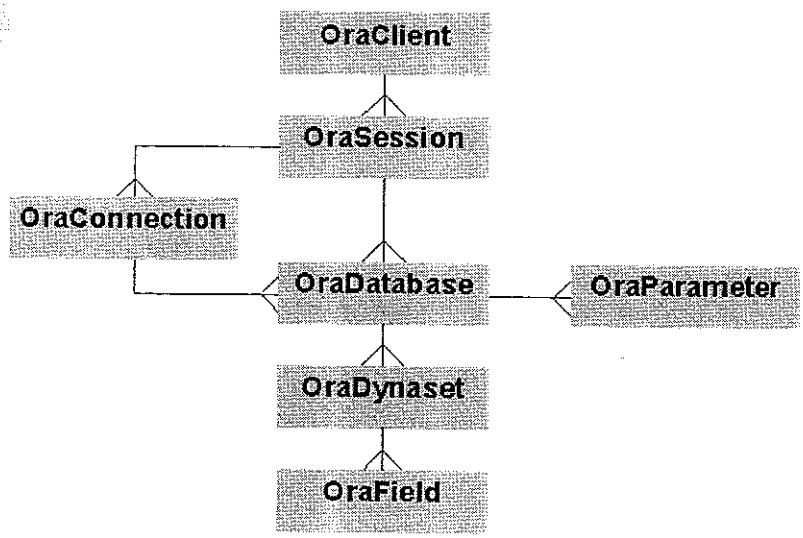
ORACLE Windows ortamları için, ORACLE Objects For Ole'yi(Oo4o) adında bir OLE sunucusu geliştirmiştir. Bu sunucuyu kullanan bir istemcinin gereksinimleri şunlardır:

- Ağ yazılımı (TCP, Novell Network, Micosoft Network,vb.)
- OLE kullanımını destekleyen bir yazılım geliştirme aracı. Örneğin Visual Basic3.0, Clarion 2.0, Delphi,vb.
- Windows 3.1, Windows 95 veya Windows NT.
- Oracle SQL*Net 1.x veya 2.x.
- Microsoft OLE 2.0.1 veya üstü.
- Doğru tanımlanmış TNSNames.ora dosyası

Şekil-14'te OLE kullanımıyla istemci/sunucu iletişimini sağlayan nesneler ve bu nesneler arasındaki ilişkiler görülmektedir. Nesnelerin işlevleri şunlardır:

- OraClient:Oracle ile iletişim kuran işistasyonlarını tanımlayan nesnedir. Bir istemci birçok oracle veritabanında oturum başlatabilir.

- OraSession:Bir uygulama yazılımının Oracle veritabanı üzerindeki oturumunu tanımlayan nesnedir.
- OraDatabase: Bir veritabanından hizmet talep eden istemcilerin veritabanı ile ilişkisini kuran nesnedir. Kullanıcı adı, şifresi ve veritabanı adları verilerek bir OraDatabase nesnesi yaratılabilir.
- OraConnection: Oracle veritabanından yapılan istemler için bir bağlantı gereksinimi vardır ve veritabanı üzerindeki işlemler yapılan bağlantı bilgileriyle tanımlanır. OraConnection nesnesi istemcilerin bağlantılarını tanımlayan nesnedir. Bu nesne OraDatabase nesnesi yaratılırken otomatik olarak yaratılır.
- OraDynaset: SELECT SQL cümleciği ile yaratılan verilerin sorgulanmasını ve günlenmesini sağlayan bir nesnedir. Bu nesne yoluyla istemci verileri alıp işletebilmektedir.
- OraParameter: Oracle sorguları için yerel değişkenleri yöneten nesnedir.
- OraField: Sorgulardan elde edilen veri alanlarını tanımlayan nesnelerdir.



Şekil-14: Oracle Object For OLE'nin Nesne Hiyerarşisi

Aşağıdaki Visual Basic kodunda, enterprice1 sunucusunun db_hos1 veritabanından faturası ödenmeyen hastaların fatura bilgilerini izlenmeyi gerçekleştirmektedir.

```
Sub zzz ()
'OLE Nesnelerinin tanımlanması
Dim OraSession As Object
Dim OraDatabase As Object
Dim OraDynaset As Object

'OraSession nesnesi ile Oturum başlatma
Set OraSession = CreateObject("OracleInProcServer.XOraSession")

'Enterprice1 sunucusunun db_Hos1 veritabanı ile bağlantı gerçekleştirme
```

```
Set OraDatabase = OraSession.OpenDatabase("db_Hos1", "hos/torhos", 0&)
'Sorgu için OraDynaset Nesnesi yaratılmaktadır
Set OraDynaset = OraDatabase.DbCreateDynaset("select
        hasta.ad,hasta.soyad,fatnum,fattar,to_char(fattut,'999,999,999,999')
        from hos.hfatura,hos.hasta
        where odemik is null and hasta.dosnum=hfatura.dosnum", 0&)
'Sorgunun sonuna kadar kayıtlar sunucudan alınmaktadır
Do Until OraDynaset.EOF
 OraDynaset.DbMoveNext
 'islemler
Loop
End Sub
```

Sonuç olarak Oracle tarafından geliştirilen bu yazılım, Oracle'ın tüm veritabanı nesnelerine erişime (PL/SQL ve SQL ile birlikte) ve kullanımına olanak vermesi, kullanım ve kurma kolaylığı sağlaması, sunucu veritabanlarındaki günlemelerin istemci yazılımlarına yansımaması açısından iyi bir tercihtir.

3.5.4. Clarion ORACLE Connect

Bu ürün Topspeed firmasının Clarion yazılım geliştirme aracı kullanılarak ORACLE veritabanlarına (uzak,yerel) erişim için geliştirilmiş bir yardımcı veritabanı sürücüsüdür. Ancak bu ürünün kullanılabilmesi için uygulamanın çalıştığı bilgisayarlarda, ağ yazılımı, SQL*NET yazılımı ve yukarıda sözü edilen gibi bir TNSNames dosyası bulunmalıdır.

Oracle nesneleri, Clarion uygulamalarında tanımlanırken Clarion veri tiplerine göre tanımlanmalıdırlar. Bu eşleştirmeler aşağıdaki tabloya göre yapılabilmektedir[11]

ORACLE Veri tipleri	**Clarion Veri Tipleri**
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)

Tablo-3: ORACLE veri tiplerinin Clarion veri tipleri ile eşleştirilmesi

ORACLE Connect ürünününde, ORACLE dosyaları için aşağıdaki filitreler ve özellikler kullanılabilmektedir:

- **/LOGFILE**: Sunucu tarafından yollanan bilgiler sırasında oluşan SQL cümlecikleri ve sorgular sırasında oluşan hata kodları bir LOG dosyasında tutulurlar. Bu özellik SEND komutuyla ve PROP:LOGFILE filitresi ile sağlanmaktadır. Yazım kuralı ise şu şekildedir:

```
DRIVER('ORACLE','/LOGFILE=logdosyasi[mesaj]')
dosya({PROP:LOGFILE}='logdosyasi[mesaj]'
```

- **/WHERE**: Oracle tablolarının birleştirilmesi (Join) ve kısıtların kurulması bu özellik sayesinde sağlanmaktadır, böylece tablolar arasındaki ilişkiler de kurulmaktadır. Bu filitre üç şekilde kullanılabilmektedir:
    - Bir dosya tanımı birden çok tablonun birleştirilmesini gerektiriyorsa, Clarion her dosyanın hangi veri alanları üzerinden birleştirilmesi gerektiğini belirtmektedir. Örneğin dağıtık veritabanlarındaki hastaların her kabülüne (Enterprice1 sunucusu db_hos1 Veritabanı) ilişkin yapılan istemlerinin(Enterprice2 sunucusu db_hos2 Veritabanı) ilişkisini tek dosya üzerinde kurulması aşağıdaki tanımlar ile yapılmaktadır:

```
Kabul_istem   FILE,DRIVER('ORACLE','/WHERE Kabul.kabnum=hareket.kabnum'),|
        NAME('hos/torhos.hkabul@db_hos1,hos/torhos.hharaket@db_hos2'),|
        PRE(KabHar),BINDABLE,THREAD
kabulkey      KEY(-KabHar:Kabnum),NAME(PK_Kabul),PRIMARY
Rec           RECORD,PRE()
hKabul        GROUP
DOSNUM              LONG          !Dosya Numarası
KABNUM              LONG          !Kabul numarası
KABTAR              DATE          !Kabul tarihi
KABTUR              STRING(1)     !Ayakta , Yatan
ISLTUR              STRING(1)     !Resmi , Özel (Acil , Normal)
SEVKNUM             CSTRING(10)   !Kurum sevk numarası
SEVKTAR             DATE          !Sevk tarihi
KURKOD              N(8)          !Kurum Kodu
KURNUM              CSTRING(20)   !Kurum içi sicil no
PRONUM              LONG          !Ücretlendirme protokol numarası
TABTAR              DATE          !Taburcu tarihi veya Çıkış tarihi
              END
HHareket      GROUP
KABNUM              LONG          !Kabul numarası
ISTNUM              LONG          !İstem numarası, Sequence değer
ISTTUR              STRING(1)     !İstem türü (Resmi ,Özel)
ISTTAR              DATE          !İstem Tarihi
ISTBIR              CSTRING(8)    !İsteyen birim
ISLTUR              STRING(1)     !İşlem türü (Acil,Normal)
YAPBIR              CSTRING       !Yapan birim
PRONUM              LONG          !Ücretlendirme ile ilgili Protokol numarası
```

```
YAPONAY          STRING(1)       !Yapıldı onayı
ACIKLAMA         CSTRING(100)    !İsteme ilişkin notlar
              END
              END
    END
```

- Dosya üzerindeki bir sorgulama sırasında bir kısıt kullanılabilir. Örneğin aşağıda *kabul_istem* dosyasında yatan hastaların kabul ve istemlerini elde etmek için şu şekilde bir sorgu yapılmaktadır:

```
...
CODE
OPEN(Kabul_istem)
SHARE(Kabul_istem,42H)
SENDreturn=SEND(Kabul_istem,'/WHERE kabtur=''Y''')
LOOP
        NEXT(Kabul_Istem)
        !İşlemler
END
```

- Yukarıdaki sorgu PROP:SQL filitresi sağlanarak da gerçekleştirilmektedir:

```
.
CODE
OPEN(Kabul_istem)
SHARE(Kabul_istem,42H)
Kabul_istem({PROP:SQL} = 'SELECT * from Kabul_istem where kabtur=''Y''')
LOOP
        NEXT(Kabul_Istem)
        !İşlemler
END
```

- **/HINT**: Bu özellik sorgularda ORACLE ipuçlarının (Hint) kullanılmasını sağlamaktadır. İpuçları uygulama geliştirenler tarafından belirlenmekte olup ORACLE sorgularını ve optimizasyonu yönlendirmektedir. İpuçları sağlıklı belirlenirse sorguların performansı ve hızı olumlu yönde etkilenir. Yazım kuralları aşağıdaki gibidir. & karekteri önceden tanımlanan ipuçlarının sonuna ekleme yapılır aksi taktirde önceki ipuçları silinir yerine son yazılan ipucu geçerli olur

```
DRIVER('Oracle','/HINT=ipucu')
Anahtar KEY(verialanilistesi),NAME('isim /HINT=[&]ipucu')
SENDreturn = SEND(dosya,' /HINT=[&]ipucu')
dosya{PROP:Hint} = '[&]ipucu'
dosya{PROP:SQL} = 'select /* ipucu */ sqlcümleciği'
```

- PL/SQL ve SQL desteği: Oracle Connect ürününde, ORACLE'ın standart uygulama geliştirme dili olan PL/SQL ve sorgulama aracı olan SQL desteği vardır. Bu olanaklar ile her türlü sorguların, program parçacıklarının ve veritabanına yüklenmiş yordamların (Stored Procedures) kullanılması mümkündür ve bunlar

doğrudan sunucu tarafından işletilmektedirler. Böylece 4.Kuşak bir dil olan PL/SQL ve SQL, 3. Kuşak bir dil olan Clarion'a entegreli olup uygulama geliştirmeyi kolaylaştırmaktadır. Ek-2'de Akdeniz Üniversitesi Hastanesinde temel sorguları gerçekleştiren ve veritabanında yüklü yordam, paketler ve tetikleyiciler görülmektedir. Bunlar PROP:SQL filitresi ile Clarion programlarından işletilmektedirler.

Sonuç olarak ORACLE Connect yazılımı, kurması ve programlaması kolay, ayrıca istemcilerde az donanım kaynağı gerektiren bir yazılımdır. Ancak ORACLE'ın tüm olanaklarının kullanılmasına olanak vermemektedir. Bunlardan bir tanesi OUT ve INOUT tipinde parametre kullanan yordamların kullanılamaması ve SEQUENCE kullanım zorluğudur. ORACLE verilerini kullanan dosyalar ileride local olarak bulunması gerekiyorsa yazılımın bakımı kolaylaşmaktadır. Ancak veritabanlarındaki günlemeler ilgili arayüzlerin de günlenmesini zorunlu kılmaktadır. Bu ise bir dezavantaj olarak karşımıza çıkmaktadır.

Görüldüğü gibi, sözü edilen dört iletişim yazılımlarından olanakları, gerektirdiği kaynaklar ve performans açısından, en uygun olanı OLE yoluyla iletişim yazılımının kullanılmasıdır. Ancak daha öncede anlatıldığı gibi uygulama arayüzleri Clarion ile geliştirilecektir. HBS'ni geliştiren grup Clarion'un 1.5 sürümüne sahiptir ve bu yazılım OLE desteği vermemektedir. Dolayısıyla OLE kullanımı mümkün görünmemektedir. Bu nedenle diğer bir iletişim yazılımı olan ORACLE Connect yazılımının kullanılması uygundur.

4. Sonuç ve Tartışma

İşlemlerin, verilerin ve bilgisayar kaynaklarının paylaşımı amacıyla ortaya çıkan istemci/sunucu mimarisi, kuruluşların bilgisayarlaşma ve sistemin işleyişi sırasında ortaya çıkan yeni gereksinimlerin uyarlanması maliyetini düşürmektedir. Ayrıca istemci/sunucu mimarisi, donanım, yazılım ve protokol bağımlığını ortadan kaldırmış ve buna bağlı olarak Açık Sistem kavramı ortaya çıkmıştır. En önemlisi sistemin, bilgisayar ağı yapısına bağlı olarak performansı merkezi sistemlere göre arttırdığı görülmüştür.

Bu felsefe aynı zamanda veritabanı teknolojisine yansımış ve sistemin bilgi ihtiyacını karşılayan bir global veritabanının işlem ve veri türlerine göre dağıtılmasını ortaya çıkarmıştır. Her veritabanı, hem bağımsız hem de veri ve işlem paylaşımını karşılamak amacıyla da global veritabanının ayrılmaz bir parçasıdır. Parçaların farklı protokollerde olmasının bütünlüğü bozmadığı da gözlenmiştir.

Bu çalışmada, Akdeniz Üniversitesi Tıp Fakültesi Hastanesi için geliştirilen Hastane Otomasyon Sisteminin istemci/sunucu mimarisi çerçevesinde dağıtık veritabanı modeliyle nasıl gerçekleştirileceği tartışılmıştır. Kuşkusuz bir hastane farklı işlevleri olan çok sayıda birimi içinde barındırmaktadır. Ancak bu çok çeşitlilik farklılıktan çok bütünlüğü oluşturan parçalarmış gibi düşünülmelidir. Dolayısıyla veri ve işlem paylaşımının sağlıklı kurulması gereksinimi vardır. Bu gereksinimi karşılamak amacıyla bir sistem çözümleme çalışması yapılmıştır ve HBS'nin bilgi gereksinimi karşılayan gobal veritabanı oluşturulmuştur. Daha sonra bu veritabanının hastanenin işyükü ve işlem türüne göre nasıl dağıtılacağı tartışılmıştır.

İstemci/sunucu mantığına göre geliştirilecek olan istemci arayüzlerinin Windows ortamlarında bağımsız çalışması ve bunların Clarion uygulama geliştirme aracı ile geliştirilmesi önerilmiştir. Ancak veritabanı sunucularındaki verilere erişmek için kullanılabilecek dört iletişim yazılımdan hangisinin uygulama yazılımlarında kullanılacağı üzerinde ayrıca durulmuştur. Sonuç olarak kurma, kodlama ve bakım kolaylığı ile birlikte performans açısında ORACLE Connect yazılımın kullanılmasının uygun olacağına karar verilmiştir.

AKDENİZ ÜNİVERSİTESİ TIP FAKÜLTESİ HASTANESİ İÇİN GELİŞTİRİLEN HASTANE OTOMASYON SİSTEMİNİN DAĞITIK VERİTABANI MODELİ İLE TASARLANMASI

ALTUN Şeyhmus

Akdeniz Üniversitesi

Sağlık Bilimleri Enstitüsü

Yüksek Lisans Tezi

Tez Yöneticisi: YRD.DOÇ.DR.MEHMET YARDIMSEVER

Antalya,1997

ÖZET

Bu çalışmada, Akdeniz Üniversitesi Tıp Fakültesi Hastanesi için geliştirilen Hastane Otomasyon Sisteminin, istemci/sunucu mimarisi çerçevesinde dağıtık veritabanı modeli ile gerçekleştirilmesi için gerekli olan tasarım yapılmıştır. Ayrıca yapılan sistem çözümleme çalışmalarına göre, global bir veritabanı oluşturulmuştur ve bu global veritabanının dağıtık veritabanı modeline nasıl uyarlanacağı tartışılmıştır. İstemci/sunucu arayüzleri performans, bakım ve gerçekleştirim gibi kriterlere göre geliştirilmiştir.

Anahtar Kelimeler: Dağıtık Veritabanı, İstemci,Sunucu, Hastane Bilişim Sistemi

DESIGNING THE HOSPITAL INFORMATION SYSTEM OF THE HOSPITAL OF THE AKDENIZ UNIVERSITY'S FACULY OF MEDICINE WITH DISTRIBUTED DATABASE STRUCTURE

ALTUN Şeyhmus

Akdeniz University

Institute of Health Sciences

M.S. Thesis

Supervisor:Ph.D MEHMET YARDIMSEVER

Antalya,1997

ABSTRACT

In this study, the Hospital Information System, which is developed for the Hospital of the Akdeniz University's Faculty of Medicine, has been designed using the Distributed Database Model and Client/Server Architecture. According to the system analysis, a global Database has been implemented and discussed how this global Database can be made in Distributed Structure. The Client/Server Interfaces have been implemented under the some kinds of critters related with performance, maintenance and development.

Keywords: Distributed Database, Client/Server, Hospital Information System

KAYNAKLAR

1.**YARIMAĞAN Ünal**,"Veri Tabanı Dizgeleri",HÜ/BBM Yayınları,Nisan 1985

2.**MCFADDEN R.Fred,HOFFER A. Jeffrey**,"Modern Database Management",The Benjamin/Cummings Publishing Company Inc, 1994

3.**BOBROWSKI Steven,ARMSTRONG Eric**,"ORACLE7 Server Concepts Manual",December 1992

4.**TÖRECİ Ersin**,"BSG Ve YMT", HÜ/BBM Yayınları,Ankara,1992

5.**ÖCAL Gürhan**, "Dağıtık Ve Merkezi Sistemlerin Entegrasyonunda Karşılaşılan Yapısal Sorunlar Ve Veri Tabanı Entegrasyonu",8.Türkiye Bilgisayar Kongresi, 27-30 Mayıs 1991,İstanbul,S:184-189,

6.**ÇAVUŞOĞLU Esma**,"Client/Server:Bir Bilgi İşlem Felsefesi",Computerworld White Paper,7 Ekim 1996

7.**ŞARLAK Sabutay**,"Akdeniz Üniversitesi Tıp Fakültesi Hastanesi İçin Geliştirilen Hastane Otomasyon Sisteminde Yer Alacak Olan İdari Ve Akademik Modüllerin İstemci/Sunucu (Client/Server) Mimarisi Kullanılarak Geliştirilmesi",Yüksek Lisans Tezi,Antalya,1996

8.**SHORTLIFFE Edward H.,WIEDERHOLD G.**,"Medical Informatics:Computer Application In Health Care", Addison-Wesley Publishing Company, 1990

9.**BOBROWSKI Steven,ARMSTRONG Eric**,"ORACLE7 Server Administrator's Guide", December 1992

10.**YARDIMSEVER Dilek**,"Akdeniz Üniversitesi Tıp Fakültesi Hastanesi Bünyesinde Kurulacak Olan Bilgisayar Ağının Tasarımı",Yüksek Lisans Tezi,Antalya,1993

11. **Topspeed Corporation,**"ORACLE Connect Driver Reference", Topspeed Corporation,1996

12. **NEUMANN Larry,VENNING Sandy,** "SQL*NET Administrator's Guide", Oracle Corporation,1993

EK1: Listener.ora,Tnsnames.ora Dosyaları Yapısı

```
############
# FILENAME: tnsnames.ora
# TIME....: 95-12-11 05:54:30
# NETWORK : AKHOS
# NODE....: Enterprice
# SERVICE : ALL_Servers:Tüm sunucular birbirleriyle bağlı olacağından hepsi için aynı dosya geçerlidir
############
db_Hos1 =
 (DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS =
     (COMMUNITY = TCPNET)
     (PROTOCOL = TCP)
     (HOST = enterprice1)
     (PORT = 1521)
    )
  )
  (CONNECT_DATA =
   (SID = hos1)
  )
 )
db_Hos2 =
 (DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS =
     (COMMUNITY = TCPNET)
     (PROTOCOL = TCP)
     (HOST = enterprice2)
     (PORT = 1521)
    )
  )
  (CONNECT_DATA =
   (SID = hos2)
  )
 )
db_Lis =
 (DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS =
     (COMMUNITY = TCPNET)
     (PROTOCOL = TCP)
     (HOST = lisrv)
     (PORT = 1521)
    )
  )
  (CONNECT_DATA =
   (SID = lis
  )
 )
 db_Radyoloji=
  (DESCRIPTION =
   (ADDRESS_LIST =
     (ADDRESS =
      (COMMUNITY = TCPNET)
      (PROTOCOL = TCP)
      (HOST = imagesrv)
      (PORT = 1521)
     )
   )
   (CONNECT_DATA =
    (SID = img)
   )
  )
```

```
db_Muhasebe=
(DESCRIPTION =
 (ADDRESS_LIST =
  (ADDRESS =
   (COMMUNITY = TCPNET)
   (PROTOCOL = TCP)
   (HOST = muhsrv)
   (PORT = 1521)
  )
 )
 (CONNECT_DATA =
  (SID = muh)
 )
)
db_MPlanlama
(DESCRIPTION =
 (ADDRESS_LIST =
  (ADDRESS =
   (COMMUNITY = TCPNET)
   (PROTOCOL = TCP)
   (HOST = malzemesrv)
   (PORT = 1521)
  )
 )
 (CONNECT_DATA =
  (SID = mal)
 )
)
db_Stok
(DESCRIPTION =
 (ADDRESS_LIST =
  (ADDRESS =
   (COMMUNITY = TCPNET)
   (PROTOCOL = TCP)
   (HOST = stoksrv)
   (PORT = 1521)
  )
 )
 (CONNECT_DATA =
  (SID = stk)
 )
)

###########
# FILENAME: Listener ora
# TIME . .: 95-12-11 05:54:30
# NETWORK : AKHOS
# NODE ...: Enterprice2
# SERVICE : db_Hos2: Bu dosya tüm sunucular için aynı yapıdadır,sadece veritabanı adı,sid ve sunucu adı değişir
###########
LISTENER =                                  'Dinleyici_adı
 (ADDRESS_LIST =                            'Dinleyicinin Servis Tanımı
  (ADDRESS=
   (PROTOCOL=IPC)
   (KEY= Db_hos2)                           ' Servis Adı
  )
  (ADDRESS=
   (PROTOCOL=IPC)
   (KEY= hos2)                              ' Sunucunun veri tabanı sistem tanımlayıcısı
  )
  (ADDRESS =                                ' Sunucun ağ adres bilgileri
   (PROTOCOL = TCP)                         ' Sunucunun üzerinde bulunduğu ağın protokolü
   (HOST = enterprice2)                     ' Sunucunun ağ üzerindeki adı veya IP numarası
   (PORT = 1521)                            ' Dinleyicinin hizmet verdiği port numarası
  )
 )
```

```
SID_LIST_LISTENER =                                    ! Sunucunun Veritaban(lar)ı Tanımları
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = hos2)                                   ! Veritabanı sistem tanımlayıcısı
   (ORACLE_HOME=/d01/home/oracle/product/7.0.16.4)     ! ORACLE ın bulunduğu dizin
  )
 )
LOG_DIRECTORY_LISTENER = /tmp/listener/tmp
LOG_FILE_LISTENER = /tmp/sqlnet log
TRACE_DIRECTORY_LISTENER= tmp//listener/tmp
TRACE_FILE_LISTENER= listener_trace trc
```

EK2: Sunucularda ORACLE Yordam,Paket ve Tetikleyiciler

```
CREATE PACKAGE HOS AD_BUL AS
  FUNCTION hasta_ad_bul (dosyanumarasi number) return char;
  FUNCTION il_ad_bul (ilkodu number) return char;
  FUNCTION birim_ad_bul (birimkodu number) return char;
  FUNCTION hasta_kabulno_bul (dosyanumarasi  number) return number;
  FUNCTION hasta_protokolno_bul (kabulnumarasi  number) return number;
  FUNCTION hizmet_ad_bul (hizmetkodu number) return char ;
  FUNCTION protokol_ad_bul (protokolnum  number) return char;
  FUNCTION malzeme_grp_tur_ad_bul (malzemekod number) return char;
  FUNCTION altkurum_ad_bul (kurumkod  number) return char;
  FUNCTION anakurum_ad_bul (kurumkod  number) return char;
  FUNCTION hasta_dosyano_bul (kabulnumarasi  number) return number;
  FUNCTION hasta_istemno_bul (makbuznumarasi  number) return number;
  FUNCTION kurum_ad_bul(kurumnumarasi number) return char;
  FUNCTION sarfli_mi(hizmetkodu number) return char;
  FUNCTION ilac_ad_bul(ilackodu char) return char;
  FUNCTION ilac_no_bul(ilackodu char) return number;
  FUNCTION ilac_birfiy_bul(ilackod char) return number;
  FUNCTION ilac_birim_bul(ilackod  char) return char;
  FUNCTION eczane_birim_bul(ilackod  char) return char;
  FUNCTION ilcdepo_birfiy(ilackod char) return number;
  FUNCTION kullanici_bul return char;
  FUNCTION ilac_kod_bul(ilacnumarasi number) return char;
  FUNCTION stk_cik_mik(ilackodu char) return number;
  PROCEDURE stoga_aktar(ilackodu char, mik number);
  FUNCTION hizmet_birimfiyat(protokol number, hizmet number) return number;
  FUNCTION hasta_fat_kabul(faturano number) return number;
  FUNCTION icm_kurkod_bul(icmalno number) return number;
  PROCEDURE kurum_borc_ode(borc out number,ode out number,kurumkod number);
  FUNCTION hasta_fatura_bul(kabulno number) return number;
END AD_BUL;
/
  CREATE PACKAGE BODY HOS AD_BUL AS
   -- Koda sahip hastanin goruntulenmesi
   FUNCTION hasta_ad_bul (dosyanumarasi number) return char is
   adsoyad  char(36);
   BEGIN
     select ad||' '||soyad
       into adsoyad
       from hos hasta
       where hasta.dosnum = dosyanumarasi;
     return(adsoyad);
   END hasta_ad_bul;

   -- Koda sahip il adini bulur
   FUNCTION il_ad_bul (ilkodu number) return char is
   il_ad   varchar2(25) ;
   BEGIN
     if ilkodu is not null then
       select ad
         into il_ad
         from hos il
         where il.ilkod = ilkodu;
       return(il_ad):
     else
       return(null);
     END if;
```

```
END il_ad_bul;

-- Isteyen ve yapan birimlerin adlarinin goruntulenmesi
FUNCTION birim_ad_bul (birimkodu number) return char is
birim_ad   varchar2(30);
BEGIN
 if birimkodu is not null then
   select ad
    into birim_ad
    from hos.birim
    where birim.birkod = birimkodu;
   return(birim_ad);
 else
  return(null);
 END if;
END birim_ad_bul;

-- hastanin kabul numarasinin bulunmasi
FUNCTION hasta_kabulno_bul (dosyanumarasi  number) return number is
kabul  number(10) :=0;
BEGIN
   select kabnum
    into kabul
    from hos.hasta
    where hasta.dosnum = dosyanumarasi;
   return(kabul);
END hasta_kabulno_bul;

-- Hastanin protokolunun goruntulenmesi
FUNCTION hasta_protokolno_bul (kabulnumarasi  number) return number is
protokol   number(10) :=0;
BEGIN
   select pronum
    into protokol
    from hos.hkabul
    where hkabul.kabnum = kabulnumarasi;
   return(protokol);
END hasta_protokolno_bul;

-- Koda sahip hizmetin adinin bulunmasi
FUNCTION hizmet_ad_bul (hizmetkodu number) return char is
hizmet_ad   varchar2(50);
BEGIN
 if hizmetkodu is not null then
   select ad
    into hizmet_ad
    from hos.hizmet
    where hizmet.hizkod = hizmetkodu;
   return(hizmet_ad);
 END if;
END hizmet_ad_bul;

-- Koda sahip protokol adinin goruntulenmesi
FUNCTION protokol_ad_bul (protokolnum  number) return char is
protokol_ad  varchar2(50) ;
BEGIN
 if protokolnum is not null then
   select aciklama
    into protokol_ad
```

```
    from hos protokol
    where protokol.pronum = protokolnum;
  return(protokol_ad);
 else
  return(null);
 END if;
END protokol_ad_bul;

FUNCTION malzeme_grp_tur_ad_bul (malzemekod number) return char is
ad     varchar2(25);
BEGIN
   if malzemekod is not null then
     select aciklama
      into ad
      from hos sozluk
      where sozluk.sozkod = malzemekod;
     return(ad);
   else
     return(null);
   END if;
 END malzeme_grp_tur_ad_bul;

 -- Altkurum adini bulur
 FUNCTION altkurum_ad_bul (kurumkod  number) return char is
 kurumad varchar2(40);
 BEGIN
    if kurumkod is not null then
       select ad
         into kurumad
         from hos altkur
         where altkur altkod = kurumkod;
       return(kurumad);
    else
       return(null);
    END if;
  END altkurum_ad_bul;

  -- Anakurum adini bulur
  FUNCTION anakurum_ad_bul (kurumkod  number) return char is
  kurumad varchar2(50);
  BEGIN
     if kurumkod is not null then
        select ad
          into kurumad
          from hos anakur
          where anakur anakod = kurumkod;
        return(kurumad);
     else
        return(null);
     END if;
   END anakurum_ad_bul;

   FUNCTION hasta_dosyano_bul (kabulnumarasi  number) return number is
   dosyanumarasi  number(10);
   BEGIN
      if kabulnumarasi is not null then
        select dosnum
          into dosyanumarasi
          from hos.hkabul
```

```
        where hkabul.kabnum = kabulnumarasi;
      return(dosyanumarasi);
    else
      return(null);
    END if;
  END hasta_dosyano_bul;

  FUNCTION hasta_istemno_bul (makbuznumarasi  number) return number is
  istemnumarasi    number(10);
  BEGIN
     if makbuznumarasi is not null then
          select istnum
             into istemnumarasi
             from hos hharaket
             where hharaket.maknum = makbuznumarasi;
          return(istemnumarasi);
     else
          return(null);
     END if;
  END hasta_istemno_bul;

  FUNCTION kurum_ad_bul(kurumnumarasi number) return char is
  kurumadi    varchar2(50);
  BEGIN
     if kurumnumarasi is not null then
          select ad
             into kurumadi
             from hos.kurum
             where kurum.kurkod = kurumnumarasi;
          return(kurumadi);
     else
          return(null);
     END if;
  END kurum_ad_bul;

  FUNCTION sarfli_mi(hizmetkodu number) return char is
  e_h  char(1);
  BEGIN
    if hizmetkodu is not null then
     select sarf
       into e_h
       from hos hizmet
       where hizmet hizkod = hizmetkodu;
     return(e_h);
    else
     return(null);
    END if;
  END sarfli_mi;

  FUNCTION ilac_ad_bul(ilackodu char) return char is
  ilacadi  varchar2(50);
  BEGIN
    if ilackodu is not null then
     select ad
      into ilacadi
      from hos.ilac
      where ilac.ilckod = ilackodu;
     return(ilacadi);
    else
```

```
  return(ilacadi);
 END if;
END ilac_ad_bul;

FUNCTION ilac_no_bul(ilackodu char) return number is
ilacno number(8);
BEGIN
  if ilackodu is not null then
    select ilcnum
     into ilacno
     from hos.ilac
     where ilac.ilckod = ilackodu;
    return(ilacno);
  else
    return(null);
  END if;
END ilac_no_bul;

FUNCTION ilac_birfiy_bul(ilackod char) return number is
birimfiyat   number(12) :=0;
BEGIN
   if ilackod is not null then
     select birfiy
      into birimfiyat
      from hos.eczane
      where eczane.ilckod = ilackod;
     return(birimfiyat);
   else
     return(null);
   END if;
END ilac_birfiy_bul;

FUNCTION ilac_birim_bul(ilackod  char) return char is
birim  char(5);
BEGIN
   if ilackod is not null then
       select olcbir
        into birim
        from hos.ilac
        where ilac.ilckod = ilackod;
       return(birim);
   else
       return(null);
    END if;
  END ilac_birim_bul;

 FUNCTION eczane_birim_bul(ilackod  char) return char is
 birim  char(5);
 BEGIN
    if ilackod is not null then
        select olcbir
         into birim
         from hos.eczane
         where eczane.ilckod = ilackod;
        return(birim);
     else
        return(null);
     END if;
   END ilac_birim_bul;
```

```
FUNCTION ilcdepo_birfiy(ilackod char) return number is
birimfiyat   number(12) :=0;
BEGIN
  if ilackod is not null then
    select birfiy
     into birimfiyat
     from hos ilac
     where ilac ilckod = ilackod;
    return(birimfiyat);
  else
    return(null);
  END if;
END ilcdepo_birfiy;

FUNCTION kullanici_bul return char is
kullanici varchar2(30);
BEGIN
   select USER
     into kullanici
     from dual;
   return(kullanici);
END kullanici_bul;

FUNCTION ilac_kod_bul(ilacnumarasi number) return char is
ilackodu   char(10);
BEGIN
   if ilacnumarasi is not null then
     select ilckod
       into ilackodu
       from hos ilac
       where ilac ilcnum = ilacnumarasi;
     return(ilackodu);
   else
     return(null);
   END if;
END ilac_kod_bul;

FUNCTION stk_cik_mik(ilackodu char) return number is
miktar   number(12)  :=0;
BEGIN
   if ilackodu is not null then
     select cikmik
      into miktar
      from hos eczane
      where ilckod = ilackodu;
      return(miktar);
    else
      return(null);
    END if;
END stk_cik_mik;

PROCEDURE stoga_aktar(ilackodu char, mik number) as
BEGIN
   update eczane
    set cikmik = nvl(cikmik,0) - mik,
       ciktut = (select nvl(ciktut,0) - birfiy*mik  from eczane
                    where ilckod = ilackodu)
     where ilckod = ilackodu;
```

```
END stoga_aktar;

FUNCTION hizmet_birimfiyat(protokol number, hizmet number) return number is
fiyat  number(12) :=0;
BEGIN
 select birfiy
  into fiyat
  from hos.ucret
  where ucret.hizkod = hizmet and
       ucret.pronum = protokol;
return(fiyat);
EXCEPTION
 when no_data_found then return(0);
END hizmet_birimfiyat;

FUNCTION hasta_fat_kabul(faturano number) return number is
kabulno   number(10) :=0;
BEGIN
  if faturano is not null then
     select kabnum
      into kabulno
      from hos.hfatura
      where hfatura.fatnum = faturano;
     return(kabulno);
   else
     return(0);
   END if;
END hasta_fat_kabul;

FUNCTION icm_kurkod_bul(icmalno number) return number is
kurumkod   number(8) :=0;
BEGIN
   if icmalno is not null then
     select kurkod
      into kurumkod
      from hos.hfatura,hos.hkabul
      where hfatura.icmnum = icmalno and
           hkabul.kabnum = hfatura.kabnum
      group by kurkod;
     return(kurumkod);
   else
     return(0);
   END if;
END icm_kurkod_bul;

PROCEDURE kurum_borc_ode(borc out number,ode out number,kurumkod number) as
BEGIN
   if kurumkod is not null then
     select bormik,odemik
      into borc,ode
      from hos.kurum
      where kurkod = kurumkod;
   END if;
END kurum_borc_ode;

FUNCTION hasta_fatura_bul(kabulno number) return number is
faturano number(10);
BEGIN
   if kabulno is not null then
```

```
    select fatnum
       into faturano
       from hos hfatura
       where hfatura.kabnum = kabulno;
     return(faturano);
   else
     return(0);
   END if;
 END hasta_fatura_bul;
END AD_BUL;
/

CREATE PACKAGE seqpack AS
 FUNCTION seq (dosya CHAR,t NUMBER) RETURN NUMBER;
END seqpack;
/
CREATE PACKAGE BODY seqpack AS
FUNCTION seq (dosya CHAR,t NUMBER) RETURN NUMBER  IS
        s NUMBER(10);
        d NUMBER;
BEGIN
    if dosya='seq_kabnum' then
                select hos.seq_kabnum.NEXTVAL into s from DUAL;
          elsif dosya='seq_maknum' then
                select hos.seq_maknum.NEXTVAL into s from DUAL;
          elsif dosya='seq_dosnum' then
                select hos.seq_dosnum.NEXTVAL into s from DUAL;
          elsif dosya='seq_fatnum' then
                select hos.seq_fatnum.NEXTVAL into s from DUAL;
          elsif dosya='seq_istnum' then
                select hos.seq_istnum.NEXTVAL into s from DUAL;
          elsif dosya='seq_ciknum' then
                select hos.seq_ciknum.NEXTVAL into s from DUAL;
          elsif dosya='seq_girnum' then
                select hos.seq_girnum.NEXTVAL into s from DUAL;
          elsif dosya='seq_icmnum' then
                select hos.seq_icmnum.NEXTVAL into s from DUAL;
          elsif dosya='seq_pronum' then
                select hos.seq_pronum.NEXTVAL into s from DUAL;
          elsif dosya='seq_ilcnum' then
                select hos.seq_ilcnum.NEXTVAL into s from DUAL;
          elsif dosya='seq_partinum' then
                select hos.seq_partinum.NEXTVAL into s from DUAL;
          elsif dosya='seq_altkod' then
                select hos.seq_altkod.NEXTVAL into s from DUAL;
          elsif dosya='seq_kurkod' then
                select hos.seq_kurkod.NEXTVAL into s from DUAL;
       else return(0);
    END if;
     select to_number(to_char(SYSDATE,'YY')) into d from dual;
     if t=0 then
        return(s);
      else
        return(d*100000000+s);
     END if;
   END seq;
 END seqpack;

 CREATE FUNCTION diger_odeme_getir(kurumkodu number)  return number is
```

```
  toplam   number(12) :=0;
BEGIN
      select nvl(sum(hodeme.odemik),0) into toplam
       from hodeme
      where hodeme.kurkod=kurumkodu;
      return(toplam);
EXCEPTION
       when no_data_found then return(0);
END diger_odeme_getir;
/

CREATE PROCEDURE ecz_iade(ilackod char,miktar number,fiyat number) as
BEGIN
  update eczane set girmik = nvl(eczane.girmik,0) - nvl(miktar,0),
                    girtut = nvl(eczane.girtut,0) - nvl(miktar,0)*nvl(fiyat,0)
          where eczane.ilckod = ilackod;
END;
/
CREATE PROCEDURE ilc_iade(ilackod char,miktar number,fiyat number) as
BEGIN
  update ilac set cikmik = nvl(ilac.cikmik,0) - nvl(miktar,0),
                  ciktut = nvl(ilac.ciktut,0) - nvl(miktar,0)*nvl(fiyat,0)
      where ilac.ilckod = ilackod;
END;
/
CREATE PROCEDURE depo_iade(ilackod char,miktar number,fiyat number) as
BEGIN
  update ilac set girmik = nvl(ilac.girmik,0) - nvl(miktar,0),
                  girtut = nvl(ilac.girtut,0) - nvl(miktar,0)*nvl(fiyat,0)
          where ilac.ilckod = ilackod;
END;
/

CREATE PROCEDURE ecz_devir as
BEGIN
  update eczane set devmik = nvl(devmik,0) + nvl(girmik,0) - nvl(cikmik,0),
                    devtut = (nvl(devmik,0) + nvl(girmik,0) - nvl(cikmik,0)) * nvl(birfiy,0),
                    girmik = 0,
                    cikmik = 0,
                    devtar = sysdate
                where trunc(eczane.devtar,'YY')  < trunc(sysdate,'YY') or
                      devtar is null;
   commit;
END ilac_devir;
/

CREATE PROCEDURE borc_devir as
BEGIN
  update kurum set devmik = nvl(devmik,0) + nvl(bormik,0) - nvl(odemik,0),
                   odemik = 0,
                   bormik = 0,
                   devtar = sysdate
                where trunc(kurum.devtar,'YY')  < trunc(sysdate,'YY') or
                      devtar is null;
   commit;
END borc_devir;
/
```

```
CREATE PROCEDURE eczane_giris(miktar number ,birimfiyat number,giristarih date ,ilackodu char)
is
BEGIN
  update hos.eczane set girmik = nvl(eczane.girmik,0) + miktar,
                girtut = nvl(eczane.girtut,0) + miktar * birimfiyat,
                birfiy = birimfiyat,
                girtar = giristarih
       where eczane ilckod = ilackodu;
END;
/

CREATE trigger TRG_BORC_ISLE
after update of onay  on hos hkabul
for each row
when (new.onay ='E' or new onay is null)
declare
   fatura_tut     number(12) :=0;
BEGIN
   select fattut into fatura_tut
       from hos hfatura
       where kabnum = :new kabnum;
   if :new.onay = 'E' then
     update hos kurum set bormik = nvl(bormik,0) + nvl(fatura_tut,0)
        where kurum kurkod = :new.kurkod ;
   elsif :new onay is null then
     update hos.kurum set bormik = nvl(bormik,0) - nvl(fatura_tut,0)
        where kurum.kurkod = :new kurkod ;
    END if;
END;
/
CREATE trigger TRG_BORC_SIL
after delete on hos.hfatura
for each row
declare
   fatura_tut     number(12) :=0;
   kabulno        number(10) :=0;
BEGIN
 if :old.dokctl='E' then
   kabulno := :old kabnum;
   fatura_tut := :old fattut;
   update hos kurum set bormik = nvl(bormik,0) - nvl(fatura_tut,0)
       where kurum kurkod = (select kurkod from hkabul
                        where kabnum = kabulno);
 END if;
END;
/
```

EK3: Akdeniz Üniv. Hastanesinin Bilişim Sisteminin Global Veritabanı Tanımı

SOZLUK

Veritabanında yer alan nesnelerin özelliklerini tanımlayan kodların yer aldığı tablo Bu özellikler iki seviyeli olmalıdır.

Alanlar

SOZKOD	VC(5)	İlişkin nesnenin özellik kodu	(PK)
GRPKOD	VC(10)	Sözluk kodunun üst grubu	(IND)
ACIKLAMA	VC(25)	Nesne özelliğinin açıklaması	

Constraint

SOZKOD	NN,UPPER
GRPKOD	NN,UPPER

İki farklı grup için grup kodunun aynı olmamasına dikkat ediniz.

Örnek

GRPKOD	SOZKOD	ACIKLAMA
CINS	E	Erkek
CINS	K	Kadın
HIZTUR	TET	Tetkik
HIZTUR	ILAC	İlaç
KANGR	ARH+	A Grubu RH Pozitif

ANAKURUM

Hizmet talebinde bulunan hastaların bağlı olabileceği en üst kurum tablosu.

Alanlar

ANAKOD	N(4)	Kurum Ana Kodu	(PK)
AD	VC(50)	Kurum Adı	

Örnek

ANAKOD	AD
1	Milli Eğitim Bakanlığı
2	SSK
3	Emekli Sandığı

IL

Türkiyede yer alan tüm il ve ilçe adlarının bulundugu tablo

Alanlar

ILKOD	N(6)	İl ve ilçe Kodları (3+3)	(PK)
AD	VC(50)	İl ve ya ilçe Adı	

Örnek

ILKOD	AD
007000	Antalya
007001	Korkuteli
006112	Polatlı

KURUM

Hastaneden hizmet talebinde bulunan dışsal varlıklar ,nesneler . (Sevk eşliğinde gelen hastaların sevklerinde yer alan kurumlar veya hastanenin alış veriş yaptığı tüm firmalar,şirketler)

Alanlar

KURKOD	N(8)	Kurum Kodu, KURKODSEQUENCE'dan	(PK)
ANAKOD	N(4)	Kurum Ana Kodu , ANAKURUM tablosundan	(UK,FK)
ILKOD	N(6)	İl Kodu, IL tablosundan	(UK,FK)
SAYKOD	N(8)	Bağlı olduğu saymanlık ,SAYMAN tablosu	(FK)
PRONUM	N(2)	Kurumun aktif protokol numarası, ücret sıra no	
AD1	VC(50)	Kurum Adı,Ana kurum	
AD2	VC(50)	Kurum Adı,Alt kurum	
AD3	VC(30)	Kurum Adı ,İl	
ADRES1	VC(50)	Adres satırı 1	
ADRES2	VC(50)	Adres satırı 2	
POSTKOD	VC(8)	Posta Kodu	

Constraints

KURKOD,ANAKOD,ILKOD,ALTKOD,SAYKOD	NotNull

SAYMAN
Kurumların bağlı bulunduğu ve ödemelerini gerçekleştiren saymanlık tablosu.
Alanlar

SAYKOD	N(8)	Saymanlık Kodu ,SAYKODSEQUENCE'dan	(PK)
AD	VC(100)	Saymanlık Adı	
ADRES1	VC(50)	Adres satırı 1	
ADRES2	VC(50)	Adres satırı 2	
POSTKOD	VC(8)	Posta Kodu	
POSTILKOD	N(6)	Posta il kodu, IL tablosundan	

KHESAP
Kurumun hesaplarının takip edildiği tablo
Alanlar

KURKOD	N(8)	Kurum Kodu , KURUM tablosundan	(PK,FK)
HESTAR	DATE	Hesap tarihi	(PK)
DEVTUT	N(15)	Devreden fatura tutarlarının toplamı	
DEVFAT	N(7)	Devreden fatura sayısı	
TOPTUT	N(15)	Fatura Toplamı	
TOPFAT	N(7)	Toplam fatura sayısı	
ODETUT	N(15)	Ödeme Miktarı	
ODEFAT	N(7)	Ödenen fatura sayısı	
GECTUT	N(15)	Gecikme Faizi tutarı	
GECFAT	N(7)	Gecikmiş fatura sayısı	
UPDTAR	DATE	Bilgilerin son günlendiği tarih	

ORAN
Sistemede yer alacak bazı sabit rakamların yer alacağı tablo. KDV , Gecikme faizi, döviz oranları gibi
Alanlar

KOD	VC(10)	Oran kısa adı	(PK)
ORAN	N(8 3)	Oran değeri	

BIRIM
Hastane bünyesinde hizmet üreten veya hizmet üretimine dolaylı olarak katkıda bulunan tüm birimler
Alanlar

BIRKOD	VC(8)	Harf+BIKODSEQUENCE den oluşan kod	(PK)
TURKOD	VC(5)	Sözlükten birim tur kodu	
SICNUM	VC(10)	Surumlu kişinin sicil numarası	(FK)
AD	VC(50)	Birim Adı	

!

Birim türleri :	Lab , Eczane , Poliklinik , Doktor , vb
Birim kodunda yer alan Harf:	(I) Idari , (H)Hizmet , (A) Akademik

HESAP
Hesap tanımlama tablosu
Alanlar

HESKOD	VC(9)	3+3+3 Seviyeli hesap kodu	(PK)
ACIKLAMA	VC9(50)	Hesap Adı	

BIRIMHESAP
Birimlerin hesap tablosu
Alanlar

HESKOD	VC(9)	Hesap kodu	(FK)
BIRKOD	VC(8)	Birim kodu	(FK)

HIZMET

Hastane bünyesinde verilen tüm hizmetlerin yer aldığı ana hizmet tablosu

Alanlar

HIZKOD	VC(12)	Harf ve Std Hizmet Kodu	(PK)
HIZTUR	VC(5)	Hizmet türü ,Sözlük tablosundan	(FK)
HIZAD	VC(50)	Hizmet Adı	
BIRKOD	VC(8)	Hizmeti üreten Birim Kodu	(FK)
SARF	C(1)	Sarf var mı?	
VERSIYON	C(1)	Versiyon var mı?	
OLCUM	C(1)	Ölçüm var mı?	
RESUCRET	N(12)	Resmi ücreti	
OZLUCRET	N(12)	Özel fiyatı	
PROUCRET1	N(12)	1. Protokole göre ücreti	
PROUCRET2	N(12)	2. Protokole göre ücreti	
PROUCRET3	N(12)	3. Protokole göre ücreti	
PROUCRET4	N(12)	4. Protokole göre ücreti	
PROUCRET5	N(12)	5. Protokole göre ücreti	
PROUCRET6	N(12)	6. Protokole göre ücreti	
PROUCRET7	N(12)	7. Protokole göre ücreti	
PROUCRET8	N(12)	8. Protokole göre ücreti	
PROUCRET9	N(12)	9. Protokole göre ücreti	
PROUCRET10	N(12)	10.Protokole göre ücreti	

!

HIZKOD : X nnnnnn . nnn

X:	L	Lab testleri
	T	Tetkik ve tedaviler
	S	Sarf Malzemeleri
	I	İlaçlar
	J	Majestral İlaçlar
	M	Majestral Malzeme
	D	Destek Hizmetleri
	O	Oda hizmetleri
	Y	Yemekhane Hizmetleri
	X	Diğer Hizmetler
	P	Paneller

Muayene versiyon tipinde olacaktır.Örneğin , 1 poliklinik muayenesi , 1.01 mesai içi poliklinik muayenesi , 1.02 mesai dışı poliklinik muayenesi

BILESEN

Panel sarf v b hizmetlerin bileşenleri

Fileds

USTKOD	VC(12)	Bileşenli Hizmet Kodu	(FK)
HIZKOD	VC(12)	Bileşen Kodu	(FK)
ORAN	N(8.3)	Bileşen Oranı	

!

USTKOD (P xxx , sarfli icin T xxx , L xxx olabilir) Index kurulursa yararlı olabilir

HASTA

Hastaneden hizmet alan hastalara açılan ve kimlik bilgileri ve kurum bilgilerini içeren dosya.

Alanlar

DOSNUM	N(10)	Hastanın fiziksel dosya numarası	(PK)
DOSTAR	DATE	Dosyanın açılış tarihi	
KABNUM	N(12)	Son kabul numarası	
AD	VC(20)	Adı	(IND)
SOYAD	VC(22)	Soyadı	(IND)
CINS	VC(1)	Cinsiyeti	
DOGYER	N(6)	Doğum yeri , IL tablosundan	(FK)
DOGTAR	DATE	Doğum tarihi	
BABA	VC(20)	Baba adı	
ANNE	VC(20)	Anne Adı	
TELEFON	VC(13)	Telefonu	
ADRES1	VC(50)	Adres	
ADRES2	VC(50)		
POSTKOD	VC(10)	Posta kodu	
POSTILKOD	N(6)	Posta il kodu , IL tablosundan	(FK)
KURKOD	N(8)	Hastanın bağlı olduğu kurum kodu	(FK)
KURNUM	VC(20)	Kurum için sicil numarası (SSK , Emekli sandığı)	
BIRKOD	VC(8)	Dosya kimde bilgisini tutar	
ALMTAR	DATE	Dosyanın birim tarafından alınma tarihi	

HOZELLIK

Hastaların belirli bazı özellikleri tutmak için hazırlanmış tablo.

Alanlar

DOSNUM	N(10)	Hasta dosya numarası	(FK)
SOZKOD	VC(5)	Sözlük tablosunda yer alan kod	
GRPKOD	VC(10)	Gurup kodu	

Örnek

DONUM	GRPKOD	SOZKOD
105	HLA	Var
13432	KANGRP	A rh +

SERVIS (ODA)

Hastanede hastalara tahsis edilen oda ve yatak durumlarının tutulduğu tablo.

Alanlar

YATAKNO	N(10)	Yatak numarası	(PK)
ODANO	VC(10)	Oda numarası	
BIRKOD	VC(8)	Odadan sorumlu birim,BIRIM tablosundan	(FK)
YATAKTUR	VC(5)	Yatağın türü, SOZLUK tablosuna	(FK)
DURUMKODU	VC(5)	Durum bilgisi, SOZLUK tablosuna	(FK)
KABNUM	N(12)	Yatağın verildiği hastanın kabul nosu	(FK)
DOSNUM	N(10)	Hastanın dosya numarası	(FK)

!

Dosnum :Yatağı işgal eden hastanın dosya numarası Eger Durum kodu RESERV ise o yatağı reserv eden hastanın dosya numarasıdır

Durum Kodu : Boş , Dolu , Bakımda , Rezerv gibi

Yatak Tür : Normal , 1. Sınıf , 2. Sınıf , 3 Sınıf gibi

HKABUL

Hastaneye başvuran hastaların hastane içindeki hareketlerinin takibini sağlamak amacıyla her gelişte hastaya bir kabul açılır.

Alanlar			
DOSNUM	N(10)	Dosya Numarası	(FK)
KABNUM	N(12)	Kabul numarası	(PK)
FATNUM	N(12)	Fatura numarası	
KABTAR	DATE	Kabul tarihi	
KABTUR	C(1)	Ayakta , Yatan	
ISLTUR	C(1)	Resmi , Özel (Acil , Normal)	
SEVKNUM	VC(10)	Kurum sevk numarası	
SEVKTAR	DATE	Sevk tarihi	
KURKOD	N(8)	Kurum Kodu	(FK)
KURNUM	VC(20)	Kurum içi sicil no	
PRONUM	N(6)	Ücretlendirme protokol numarası	(FK)
TABTAR	DATE	Taburcu tarihi veya Çıkış tarihi	
YAKIN	VC(20)	Yakınının adı	
YAKINLIK	VC(10)	Yakınlık tarihi	
FONAY	C(1)	Fatura basıldı mı?	
YONAY	C(1)	Yatis yapıldı mı?	
DEPOZIT	N(12)	Ön ödeme miktarı	

!
KABTAR ve DOSNUM üzerinde index olmalı

HISTEM

Hastaya verilen tüm hizmetlerin yer aldığı master hizmet dosyasıdır. Istem bazında her istem formuna karşılık gelen tek bir kayıt.

Alanlar			
KABNUM	N(12)	Kabul numarası	(FK)
ISTNUM	N(12)	İstem numarası, Sequence değer	(PK)
ISTTUR	C(1)	İstem türü (Resmi ,Özel)	
ISTTAR	DATE	İstem Tarihi	
ISTBIR	VC(8)	İsteyen birim	(FK)
ISLTUR	C(1)	İşlem türü (Acil,Normal)	
YAPBIR	VC(8)	Yapan birim	(FK)
PRONUM	N(6)	Ücretlendirme ile ilgili Protokol numarası	
MAKNUM	N(10)	Makbuz numarası	
YAPONAY	C(1)	Yapıldı onayı	
ACIKLAMA	VC(100)	İsteme ilişkin notlar	

HHIZMET

Hastaya verilen tüm hizmetlerin yer aldığı detail hizmet dosyasıdır HISTEM dosyasının detay dosyasıdır

Alanlar			
ISTNUM	N(12)	İstem Numarası,HISTEM tablosuna	(FK)
KABNUM	N(12)	Kabul no	(FK)
MAKNUM	N(12)	Makbuz numarasi, HMAKBUZ tablosuna	(FK)
HIZKOD	VC(12)	Hizmetin kodu,HIZMET tablosuna	(FK)
MIKTAR	N(6)	Miktar	
BIRFIY	N(12)	Birim Fiyat	
YAPTAR	DATE	Yapılma tarihi	
VEZONAY	C(1)	Veznede ödeme yapıldı mı?	
VEZODE	N(12)	Veznede ödenen miktar	

!Yarı resmi yarı özel istemler için VEZODE <> NULL , VEZONAY <> NULL için makbuz tutarı VEZODE ile hesaplanır VEZODE = BIRFIY * MIKTAR ise hizmet tamamen özeldir

HMAKBUZ
Vezne makbuzunun master dosyasıdır. Detail olarak HHIZMET dosyasını HISTEM ile ortak olarak kullanır.

Alanlar

MAKNUM	N(10)	Makbuz numarası	(PK)
MAKTUR	VC(5)	Makbuz türü , Sözlük tablosundan	(FK)
VEZKOD	N(3)	Vezne Kodu	
MAKTAR	DATE	Makbuz tarihi	
MAKTUT	N(12)	Makbuz tutarı	
MAKNO	N(12)	Makbuz formunun basılı numarası	

!
MAKNO , makbuz formunda yer alan ve maliye tarafından belirlenen basılı numaradır

HFATURA
Hastaya verilen hizmetleri kapsayan kişiye veya hastanın gönderildiği kuruma ait fatura dosyası

Alanlar

KABNUM	N(12)	Faturanın ait alduğu kabul numarası	(FK)
DOSNUM	N(10)	Hastanın dosya numarası	(FK)
FATNUM	N(12)	Fatura numarası sequence	(PK)
FATTAR	DATE	Fatura tarihi	
KURKOD	N(8)	Faturanın ait olduğu kurum kodu	(FK)
ICMNUM	N(10)	Faturanın yer aldığı kurum icmal numarası	
IPTAL	C(1)	Fatura iptal edildi mi?	
DOKUM	C(1)	Dökümü yapıldı mı?	
FATTUT	N(15)	Fatura toplamı	
ODENDI	C(1)	Ödendi mi?	
ODENUM	VC(15)	Ödeme belge numarası	
ODETAR	DATE	Ödeme tarihi	
ODEMIK	N(12)	Ödeme miktarı	
RESNUM	N(7)	Resmi yazının sayısı	

KODEME
Kurum ödemelerinin takip edildiği dosya.

Alanlar

KURKOD	N(8)	Kurum kodu ,KURUM tablosundan	(FK)
BELNUM	VC(15)	Ödeme belge numarası	
BELTAR	DATE	Belge tarihi	
ODEMIK	N(12)	Ödeme Mıktarı	
ICMNUM	N(10)	İcmal numarası	
RESNUM	N(7)	Resmi sayı numarası	

(FK) :Foreign Key
(PK) :Primary Key
(UK) :Unique Key
(IND) :Indeks

SİSTEMDE YER ALAN SEQUENCES

.KURKODSEQUENCE	Kurum kodunun belirlendiği sequence
.SAYKODSEQUENCE	Sayman kodunun belirlendiği sequence
.KABNUMSEQUENCE	Hasta kabul numarasının belirleyen sequence
.YATAKNOSEQUENCE	Yatak numarasini belirleyen sequence
.ISTNOSEQUENCE	İstem numarasını belirleyen sequence
.MAKNUMSEQUENCE	Makbuz numarasını belirleyen sequence
.FATNUMSEQUENCE	Fatura numarasını belirleyen sequence

TEŞEKKÜR

Bu tezin hazırlanmasında yardımlarını esirgemeyen, danışman hocam sayın Yrd Doç Dr Mehmet YARDIMSEVER'e, sayın hocam Prof Dr Osman SAKA'ya, mesai arkadaşlarım Sabutay ŞARLAK, Yavuz Selim KÖMÜR, Dilek YARDIMSEVER, Gülbin BİLGİÇ ve Gökhan POLAT'a teşekkürlerimi sunarım.