

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GÖRSEL VERİTABANI MODELİ
“İ.T.Ü. SLAYT ARŞİVİ VERİTABANI”

YÜKSEK LİSANS TEZİ

Mimar Ayşegül GÖNENÇ

Anabilim Dalı : Mimarlık

Programı : Bilgisayar Ortamında Mimari Tasarım

Tez Danışmanı: Doç.Dr. Sinan Mert ŞENER

ARALIK 2004

**GÖRSEL VERİTABANI MODELİ
“İ.T.Ü. SLAYT ARŞİVİ VERİTABANI”**

**YÜKSEK LİSANS TEZİ
Mimar Ayşegül GÖNENÇ
5211352**

**Tezin Enstitüye Verildiği Tarih : 27 Aralık 2004
Tezin Savunulduğu Tarih : 24 Ocak 2005**

**Tez Danışmanı : Doç.Dr. Sinan Mert ŞENER
Diğer Jüri Üyeleri Prof.Dr. Gülen ÇAĞDAŞ (İ.T.Ü.)
Doç.Dr. Murat SOYGENİŞ (Y.T.Ü.)**

ARALIK 2004

ÖNSÖZ

Görsel Mimari Veritabanı Modeli “SlidePack” Projesi’nin, mimarlık öğrencileri eğitim ve öğretimine faydalı olabilmesi umuduyla,

Görsel Veritabanı Modeli “İ.T.Ü. Slayt Arşivi Veritabanı” konulu Yüksek Lisans Tez çalışmamda, yapıcı görüş ve eleştirileri ile bana yol gösteren, yardımları için hayatım boyunca minnet duyacağım hocam Sayın **Doç. Dr. Sinan Mert Şener'e**,

Bana her zaman inanan, tüm sevgisi, özverisi, desteği ve inancıyla her an yanımda olan, başarılarımda en büyük payı olan çok sevdiğim **biricik eşime**,

Teşekkürlerimi sunarım.

Aralık, 2004

Ayşegül GÖNENÇ

İÇİNDEKİLER

ÖNSÖZ	ii
İÇİNDEKİLER	iii
KISALTMALAR	vii
TABLO LİSTESİ	viii
ŞEKİL LİSTESİ	ix
ÖZET	xi
SUMMARY	xiii
1. GİRİŞ	1
1.1. Araştırmanın Amacı	1
1.2. Araştırmanın Kapsamı ve Sınırları	2
1.3. Araştırmada İzlenen Yöntem	3
2. VERİTABANI SİSTEMLERİ	4
2.1. Veritabanı Sistemi Tanımı	4
2.2. Veri Yapıları ve Modelleri	6
2.2.1. Temel Veri Yapıları	7
2.2.2. Tanımlamalı Veri Yapıları	8
2.2.3. Veri Modelleri	9
2.3. Veritabanı Sistemlerinin Ortaya Çıkışı ve Gelişme Aşamaları	13
2.4. Veritabanı Sistemlerinin Mimari Yapısı	19
2.5. Veritabanı Sistemlerinin Yeni Kullanım Alanları	22
2.5.1. Mimarlık, Mühendislik Tasarımı ve Üretimi	22
2.5.2. Büro Sistemleri ve Karar Verme Sistemleri	24
2.5.3. İstatistiksel ve Bilimsel Veritabanı Yönetimi	24
2.5.4. Uzman Sistemler	25
2.5.5. Çoklu Ortam Veritabanları	26
2.5.6. "Web" Ve Veritabanı Uygulamaları	26
3. VERİTABANI SİSTEMİ TASARIM AŞAMALARI	28
3.1. Kavramsal Tasarım Aşaması	29
3.1.1. Gereksinimlerin Belirlenmesi ve Analizi	30
3.1.2. Gereksinimler Veri Modelinin Oluşturulması	31
3.1.3. Normalleştirme İşlemi	32
3.1.4. Sistem Kataloğu (Veri Sözlüğü) Oluşturulması	33
3.2. İlişkisel Veritabanı Yönetim Sisteminin Seçimi	35
3.3. Mantıksal Tasarım Aşaması	36
3.3.1. İlişkisel Veri Modeline Uyarlama	37

3.3.2. Hiyerarşik Veri Modeline Uyarlama	37
3.3.3. Ağ Veri Modeline Uyarlama	38
3.4. Fiziksel Tasarım Aşaması	38
3.4.1. Veri Gösteriminin Belirlenmesi	39
3.4.2. Erişim Yöntemlerinin Seçimi	40
3.4.3. Verilerin Dış Belleklere Atanması	40
3.4.4. Veritabanının Yüklenmesi	41
3.4.5. Fiziksel Modelin Değerlendirilmesi	41
3.5. Veritabanı Sistemi Yazılım Geliştirme (Kodlama) Aşaması	43
3.5.1. Yazılım Geliştirme Süreçleri	43
3.5.2. Program/Yazılım Tasarımı	45
3.5.3. Algoritma Tasarımı	46
3.5.3.1. Sıralama Algoritmaları	47
3.5.3.2. Arama Algoritmaları	52
3.5.3.3. Graf Algoritmaları	57
3.5.4. Akış Şemaları	60
3.6. Veritabanı Sisteminin Tamamlanması ve Test Edilmesi	63
3.6.1. Veritabanı Sistemi İçin Kullanıcı Arayüzü Hazırlanması	63
3.6.2. Veritabanı Sisteminin Yüklenmesi	65
3.6.3. Dönüştürme ve Bütünleştirme İşlemleri	66
3.6.4. Veritabanı Sisteminin Test Edilmesi	67
4. VERİTABANI SİSTEMİ YÖNETİMİ	70
4.1. Veritabanı Sistemi Yönetim Fonksiyonu	70
4.1.1. İzleme (Monitoring)	70
4.1.2. Sistem Performansını Düzenleme (Tuning)	71
4.1.3. Sistemde İyileştirme ve Genişletme (Upgrading)	72
4.1.4. Veritabanı Yeniden Düzenleme (Reorganization)	73
4.2. Sistemin Çalışma Hızı ve Bellek Gereksinimi	73
4.2.1. Kıyaslama (Benchmarking)	74
4.2.2. Algoritma Analizi	75
4.2.3. Bellek Gereksinimi	75
4.3. Veritabanı Sisteminin Doğruluğu ve Hata Kaynakları	76
4.4. Veri Sıkıştırma İşlemi	78
4.5. Veri Güvenliğinin Sağlanması	79

5. VERİTABANI SİSTEMİ VE MİMARLIK İLİŞKİSİ	84
5.1. Mimarlıkta Veritabanı Uygulamaları	84
5.2. Tasarımda Veritabanı Uygulamaları	85
5.3. Tasarım Öğreniminde Veritabanı Uygulamaları	89
5.4. Mimari Sunumlarda Veritabanı Uygulamaları	91
5.5. İnternet ve İnternet Teknolojilerinde Veritabanı Uygulamaları	93
5.6. Görsel Mimari Tasarımda Veritabanı Projeleri Örnekleri	94
5.6.1. DOORS-Design Oriented Online Resources System Projesi	94
5.6.2. AVIDOR Projesi	96
5.6.3. The Electronic Cocktail Napkin Projesi	96
5.6.4. Archie Projesi	99
5.6.5. Greatest Hits Slide Library Projesi	100
6. GÖRSEL MİMARİ VERİTABANI İNTERNET UYGULAMALARI	101
6.1. "SPIRO" Projesi / California Üniversitesi	101
6.2. "AVRL" Projesi / California Üniversitesi	103
6.3. "The Great Buildings Collection" Projesi	104
6.4. "Architecture & CAAD, ETHZ" Excite Search Projesi	106
6.5. "The Architectural Index" Projesi	108
6.6. "Dalhouse Slide Library" Projesi	109
6.7. "archINFORM" Projesi	110
6.8. "SMART – Simple Modular Architecture Research Tool " Projesi	111
6.9. "Virginia Tech Art&Architecture Library" Projesi	112
6.10."Walking Melbourne" Projesi	113
7. GÖRSEL MİMARİ VERİTABANI MODELİ -	
"SlidePack" SLAYT ARŞİVİ PROJESİ	114
7.1. Projenin İçeriğine Göre Tanımı	114
7.2. Hazırlanan Model Uygulama İçin Amaç ve Değerlendirme	114
7.3. Elektronik Ortamda Görsel Mimari Veritabanı Oluşumu	116
7.3.1. Paradigma	117
7.3.2. Veritabanı Uygulamasının Yapılanma Süreci	119
7.3.3. Hazırlanan Veritabanı İçin Arama Motorunun Oluşturulması	121
7.3.4. Kullanıcı Arayüzünün Hazırlanması	124
7.3.2. Tamamlanan Projenin İnternet Ortamına Aktarılması ve Test Edilmesi	126
7.4. "SlidePack" Slayt Arşivi Projesinde Kullanılan Bilgisayar Teknolojileri	128

7.4.1. Veritabanı Oluşturulmasında Microsoft Access 2002	128
7.4.2. Kişisel Bilgisayarlarda Server Ortamının Yaratılması İçin IIS 5.1	128
7.4.3. Veritabanı Erişimi İçin Web Arayüzü Oluşturulmasında HTML	130
7.4.4. Web Sayfalarının Etkileşimli Hale Getirilmesinde ASP	131
7.4.5. Web Ortamında Veritabanı Sorgulama İşlemleri İçin SQL	136
7.4.6. HTML Editörü olarak Macromedia Dreamweaver MX	136
7.4.7. Resim ve Fotoğrafların Düzenlenmesinde Adobe Photoshop 8.0 CS	137
7.4.8. Projenin Hazırlanmasında Kullanılan Bilgisayar Donanımları	138
7.5. Görsel Mimari Veritabanı Modelinin İşleyişi	139
7.5.1. Modelin Yapısal İşleyişi (Three-Tier Web Mimarisi)	139
7.5.2. Veritabanına Erişimde Kullanıcı Arayüzünün İşleyişi	141
8. SONUÇ	150
KAYNAKLAR	152
ÖZGEÇMİŞ	157

KISALTMALAR

SQL	: Structured Query Language
CAD	: Computer Aided Design
CAM	: Computer Aided Manufacturing
CAE	: Computer Aided Engineering
CIM	: Computer Integrated Manufacturing
OBE	: Office By Example
QBE	: Query By Example
SSDB	: Scientific And Statistical Database
DBMS	: Scientific Management System
DDL	: Veri Tanımlama Dili
G/Ç	: Giriş Çıkış
I/O	: Input/Output
IPO	: Input Processing Output
DFS	: Dept First Search
BFS	: Breath First Search
GUI	: Graphical User Interface
mb	: megabyte
ADC	: Analog Digital Converter
DES	: Data Encyption Standard
VBA	: Visual Basic Application
CAAD	: Computer Aided Architectural Design
SMTP	: Simple Mail Transfer Protocol
WWW	: World Wide Web
HTML	: Hypertext Markup Language
ID	: Identifier
AVRL	: Architecture Visual Resources Library
ASP	: Active Server Pages
IIS	: Internet Information Server
FTP	: File Transfer Protocol
TCP	: Transmission Control Protocol
IP	: Internet Protocol
POP3	: Post Office Protokol 3
CGI	: Gateway Interface
IDC	: Internet Database Connector
Mbps	: Mega Byte Per Second

TABLO LİSTESİ

	<u>Sayfa No</u>
Tablo 2.1. Veritabanı teknolojisinde 25 yıllık gelişmeler ve geleceğe ait beklentiler.....	14
Tablo 3.1. Yazılımın tasarım süreci adımları konfor değerlendirme tablosu...	43
Tablo 3.2. Araya sokma algoritması davranışı.....	49
Tablo 3.3. Seçmeli sıralama algoritması davranışı.....	50
Tablo 7.1. Kullanılan bilgisayar sistemlerinin konfigürasyonları.....	139

ŞEKİL LİSTESİ

	<u>Sayfa No</u>
Şekil 2.1 : Veritabanı Sisteminin Basit Yapısı.....	5
Şekil 2.2 : Bağımsız Veri Yapılarıyla Topluluk Oluşturulması.....	9
Şekil 2.3 : Bağımsız Veri Yapılarıyla Ortaklık Oluşturulması.....	9
Şekil 2.4 : Çeşitli Veri Modelleri.....	10
Şekil 2.5 : Veritabanı ve Ağ Veri Modelleri.....	11
Şekil 2.6 : Liste Veri Modelleri.....	12
Şekil 2.7 : Ağaç Veri Modelleri.....	12
Şekil 2.8 : Veritabanı Sisteminin Yapısı.....	20
Şekil 2.9 : Veritabanı Sistemlerinin Mimari Yapısı	22
Şekil 2.10 : Mimarlık, Mühendislik Tasarımlarında Veritabanı Sistemi Kullanımı.....	23
Şekil 2.11 : Bütünleşik Uzman Veritabanı Sistemi Yapısı.....	25
Şekil 2.12 : Çoklu Ortam Veritabanı Sistemi Yapısı.....	26
Şekil 2.13 : Veritabanı Tarafından Desteklenen Web Yapısı.....	27
Şekil 3.2 : Kavramsal, Mantıksal ve Fiziksel modeller arasındaki ilişki.....	37
Şekil 3.3 : Metin İçerisinde Sözcük Arama Algoritmasının İşleyişi.....	57
Şekil 3.4 : DFS Yönteminin Örnek Graf Üzerinde Açıklanması.....	58
Şekil 3.5 : BFS Yönteminin Örnek Graf Üzerinde Açıklanması.....	59
Şekil 3.6 : Akış Şeması Örneği.....	60
Şekil 3.7 : Akış Şeması Simgeleri.....	61
Şekil 3.8 : İki Sayının Aritmetik Ortalaması Akış Şeması.....	62
Şekil 3.9 : Koşullu Dallanma Simgesi Örnekleri.....	62
Şekil 3.10 : İç İç Koşullu Dallanma Simgesi Örnekleri.....	63
Şekil 4.1 : Güvenlik Katmanları.....	82
Şekil 5.1 : Great Buildings Arayüzü.....	86
Şekil 5.2 : SPIRO Arayüzü.....	87
Şekil 5.3 : Tasarımda Görsel Sorgulama.....	88
Şekil 5.4 : Öğrenme Sistemi Organizasyonu.....	90
Şekil 5.5 : Mimari Bellekte Diyagramlarla Sorgulama.....	92
Şekil 5.6 : “DOORS” Projesi Oluşturma Modülü Arayüzü	95
Şekil 5.7 : “DOORS” Projesi Strüktürü.....	96
Şekil 5.8 : The Electronic Coctail Napkin Projesi Arayüzü.....	97
Şekil 5.9 : The Electronic Coctail Napkin Çizim Tanımlama Ünitesi.....	98
Şekil 5.10 : The Electronic Coctail Napkin Sonuç Sayfası Arayüzü.....	98
Şekil 5.11 : Archie Projesinde Tasarım Hikayeleri Arayüzü.....	99
Şekil 5.12 : “Greatest Hits” Slayt Kütüphanesi Arayüzü.....	100
Şekil 6.1 : “SPIRO” Görsel Mimari Veritabanı Sorgulama Arayüzü.....	101
Şekil 6.2 : “SPIRO” Görsel Mimari Veritabanı Sonuç Arayüzü 1.....	102
Şekil 6.3 : “SPIRO” Görsel Mimari Veritabanı Sonuç Arayüzü 2.....	102
Şekil 6.4 : AVRL Giriş Sayfası.....	103
Şekil 6.5 : AVRL İndex Sayfası.....	103
Şekil 6.6 : “The Great Buildings Collection” Projesi Giriş Sayfası.....	104
Şekil 6.7 : “The Great Buildings Collection” Projesi Basit Sorgulama	

	Sayfası.....	105
Şekil 6.8	: “The Great Buildings Collection” Projesi Detaylı Sorgulama Sayfası.....	105
Şekil 6.9	: “The Great Buildings Collection” Projesi Sonuç Sayfası.....	106
Şekil 6.10	: “Architecture & CAAD, ETHZ” Projesi Index Sayfası.....	106
Şekil 6.11	: “Architecture & CAAD, ETHZ” Projesi Arama Sayfası.....	107
Şekil 6.12	: “Architecture & CAAD, ETHZ” Projesi Sonuç Sayfası.....	107
Şekil 6.13	: “The Architectural Index” Projesi Index Sayfası.....	108
Şekil 6.14	: “The Architectural Index” Projesi Sorgulama Sayfası.....	108
Şekil 6.15	: “Dalhousie Slide Library” Projesi Giriş Sayfası.....	109
Şekil 6.16	: “Dalhousie Slide Library” Projesi Sonuç Sayfası.....	109
Şekil 6.17	: “archINFORM” Projesi Giriş Sayfası.....	110
Şekil 6.18	: “archINFORM” Projesi Sorgulama Sayfası.....	110
Şekil 6.19	: “archINFORM” Projesi Anahtar Kelime Sayfası.....	111
Şekil 6.20	: “SMART” Projesi Sorgulama Sayfası.....	111
Şekil 6.21	: “Virginia Tech Art & Architecture Library” Projesi Giriş Sayfası.....	112
Şekil 6.22	: “Virginia Tech Art & Architecture Library” Projesi Giriş Sayfası.....	112
Şekil 6.23	: “Walking Melbourne” Projesi Sorgulama Sayfası.....	113
Şekil 7.1	: Access ile Hazırlanmış İlişkili Tablolar Listesi.....	120
Şekil 7.2	: Veritabanı Uygulamasının Öğeler İsimli Tablosu.....	121
Şekil 7.3	: Three-Tier Yapısının Katmanları.....	140
Şekil 7.4	: Üç Katmanlı Mimari Yapının Bölümleri.....	141
Şekil 7.5	: SlidePack Slayt Arşivi Ana Sayfası.....	142
Şekil 7.6	: SlidePack Slayt Arşivinin Arama Sayfaları Arasındaki Bağlantılar.....	143
Şekil 7.7	: SlidePack Slayt Arşivi “Yapı Adına Göre Arama” Sayfası.....	143
Şekil 7.8	: SlidePack Slayt Arşivi “Yapı Adına Göre Arama” Sonuç Sayfası.....	144
Şekil 7.9	: SlidePack Slayt Arşivi “Yapı Adına Göre Arama Sunum” Sayfası.....	145
Şekil 7.10	: SlidePack Slayt Arşivi “Detaylı Arama” Sayfası.....	146
Şekil 7.11	: SlidePack Slayt Arşivi “Slayt Listesi” Sayfası.....	147
Şekil 7.12	: SlidePack Slayt Arşivi “Tüm Slaytlar” Sayfası.....	147
Şekil 7.13	: SlidePack Slayt Arşivi “Tüm Slaytlar” Sayfasında Açılan PopUp Menü	148
Şekil 7.14	: SlidePack Slayt Arşivi “Test” Sayfası	149

ÖZET

GÖRSEL VERİTABANI MODELİ: “İ.T.Ü. SLAYT ARŞİVİ VERİTABANI”

Çalışmamızın konusu, İstanbul Teknik Üniversitesi Mimarlık Fakültesi öğretim üyelerinin katkılarıyla hazırlanan mimari slayt arşivinin, elektronik kayıt ortamında saklanması ve hem internet, hem de intranet aracılığıyla, öğrencilere ve öğretim üyelerine sunulması şeklinde tanımlanabilir. Çalışmamıza başlamadan önce, görsel mimari veritabanı uygulamaları ve bu uygulamalarla ilgili mevcut teknolojiler araştırılmış, bu teknolojilerin kullanım şekilleri, birbirine olan üstünlükleri, yetenekleri, v.s. gibi unsurlar değerlendirilmiştir.

Araştırmamızın temel noktasını veritabanı sistemleri oluşturmaktadır. Araştırmamızın amacını tanımladığımız giriş bölümünün hemen ardından, ikinci bölümde veritabanı sistemlerinin tanımı, ortaya çıkışı ve tarihsel gelişiminden bahsedilmiştir. Yıllardan beri sürekli geliştirilmekte olan bu yazılım sistemlerinin ilerleyişini tesadüfe bağlamak imkansızdır. Yazılım sektörünün veritabanı konusundaki bu gelişimi, ihtiyaçların giderilmesini sağlamak amacıyla yapılan yoğun çalışmalardan ileri gelmektedir. Dolayısıyla veritabanı sistemlerinin geçmişten günümüze kadar gelişimi, aslında bize hangi ihtiyaçlarımızı karşılamak için veritabanı sistemlerini tercih etme eğiliminde olduğumuzu göstermektedir.

Her ne kadar çalışmamızın uygulama bölümünde, sıfırdan bir veritabanı sistemi oluşturmak yerine, Microsoft firmasının ticari ürünü olan Access veritabanı programını kullanmış olsak da, veritabanı sistemin yönetilebilmesi, mantıklı ilişkisel tabloların kurulup, sistemin temellerinin doğru bir şekilde atılması için veritabanı sistemlerinin çalışma mantığını iyice kavramış olmak gerekmektedir. Üçüncü bölüm kapsamındaki veritabanı sisteminin tasarım aşamaları konusu, aynı zamanda bir veritabanı sistemi çalışma mantığının kavranmasına yardımcı olmaktadır. Access gibi hazır bir veritabanı programı ile en basit veritabanının geliştirilmesinde bile mutlaka bilinmesi gereken temel veri yapıları, normalleştirme işlemleri ve arama/sıralama algoritmaları konularının temelleri yine bu bölümde anlatılmaktadır.

Veritabanı sisteminin yönetim ve güvenliğinin doğru bir şekilde sağlanmadığı durumlarda, sistemler giderek yavaşlamakta, eskisi gibi ihtiyaçlara cevap verememekte ve dijital bir kütüphaneden çok dijital bir veri çöplüğü haline gelmektedir. Bu sonuç hem maddi, hem de zaman ve emek kaybıdır. Başarıyla oluşturulmuş veritabanı sisteminin sürekliliğinin sağlanması için, yine bir takım çalışması ile devam ettirilmesi gerekmektedir. Veritabanı sistemlerinin yönetimi, dördüncü bölümümüzün konusunu oluşturmaktadır.

Veritabanı sistemlerinin mimarlıkta, tasarım aşamasında ve sunum aşamasında kullanımı, ilk geliştirilen mimari veritabanı örnekleri, kendi mimari slayt arşivimizin geliştirilmesi çalışmamıza yol göstermiştir. Beşinci bölümde, veritabanı sistemi mimarlık ilişkisi konusu ele alınmaktadır. Mimari veritabanı çalışmamıza yön veren, şu an halen kullanılmakta olan diğer görsel veritabanı örnekleri de altıncı bölümde tanıtılmıştır.

Görsel bir mimari veritabanı modeli olarak oluşturduğumuz slayt arşivi veritabanı uygulaması hakkında detaylı bilgilere yedinci bölümde yer verilmiştir. Bu bölümde veritabanı çalışmamızın tüm süreçleri detaylı olarak ele alınmış, kullanılan yazılım ve donanım teknolojileri anlatılmıştır. Ayrıca slayt arşivi veritabanı sistemimizin işleyişi ekran görüntüleriyle birlikte bu bölümde detaylı olarak tanımlanmaktadır.

Sekiz bölümden oluşan bu çalışmamızın sonuç bölümünde, slayt arşivinin veritabanına aktarılmasının bize sağladığı faydalar ele alınmış, bilgisayar bilimlerinin ortaya çıkardığı yeni yazılım teknolojilerinin mimarlık alanındaki çalışmalara olan katkıları gözden geçirilmiştir.

SUMMARY

VIRTUAL ARCHITECTURAL DATABASE:

“I.T.U SLIDE LIBRARY PROJECT

The subject of our study is keeping Istanbul Technical University’s architectural slide library in electronic environment. The slide library, which is taken from Architectural Department teachers, is defined to be used by students and teachers on internet or intranet technologies. Before starting our study, visual architectural database examples and technological systems used for database applications are searched in detail.

The basic point of our study is making database systems. After defining our purpose on this study in introduction part, the description and historical improving of database systems subjects are defined in the second chapter. The improvement in database system software is for giving answers for the user’s studies and requirements.

In our study, Microsoft Access database program is used. But for making relational database tables and managing database system properly, the logic of database system working subject was studied in detail in the third chapter. Basic data models, normalization and the algorithm of searching and arranging are described in this chapter.

If the security and management of database system can not be provided perfectly, the system will slow down and not be able to answer the user’s requirements. And after a period, the slide library will not be able to work properly. This result may cost much in money, in time and in effort. After composing a database system, a team work is needed to improve and control its work. The subject of the fourth chapter is managing database systems.

Using database systems in architecture, design and presentation phase, first architectural database applications show way to our study on architectural slide library. In chapter five, the relation between database systems and architecture is defined. In chapter six, the examples of digital architectural database systems, used nowadays, are described in detail with screenshots from web pages.

The details of slide library database application, prepared as a visual database system model, are showed in chapter seven. The periods of database design and the technologies used for software are described in detail. This chapter gives information for our study on visual slide library by giving screenshots from our project's interface.

The benefits of transferring the slide library into a digital database and the usage of new software technologies in architecture are summarized in the result part.

1. GİRİŞ

1.1. Araştırmanın Amacı

Bilgisayarların ilk ortaya çıkışıyla, veriler üzerinde direkt kontrol imkanı veren yazılımlar geliştirilmeye başlanmıştır. Bilgisayarın ilk kullanım aşamalarında tüm bilgiler, elektronik depolama birimlerinde sadece ham veri olarak depolanmaktaydı ve mikroişlemciler günümüzde kullanılanlara göre çok daha yavaş olduklarından, yalnızca verilerin doğru bir şekilde kaydedilmesinden ve gerektiğinde tekrar programcılara sunulmasından sorumluydu. Klasik bilgi-işlem felsefesini oluşturan verilerin bilgisayara aktarılması işlemi, yalnızca bu işin eğitimi almış uzmanlar tarafından gerçekleştirilmekteydi. Günümüzde modern bilgi-işlem felsefesinde, veritabanı sistemleri oluşturularak, veriler birbirleriyle ilişkilendirilip, dosyalar halinde düzenlenerek kaydedilmektedir. Geliştirilen yeni yazılım teknolojileri sayesinde, kullanıcı dostu (user-friendly) program arayüz tasarımları geliştirilip, bilgisayarlara bilgi aktarmak ve bilgisayarlardan bilgi almak kolaylaştırılmış, bu gibi basit işlemler için uzman bilgisayar bilimcilerine ihtiyaç kalmamıştır. Sonuç olarak, bu yeni teknolojilerden faydalanan kullanıcı sayısı artmış ve bilgisayarlar artık tarih, sanat, mühendislik ve mimarlık gibi birçok bilim dalının hizmetine sunulmuştur.

Tüm bilim dalları, yeni bilgisayar teknolojilerinden ve veritabanı sistemlerinden maksimum fayda sağlayabilmek adına, yoğun araştırmalar yapmaya başlamıştır. Bu araştırmanın genel amacı da mimari uygulamaların, veritabanı sistemleri ile olan ilişkisini geliştirebilmek, mevcut teknolojilerden maksimum fayda sağlayabilmektir. Ekonomisi, iyileşme süreci içerisine girmiş olan ülkemizde, mevcut yatırımları sonuna kadar kullanmak büyük önem taşımaktadır. Çağdaş ekonomik anlayışa göre “zaman”, paradan çok daha pahalı bir kaynak olarak karşımıza çıkmaktadır. Dolayısıyla, hem mimarlık da dahil olmak üzere tüm bilim dallarında, bilgisayar teknolojileri kullanılarak, mümkün olan en kısa sürede, en doğru sonuca ulaşılması ülkemiz menfaatinedir.

Yaptığımız bu arařtırmada, amalanan diğeri bir konu ise, oğunlukla görsel konuları kapsayan mimari alıřmalarda kullanılmak üzere, bir veritabanı sistemi nasıl görselleřtirilebilir, görsel veriler hangi yollarla veritabanı sistemlerine entegre edilebilir, görsel mimari veritabanı nasıl olmalıdır, sorularına yanıt aramaktır. Görsel mimari veritabanı sistemlerinin oluřturulmasını incelemek ve gerekli olan teknolojik altyapıyı belirlemek arařtırmamızın temel hedefini oluřturmaktadır.

Bu arařtırmanın tamamlanmasında, İ.T.Ü. Mimarlık Fakóltesi öğretim üyelerinin katkısıyla oluřturulmuř olan slayt arřivinin dijital ortama aktarılmasını ve görsel bir mimari veritabanı örneęi olan “SlidePack” projesinin gerekleřtirilmesini amaladık. Bu proje ile veritabanı sistemine ek olarak, sistemin kullanımını kolaylařtıracak ve sistemi internet ortamına taşıyacak, kullanıcı dostu bir arayüz tasarımı oluřturulması, tüm öğrenci ve öğretim üyelerinin ister evden, ister okuldan ya da dünyanın herhangi bir yerinden, geliřtirdiğimiz veritabanına ulařabilmelerinin saęlanması amalanmıřtır.

1.2. Arařtırmanın Kapsamı ve Sınırları

Arařtırmanın temel konusu, görsel mimari veritabanı olması sebebiyle, veritabanı sistemi tanımı, veritabanının geliřim süreci, veritabanı sistemlerinin yapısı bu arařtırmanın kapsamında incelenmiřtir. Ancak arařtırmamızda esas olan veritabanı sistemlerini derinlemesine incelemek deęil, mimari alıřmalarda veritabanı sistemlerinden en iyi řekilde faydalanmaktır. Veritabanı sistemlerinin iřleyiři, paket programların geliřtirilmesi gibi konular bilgisayar yazılım geliřtiricilerinin alıřma konusu olduğundan, arařtırma konumuzun sınırları dıřındadır.

Arařtırmamızdaki temel yaklařım, veritabanı sistemlerine görsel unsurların katılabilirlięi, veritabanı sistemlerinin mimari alıřmalarda kullanımı, veritabanı sistemlerine sıradan kullanıcıların eriřebilmesi gibi konuları ele almaktır.

Arařtırma sonunda örnek bir görsel mimari veritabanı örneęi olan “SlidePack” projesi sunulmaktadır. Bu projenin yazılım kodları, arařtırmamızın inceleme konusu olmamakla beraber, CD ierisinde ek olarak verilmiřtir.

1.3. Arařtırmada İzlenen Yöntem

İki aşamada yürütölen arařtırmamızın ilk adımında veritabanı sistemi temelleri incelenmiştir. İnternet, süreli yayınlar ve diğör tüm kaynaklar arařtırılarak ön bilgi edinilmeye çalışılmıştır. Konuyla ilgili daha önce yapılan tezler ve çalışmalar incelenip gözden kaçan hususlar belirlenmeye çalışılmış, bu konular üzerinde yoğunlaşmaya özen gösterilmiştir. Özellikle veritabanı sistemlerinde arayüz tasarımına yeterince değinilmediğı düşünölerek, bu konu da arařtırmaya dahil edilmiştir.

Arařtırmamızın ikinci adımı ise, örnek görsel bir veritabanı projesi oluşturmaktır. Öncelikle veritabanı sisteminin yapılandırılmasında, projemize yardımcı olabilecek tüm yazılım teknolojileri incelenmiş, yaygınlık, basitlik, çabuk öğrenilebilirlik, maliyet gibi hususlar göz önünde bulundurularak, uygun görölen yazılımlar kullanılmıştır. Örnek veritabanı projesinde hangi yazılım ve programların kullanıldığı, bu programları tercih etmemizin sebepleri ayrıntılı olarak arařtırmamızda yer almaktadır.

2. VERİTABANI SİSTEMLERİ

2.1. Veritabanı Sistemi Tanımı

Bilgisayar sistemleri, bilgiyi otomatik olarak işler ve kullanıcıya faydalanılabilir biçimde sunar. Bilgisayar sistemleri, donanım ve yazılım adı verilen iki alt sistemden oluşmaktadır. Kullanıcı tarafından derlenen ham veriler, belirli bir amaca yönelik ve düzenli biçimde, dosyalar halinde bir dış bellekte (disk, disket, manyetik teyp vb.) saklanmaktadır. Gerektiğinde dosyalardaki kayıtlar, bilgisayar iç belleğine çağırılarak, işleme sokulmaktadır. Çoklu işlem yapan ve çok kullanıcıya olanak sağlayan bilgisayar sistemlerinde, bir ya da birkaç kayıt değil, sayılamayacak kadar çok kayıt işlem görmektedir. Bu durumda, Veritabanı Yönetim Sistemi'ne ihtiyaç duyulmaktadır.

Belirli bir amaçla, bir araya getirilmiş çok sayıda kayıt, bir veri dosyası oluşturmaktadır. Örneğin; mimari yapıların adı, mimarı, bulunduğu şehri gibi maddelerden (items) oluşan kayıtlar (records) bir dosyada toplanmaktadır. Veritabanındaki bir kayda erişim, anahtar kelimeler (keywords) (örneğin mimari yapı adı veya mimarı) ile gerçekleştirilmektedir.

Verileri depolayıp gruplandırarak, eş zamanlı, birden çok kullanıcıya yanıt verebilmesi amacıyla, dosyalar ve tablolar şeklinde düzenlenmiş bilgi yönetimine veritabanı adı verilmektedir. Veritabanı sisteminin amacı, konu ile ilgili tüm bilginin araştırılıp, analiz edilerek ve gereksiz, kullanılmayan bilgi minimize edilerek, veri tabloları ve dosyaları kullanılarak, geleneksel dosya organizasyonları ile ortaya çıkabilecek sorunların çözüme ulaşmasını sağlamaktır. **(Loudon, 1998.**

Management Information Systems)

Verinin değişik veritabanlarında derlenmesi ve değerlendirilerek birleştirilmesi ya da aynı anda çok sayıda kullanıcının giriş yapabilmesi için dosyalar birbirine bağlanabilmektedir (Bütünleşik Dosya Sistemi, Integrated File System).

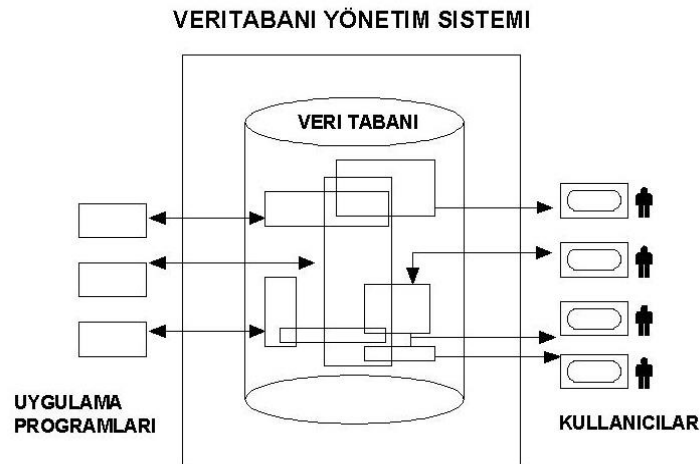
Ancak geleneksel işletim sistemi ile yürütülen bu bütünleşik veri işletim sistemlerinde, sabit kayıtların değişik dosyalarda yinelenmiş olması, bu kayıtlar için ayrı programlar düzenlenmesi ve kayıtların uygun dosyalara geçirilmesi, önemli sakıncalara yol açmaktadır. Bu sakıncalar;

- Verinin dosyalarda ve tablolarda yinelenmesi ve tutarsızlığı,
- Veri girişlerinde güçlüklerle karşılaşılması,
- Birden çok kullanıcının eş zamanlı erişimlerindeki tutarsızlıklar,
- Güvenlik sorunları,
- Bütünleştirme sorunları

olarak özetlenebilir. Bu ve benzeri güçlükler, veritabanı yönetim sistemi oluşturmakla aşılabilmektedir.(Bowers, 1993)

Veritabanı sistemi, çeşitli amaçlar için düzenli şekilde veri saklayan ve isteğe göre bilgi üreten bir bilgisayarlı kayıt tutma sistemidir. Bu sistem, kişinin veya kuruluşun özel ihtiyaçlarını karşılayacak şekilde kurulmaktadır. Genellikle büyük bilgisayar sistemlerinde çok kullanıcı, kişisel bilgisayarlarda ise bir kullanıcı için düzenlenmektedir.

Veritabanı sistemi; veritabanı, veritabanı yönetim sistemi, diğer yazılımlar, donanım ve kullanıcılar olarak, beş öğeden oluşmaktadır. Bu öğelerin basit yapısı Şekil 2.1 de şematik olarak gösterilmiştir.



Şekil 2.1 : Veritabanı Sisteminin Basit Yapısı (Kalıpsız, 2001)

Veritabanı (database); bir sistem halinde düzenlenmiş veri topluluğudur. Belirli bir ya da birden çok amaca yönelik bilgi-işlem sistemi için gerekli verinin, kolay ulaşılabilir ve denetlenebilir şekilde düzenli olarak bir dış bellekte toplanması yoluyla oluşturulmaktadır. Veritabanı sistemiyle, gerektiğinde hemen erişilebilen, düzenli bir veri bankası kurulmakta ve veri tekrarları önlenmiş olmaktadır.

Sistem kullanıcıları, aynı veritabanından kendi uygulaması için gerekli veriye her an erişebilmektedir. Veritabanı, kullanıcının uygulama programından bağımsız olarak düzenlenmekte ve sadece kalıcı veri içermektedir.

Veritabanındaki veri, bütünleşik ve paylaşılabılır (integrated and shared) biçimde düzenlenmektedir. Veritabanlarında çok sayıda bağımsız veri tabloları oluşturularak, bu tablolar anahtar kelimelerle (keywords) birbirine bağlanmaktadır. Böylece, veri tabloları arasında madde (item) ve alan (field) yinelenmeleri önlenmekte, çok sayıda kullanıcı eş zamanlı ve farklı amaçla aynı veriye erişebilmektedir.

2.2. Veri Yapıları ve Modelleri

Veri yapısı (Data Structure), verinin veya bilginin bellekte tutulma şeklini veya düzenini göstermektedir. Tüm programlama dillerinin, genel olarak, tamsayı, kesirli sayı, karakter ve sözcük saklanması için temel veri yapıları vardır. Tasarlanacak program, temel veri yapısı dışında, farklı veri yapılarına ihtiyaç duyuyorsa, bunlar kullanılan programlama dilinin elverdiği ölçüde tanımlanmalıdır. Bu amaçla, temel veri yapıları dışında, yeni veri yapıları tanımlanması için programlama dillerinde bir çok özellik geliştirilmiştir.

Veri modeli (Data Model), verilerin birbirleriyle ilişkisel veya sırasal durumunu gösteren, problemin çözümü için kavramsal bir yaklaşım yöntemidir. Bilgisayar ortamında uygulanacak tüm matematik ve mühendislik problemleri bir veri modeline yerleştirilerek veya yeni veri modelleri tanımlaması yapılarak çözülebilmektedir.

Uygulamada her problem, doğası gereği en uygun veri modeline sahiptir. Bu modelin yazılım geliştiricisi veya programcı tarafından görülebilmesi gerekmektedir. Daha sonra, bu veri modeline dayanılarak algoritmik ifadeler ortaya konulmalıdır.

2.2.1. Temel Veri Yapıları

Tüm programlama dillerinin, genel olarak, karakter, tamsayı, kesirli sayı ve sözcük saklanması için temel veri yapıları vardır. Bazıları da karmaşık sayı, mantıksal değer gibi veri yapılarını da desteklemektedir. Temel veri yapıları aşağıdaki gibi sınıflandırılabilir:

- Karakter (Character) - A, B, C, @, ?, >, V,...
- Tamsayı (IntegerNumber) -1923, 19, 29, 23, 5,-153,...
- Gerçek Sayı (Real Number) - 3.14, 2.71, 1.53-108...
- Katar (String) - "Üniversite", "Oğuzhan", "Can", "A",...
- Dizi/Matris (Array/Matrix) - [23 4 1923],...

Veri yapısı, aslında, ham olarak “1” ve “0” 'lardan oluşan verinin, yorumlanmasını belirleyen veri biçimidir. Örneğin, “62” sayısı bilgisayar belleğinde üç farklı yöntemle tutulabilir; biri sayının ikili tabandaki karşılığı, ASCII kodu ile, bir diğeri de BCD kodlama denilen yöntemle. Dolayısıyla, eğer “62” sayısı birinci yöntemle bellekte tutulursa, 111110 olarak, ikinci yöntemle ASCII karakter kümesine göre 0011011000110010 olarak, BCD kodlamasına göre 01100010 olarak saklanır. Görüleceği gibi bilgi aynı olmasına rağmen bellekte saklanan veri, yapısına göre farklılık göstermektedir. Eğer veri bir grafiğe aitse veya özel simgeler içeren bir yazı parçası ise, verinin bellekte gösterilmesi ve tutulması için özel veri yapıları kullanılmaktadır.

Karakter, en temel veri yapılarından birisi olup, tek tek karakterlerin veya ardarda gelerek sözcüklerin, cümlelerin tutulduğu bir yapıdır. Kodlama işlemi bir karakter tablosuna göre yapılmaktadır. Tablo değiştirildiğinde, karakterlere düşen kodlar değişeceğinden veritabanı, ham verinin içerdiği bilgiyi farklı şekilde yorumlar.

Tamsayı, 3, 5, 2001, 50000 gibi kesirli kısmı olmayan sayılardır. Bir tamsayı, bellekte, en yalın şekliyle ikili tabandaki doğal karşılığıyla saklanır.

Gerçek sayı, tam kısmına ek olarak kesri olan sayılardır. Kesirli sayı veri yapısı, hem kesri olan gerçek sayıların gösterilmesinde, hem de tamsayı veri yapısıyla gösterilmeyecek kadar büyük veya küçük sayıların gösterilmesinde kullanılır.

Bir yazının veya metnin tamamını, herhangi bir cümlesini, sözcüğünü veya hecesini bellek üzerinde tutmak için karakter katarı (string) veri yapısı kullanılmaktadır. Katar, karakterlerin ardarda geldiği bir karakter dizisidir. Sıradan bir karakter dizisinden farkı, metnin kaç elemanlı olduğunun bilinmesi ve “metin sonu karakteri” kullanılmasıdır. Herhangi bir şekilde ardarda gelen karakter dizisi anlamlı bir metin oluşturmayabilir. Metin oluşturması için kaç karakterli veya elemanlı olduğunun bilinmesi veya bir sonlandırma karakteri kullanılması gerekmektedir.

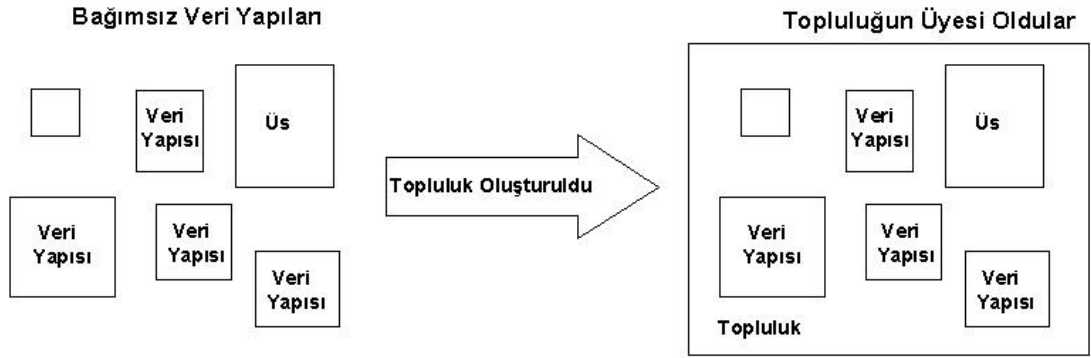
Dizi veri yapısında, kümeye ait olan veriler bellekte ardarda tutulmaktadır. Dolayısıyla kullanıcı, dizinin başlangıç adresini veya adını bildiğinde, dizi veri yapısının herhangi bir elemanına kolayca erişebilmektedir.

2.2.2. Tanımlamalı Veri Yapıları

Tanımlamalı veri yapısı, önceden tanımlanmış temel veri yapıları kullanılarak, oluşturulan yeni veri yapılarıdır. Temel veri yapıları, genellikle tüm uygulamalarda gereksinim duyulan ve bolca kullanılan bir olgudur. Ancak, veri modeli gereği, temel veri yapısıyla oluşturulamayan veri saklama şekilleri tanımlamalı veri yapılarıyla gerçekleştirilir. Yeni veri yapısı, topluluk oluşturma, ortaklık oluşturma ve bit düzeyinde erişim olmak üzere üç değişik şekilde tanımlanabilir.

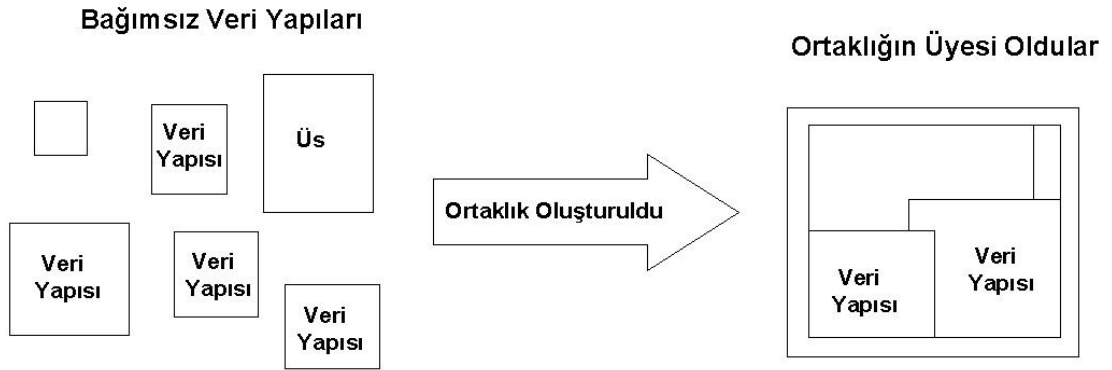
Herbirinin kullanım amacı farklı olup, uygulamaya göre biri veya hepsi bir arada kullanılabilir.

Topluluk, birden çok veri yapısı bir araya getirilerek, yeni bir veri yapısı (aile) oluşturulmasıdır. Bağlantılı liste, ağaç veya özel amaçlı veri modelleri, yeni veri yapılarına gereksinim duymaktadır. Zaman, tarih gibi birden çok değişkenin olduğu veri yapıları topluluk bildirimiyle yapılabilmektedir. Bu tanımlamalar Şekil 2.2 de şematize edilmiştir.



Şekil 2.2 : Bağımsız Veri Yapıları ile Topluluk Oluşturulması (Çölkesen, 2003)

Ortaklık, Şekil 2.3’de görüldüğü gibi, birden çok değişkenin aynı bellek alanını kullanmasını sağlayan veri yapısı tanımlamasıdır. Bellek alanı kısıtlı uygulamalarda, bazı değişkenlerin ortaklaşa kullanılması zorunluluğu olduğundan, birine atama yapıldığında diğerinin değişeceği göz önünde bulundurulmalıdır.



Şekil 2.3 : Bağımsız Veri Yapıları ile Ortaklık Oluşturulması (Çölkesen, 2003)

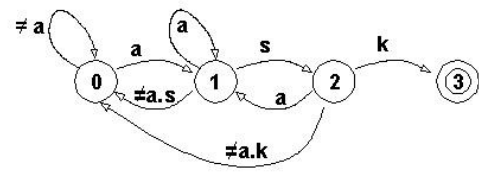
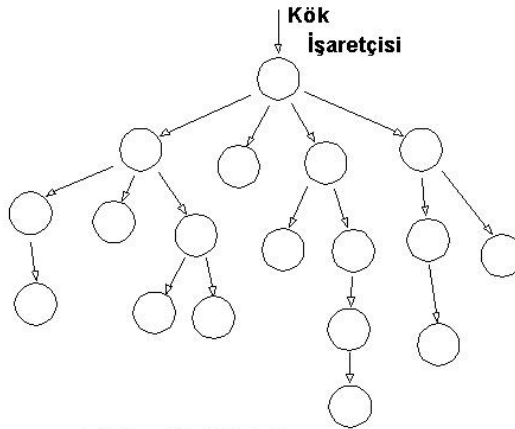
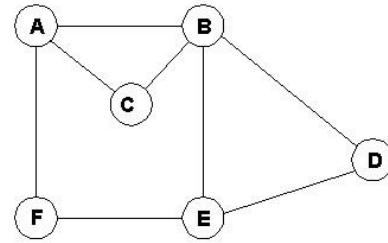
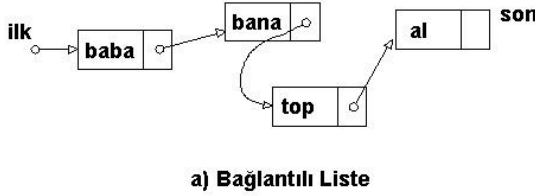
2.2.3. Veri Modelleri

Veri modelleri, aynı kategoride sayılabilecek veriler arasında, ilişkisel ve sırasal düzeni gösteren, kavramsal yaklaşım yöntemleridir. Her program, en azından bir veri modeline dayandırılmaktadır. Uygun veri modelinin seçilmesi ve kullanılması, yapılan uygulama için en etkin programın yazılmasını sağlamaktadır. Etkinlik, bazı uygulamalarda hız, bazılarında ise gereksinimi duyulacak bellek alanı miktarı ile ilgili olarak değişmektedir. Uygun veri modeli araştırılmadan, gerçekleştirilen yazılım tasarımları, çoğu zaman giderilmesi güç sorunlara yol açmaktadır. İşlenmesi gereken veri miktarı arttıkça, ortaya çıkan sorun daha da büyüyebilir. Uygun veri modeli seçilmezse, gereğinden fazla işlem miktarı ve bellek kullanılmış olduğundan

bilgisayarın çalışma hızı ve bellek miktarı arttırılsa da, veri sayısı çoğaldıkça bilgisayar yetersiz kalır.

Veri modelleri, tasarımı yapılacak programın çalışma hızı ve bellek gereksinimi hakkında baştan bilgi vererek, en uygun ve etkin şekilde olmasını sağlamaktadır. Bir kısmı Şekil 2.4’de gösterilen, bilinen veri modelleri aşağıda listelenmiştir. (Çölkesen, 2003)

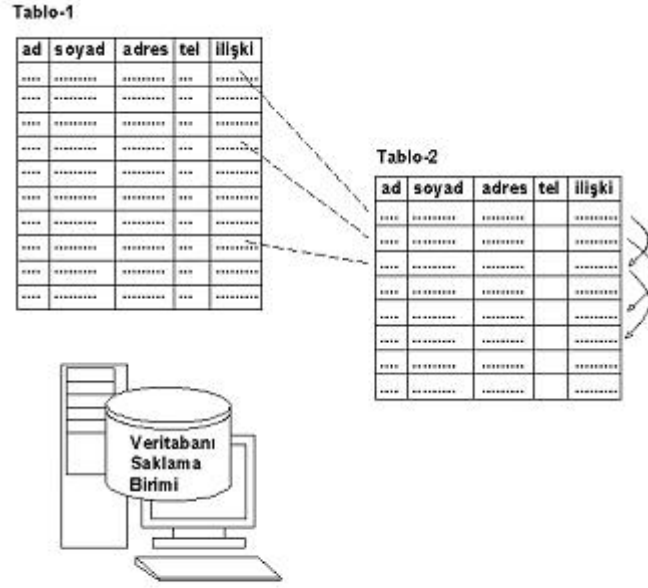
- Bağlantılı Liste (Link List)
- Ağaç (Tree)
- Graf (Graph)
- Durum Makinası (State Machine)
- Veritabanı-İlişkisel (Database Relational)
- Ağ Bağlantı (Network Connection)



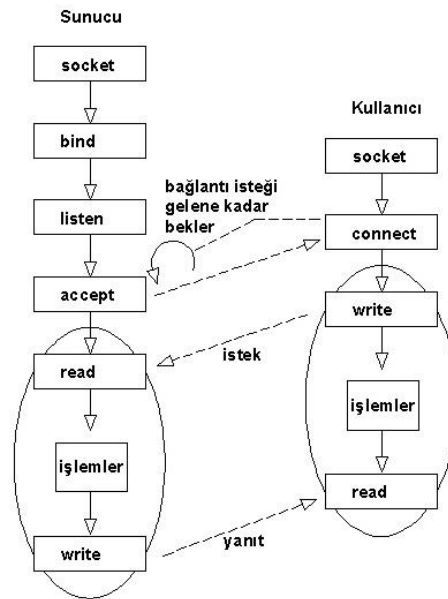
Şekil 2.4 : Çeşitli Veri Modelleri (Çölkesen, 2003)

Bir program tasarlanırken, programın çalışma hızı ile bellek gereksinimi arasında gerekli olan denge kurulmaktadır. Veri modellerine bakıldığında, daha hızlı işlem yapılması istenirse, gereksinim duyulacak bellek miktarının arttırılması

gerekmektedir. Bir uygulama, belirtilen bu veri modeli tiplerinden birine yaklaştırılırsa, uygulamanın gerçekleştirilmesi kolaylaşır. Çünkü, bilinen veri modelleri için birçok algoritma vardır ve program tasarımcısı bu algoritmalardan birini kendi uygulamasında kullanır. Şekil 2.4'de en çok karşılaşılan dört veri modeli gösterilmiştir, bunlar dışındaki veritabanı-ilişkisel ve ağ veri modeli aşağıda, Şekil 2.5'de gösterilmektedir.



a) Veritabanı - ilişkisel

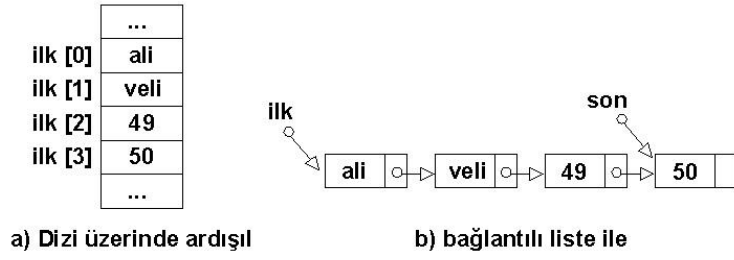


b) Ağ

Şekil 2.5 : Veritabanı Ve Ağ Veri Modelleri (Çölkesen, 2003)

Liste veri modeli, aynı kümeye ait olan verilerin bellekte ardarda tutulması ilkesine dayanmaktadır. Veriler, belirli bir düzen içerisinde olmayabilir ancak, önemli olan tüm verilerin ardarda gelen sırada tutulmasıdır.

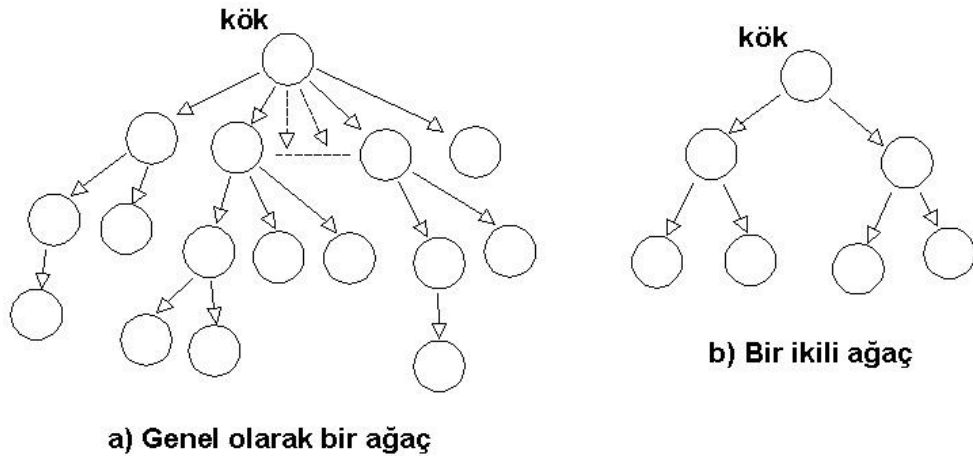
Bağlantılı liste (Link List), Şekil 2.6'da gösterildiği gibi, elemanların kendi değerlerine ek olarak, bağlantı bilgisinin kullanılmasıyla sağlanmaktadır. Bağlantı bilgisi bir sonraki elemanın adresi niteliğindedir.



Şekil 2.6 : Liste Veri Modelleri (Çölkesen, 2003)

Ağaç veri modeli, aynı kümeye ait olan verilerin Şekil 2.7'de görüldüğü gibi, ters çevrilmiş bir ağaç oluşturacak şekilde, sanal bağlanmasıyla elde edilmektedir. Ağacın en altında olan kökler, ağaç veri modelinde en üstte, ağacın üstünde olan yapraklar da modelin altlarında gösterilmektedir. Ağaç veri modeli, özellikle kümenin büyük olduğu ve arama işleminin çok kullanıldığı uygulamalarda etkin bir çözüm sunar.

Ağaç veri modeli, aşağıdaki şekilde görüldüğü gibi, düğümlerden ve dallardan oluşmaktadır. Düğüm noktalarında, verilerin kendileri veya bir kısmı tutulurken, dallar diğer verilere olan bağlantı ilişkilerini gösterir.



Şekil 2.7 : Ağaç Veri Modelleri (Çölkesen, 2003)

Graf veri modeli, aynı kümeye ait olan verilerin düğümler ve ayrıtlarla birleştirilmesinden oluşmaktadır. Düğüm noktaları birleşme noktasını, ayrıtlar da düğümlerin bağlantı ilişkisini göstermektedir. Verilerin kendileri veya bir kısmı, hem düğümlerde, hem de ayrıtların bilgi kısmında tutulabilmektedir.

Graf veri modelinde, ağaç veri modelinde olduğu gibi, hiyerarşik bir durum yoktur ve her düğüm noktası eşit hiyerarşidedir. Tersı söylenmediğı sürece, ağaç veri modeli tanımının aksine, bir ayrıt iki yönlü bağlantıyı göstermektedir. Graflar, yazılım dünyasında önemli bir yere sahiptir. Uygulamada birçok problemin çözümü, graf veri modellerine dayandırılmaktadır.

Durum makinesi veri modeli, bir sistemin davranışını tanımlamak ve ortaya çıkarmak için kullanılan bir yaklaşım şeklidir.

Tablolar üzerinde kurulan ilişkisel veri modeli, veritabanı uygulamalarında var olan sınıflardan birisidir. SQL (Structured Query Language), sorgulama dili kullanılarak veritabanı üzerinde sorgulama yapılabilir. Access, Microsoft SQL, Oracle, Sybase, Ingres gibi birçok veritabanı yönetim sistemleri, ilişkisel veri modelini desteklemektedir. Veritabanı yönetim sistemleri, veritabanı oluşturma, tablo yaratma, alanları tanımlama gibi işlerin yapılmasını ve genel olarak veritabanı yönetimini sağlamaktadırlar.

Ağ veri modeli, modern katmanlı ağ mimarilerinde, uç düğümler (bilgisayarlar) arasında eş katmanlar düzeyinde veri alış-verişini sağlayan çerçeve yapılarını ortaya koyar ve iletişim için gerekli davranışı tanımlar.

2.3. Veritabanı Sistemlerinin Ortaya Çıkışı Ve Gelişme Aşamaları

İşletmelerde bilgisayar sistemi, kayıtları işlemek ve kayıtlar arasında ilişki kurmak amacıyla kullanılmaya başlanmıştır. İlk uygulamalarda, iç bellek kapasitesinin sınırlı oluşu nedeni ile, işlem sırası gelmeyen veri ve programların dosyalar halinde dış bellekte saklanması yoluna gidilmiştir (Dosya İşlemleri - File Processing). İşlemlerin artması ve çeşitlenmesi, bilginin değişik kaynaklardan derlenmesi ve eşzamanlı ve çoklu kullanım ihtiyaçları nedeniyle, dosyaların ve tabloların birbirine bağlanması, bir çözüm olarak görülmüştür. Dosya sistemi (Integrated File System) adı verilen bu

sistemde, aynı verinin çok sayıda dosyada yinelenmesi, uygulama programına bağıllık, dosyalar arası uyumsuzluk, verinin kullanıcıya gösterim güçlüğü vb. gibi yetersizlikleri karşılamak üzere, veritabanı sistemi geliştirilmiştir. Bilgisayar donanımı ve yazılımı teknolojisinde, özellikle son otuz yılda giderek artan bir hızla getirilen yenilikler, veritabanı sisteminin kurulmasına ve hızlı gelişimine dayanak olmuştur. Otuz yıllık geçmişe sahip olan veritabanı sistemi, bu kısa süre içerisinde geliştirilerek, bugünkü düzeyine ulaştırılmıştır. Başlangıçta sadece bir işletmenin veya kurumun bilgi-işlem ihtiyacını karşılamak amacı ile kurulan veritabanı sistemi, günümüzde mühendislik ve mimarlık tasarımı ve üretimi, büro sistemleri ve karar verme sistemleri, istatistik ve bilimsel araştırma, bilgisayar yazılım mühendisliği, uzman sistemler konularında da yaygın olarak kullanılmaktadır.

Veritabanı sisteminin günümüzdeki düzeye ulaşmasında; 1965-1975, 1975-1985, 1985-1990 ve 1990 sonrası dönemleri olarak, dört basamak ayırt edilmektedir. Bu basamaklara ait veri modeli, donanım, kullanıcı arabirimi, program arabirimi, gösterim ve ekran görüntüsü, bilgi işlem biçimi bakımlarından değişiklikler ve gelişmeler, Tablo 2.1 'de özetlenmiştir.

Tablo 2.1 : Veritabanı teknolojisinde 25 yıllık gelişmeler ve geleceğe ait beklentiler.(Kahpsız, 2001)

VERİTABANI TEKNOLOJİSİNDEKİ GELİŞMELER				
	1965 - 1975	1975 - 1985	1985 - 1990	1990 SONRASI
VERİ MODELİ	Ağ Yapısı Hiyerarşik	İlişkisel	Semantik Nesneye Yönelik Mantıksal	Birleşik Veri Modelleri Bilgi Gösterimi ve Programlama Dilleri
VERİTABANI DONANIMI	Büyük Sistemler	Büyük Sistemler Küçük Sistemler Kişisel Bilgisayarlar	Hızlı Kişisel Bilgisayarlar İş İstasyonları Veritabanı Makinaları	Paralel İşlem Optik Bellekler
KULLANICI ARABİRİMİ	Yok	Sorgulama Dilleri	Grafikler Menüler Şekil Sorgulaması	Doğal Diller Ses Girdisi
PROGRAM ARABİRİMİ	İşlemsel Diller	Gömülü Sorgulama Dilleri	4 Kuşak Dilleri Mantıksal Programlama	Bütünleşik Veritabanı ve Programlama Dilleri
GÖSTERİM VE EKRAN GÖRÜNTÜSÜ	Raporlar	Rapor Üreticileri	İşletme Grafikleri	Genelleştirilmiş Ekran Görüntü Yönetimleri
BİLGİ - İŞLEM	Verinin İşlenmesi	Bilişim ve İş Kaydı İşlemleri	İş Kaydı İşlemleri Bilginin İşlenmesi	Dağıtılmış Bilgi İşlenmesi

1960'lı yıllarda ağ yapısı ve hiyerarşik veri modelleri kullanılmaya başlanmıştır. 1970'li yıllarda önerilen ilişkisel veri modeli; biçimsel tabanlı, tek düzeli, düzenli cebirsel işlemli ve hesaba dayalı dilleri kullanma özelliğine sahip oluşu nedeniyle, büyük ilgi toplamış ve uygulamaya konulmuştur. Hemen arkasından, semantik (anlamsal) ifade gücü nedeniyle, varlık-ilişki modelinin önerilmesi sonucu, semantik hiyerarşi modeli ile semantik veri modeli geliştirilmiştir. Bu modellerin sağladığı "soyutlama" özelliğinden yararlanılarak, geliştirilmiş ilişkisel model (Relational Model/Tasmania) oluşturulmuştur.

Nesneye dayalı mantıksal modeller, kavramsal ve görünüm düzeylerinde veri tanımları için kullanılmaktadır. Bu modeller, oldukça esnek yapılı olarak veri sınırlamalarının belirtilmesine izin vermektedir. Varlık ilişki (Entity-Relationship) veri modeli, varlık adı verilen nesneler ve aralarındaki ilişkilerden oluşan gerçek dünya anlayışına dayanmaktadır.

Mantık kavramı, ilişkisel ve diğer veri modellerinin şema gösterimlerinin temelinde bulunmaktadır. İlişkisel cebir (Relational Calculus) ve ilişkisel sorgulama (Relational Query), mantık bilim dalı konularına girmektedir. Mantık; sorgulama dillerinde, bütünleşik model kurma, soru değerlendirme, sıfır değerli işlemler, eksik bilişim konularında kullanılabilen bir biçimsellik sağlamaktadır. Mantık, veritabanını tümdengelim (Deduction) anlamında bir biçimselliğe götürmektedir.

Başka bir gelişim yönü de, nesneye yönelik veri modelleridir. Nesneye yönelik yaklaşım; uzama yeteneği, yönteme bağlılık, veri tipi tanımındaki esneklik, modelleştirme gücü özellikleri nedeniyle, daha da yaygınlaşması beklenen bir veri modelidir. Ayrıca; bilgi gösterimi, programlama dilleri ve veri modellerinin birleştirilmesi de tasarlanmaktadır. **(Thuraisingham – Bhavani, 1997)**

Nesneye yönelik (Object Oriented) model, nesnelerin gösterimine dayanmaktadır. Bu model nesneye yönelik programlamaya uygun olarak geliştirilmiştir. Özellikle bilgisayar destekli tasarım, imalat ve yazılım mühendisliği (CAD, CAM, CASE) uygulamaları için yararlı bulunan yeni ve ilginç bir modeldir. **(Brown, 1991)**

Son yirmi beş yılda, bilgisayar donanımında, bütünleşik devre ve daha sonra da silikon parçacıklarının kullanımı ile çok büyük değişiklikler görülmüştür.

Bilgisayarların boyutları küçültülmüş, işlem hızı arttırılmış, uzaktan erişim ve çoklu program kullanım olanakları sağlanmıştır. Mikrobilgisayarlar yapılmış, dağıtılmış sistemler ve bilgisayar ağları kurulmuş, robot yapımında büyük ilerlemeler kaydedilmiştir. Bilgisayar donanımında görülen bu hızlı gelişme, veritabanı sistemlerine de yansımıştır. Önceleri sadece büyük sistemler için kullanılan veritabanı sistemleri, giderek küçük ve mikrobilgisayarlarda da kullanılabilir hale getirilmiştir. Güçlü donanım ile, önemli veritabanı yönetim sistemi ürünleri, iş istasyonlarında ve kişisel bilgisayarlarda kullanılmaya başlanmıştır. Oracle, DB2, Sybase gibi ilişkisel ve nesneye dayalı (Vbase) veritabanı yönetim sistemlerinin ürünleri, bu konudaki örnekler içinde yer almaktadır. Microsoft Access SQL Server ya da Paradox gibi basit veritabanı ürünleri kullanımı, pencere ve grafik yeteneği olan çok hızlı ve güçlü donanım sayesinde giderek yaygınlaşmaktadır.

Bilgisayar sistemlerinin iç bellek kapasitelerinin önemli ölçüde büyümesine ve veri saklama giderlerinin düşmesine karşın, veritabanı sisteminin genel performansı büyük bir sorun içermektedir. Geliştirilmiş veri modellerinde ve yazılımlarda sağlanan ilerlemeler, veritabanı üzerinde yer alan girdi/çıkış mimarisindeki gelişmeye oranla geride kalmaktadır. Bu durum, dış bellek araçlarına erişim hızı ile merkez işlem birimi arasında uyumsuzluk yaratmaktadır. Sonuç olarak, diskte saklı büyük hacimdeki bir verinin işlenmesi gecikmektedir. Yeni mantıksal teknoloji ve paralel işlemciler kullanılarak, işlemcilerin hızının arttırılması halinde, dış bellek araçlarının hız artış oranı yetersiz kalmaktadır. Bu sorunu çözmek amacı ile, veri yönetimi işlevleri için özel donanım mekanizmaları önerilmektedir. Bu önerilerden bazıları; veritabanı makineleri veya bilgisayarları, arka-uç işlemcileri, akıllı araçlar, çok işlemcili sistemler, birleşik bellek sistemleri ve özel amaçlı işlemciler olarak sayılabilir. Fakat, bu önerilen mekanizmalar, kayıt üzerinde veya basit prototip halinde kalmıştır. Gelecekte, veritabanı makineleri ve paralel işlem mimarilerinin teknik ve ekonomik bakımlardan yapılabilir olması beklenmektedir.

Veritabanı yönetim sisteminin ilk geliştirildiği 1960'lı yıllarda, kullanıcıların veritabanına doğrudan erişimi olanaksızdı. Gereksinimlerini sözlü olarak programcılara bildirerek, iletişim sağlayabiliyorlardı. Veritabanlarındaki gelişmeler sayesinde, kullanıcılar ancak form doldurmak yoluyla ya da işlemsel dilden toplu programlarla veritabanına erişebilir hale geldiler. 1970'li yıllarda sorgulama dilleri

kullanılmaya başlandı. Ağ yapılı (Network) veri modelleri, sadece firmaların kendileri için geliştirdikleri yüksek düzeyli, özel kullanıcı komutları içeren sorgulama dilleri ile yapılandırılmıştır.

İş istasyonlarının (Workstations) grafik çizim olanağı, veritabanı yönetim sistemleri için sözdizimsiz arabirim oluşturma açısından yararlı olmuştur. Pencere açma yetenekli, çekme (Pull-down) menülü ve resimli arabirimler geliştirilmiş ve yaygınlaştırılmıştır. Çoğu veritabanı yönetim sistemlerinde, 1980 yıllarında şekil sorgulaması (Query by Form) olanağından yararlanmaya başlanılmıştır. Böylece, kullanıcının parametre halinde verdiği sorular işlenip, yorumlanmakta ve sonuçlandırılmaktadır. Bu dönemde uygulamaya konulan bir yöntem de resimli (Icon) arabirimlerdir. Bu arabirimlerde kullanıcı, fare (Mouse), imleç (Cursor) veya ışıklı kalemle yararlanarak, sorusunu resim veya imge halinde ifade edebilmektedir.

Doğal dil arabirimi konusunda, araştırmalar sürdürülmektedir. Böylece, kullanıcının sözlü olarak veritabanı ile iletişim kurabilmesi amaçlanmaktadır. 1978 yılında, “Intellect” adı verilen bir sistem de geliştirilmiş ve piyasaya sürülmüştür. Ancak, veritabanına özel bir sözlük oluşturulması, soruların spesifikasyonlarının belirlenmesi ve maliyetin yükselmesi açısından, karşılaşılan sorunlar henüz çözülmüş değildir.

Eskiden olduğu gibi günümüzde de, veritabanı işlemlerinde kullanılan hazır programların çoğunda Cobol, Fortran, Assembler, Algol, Visual Basic, Java ve C gibi genel amaçlı diller kullanılmaktadır. 1970 yıllarında büyük uygulama programlarında, sorgulama dillerinin programlama diline gömülmesi yoluna gidilmiştir.

Son dönemde, iki ayrı konu yaygınlaşmaktadır. Birincisi, kullanıcıya uygulamalarda yüksek düzeyli spesifikasyon olanağı sağlayabilen dördüncü kuşak yazılım dilleridir. Sistem veya araç, otomatik olarak uygulama kodu üretmektedir. Örneğin, INGRES (Interactive Graphics and Retrieval System), şekilli uygulama (Application by Form) arabirimi olanağı sağlamaktadır. Böylece tasarımcı, program yazmak yerine ekranda şekilleri görüntüleyerek, bir uygulama geliştirebilmektedir. dördüncü kuşak yazılım dilleri, veri yönetimi ve rapor üretiminin yerini tutamaması nedeniyle, veritabanı

sisteminde fazla bir önem taşımamaktadır. İkincisi, geleneksel ya da tümdengelimli (Deductive) bir veritabanı yönetim sistemine (Prolog gibi) mantıksal bir programlama dili eklenmesidir. Böylece, bütünleşik bir programlama sistemi oluşturulmuş olmaktadır. **(Kalıpsız, 2001)**

Bilgisayar sisteminin orijinal çıktı araçları, başlangıçta satır yazıcı ve kart deliciden ibaretti. Çıktılar, genellikle satırlar halinde rapor ya da tekrar işlenebilecek şekilde bir dizi delinmiş kart olarak alınabilmekteydi. Sonraları, yüksek düzeyde rapor üretme komutları olan RPG dili geliştirilmiştir. Günümüzde, çoğu veritabanı yönetim sisteminde rapor üreticisi bulunmaktadır. Özel rapor tanım dilleri; kullanıcı tarafından belirlenen formata, aralığa, sayfa boyutuna, toplam düzeyine, başlıklara vb. niteliklere göre düzenlemeyi sağlamaktadır.

Rapor üreticisi, veritabanına etkileşimli erişen çok sayıda kullanıcı için rapor veya şekil üretebilmektedir. Kişisel bilgisayarların varlığı ile, kullanıcılar ekrandan çok miktarda ve çeşitli bilişimi, grafik halinde sağlayabilmektedir. Çoğu veritabanı yönetim sistemi paketleri, değişik formlarda işletme grafiklerini çıktı olarak vermektedir. Bu grafikler; doğrusal regresyon, çizgi, dairesel, eğri vb. şekillerde olabilmektedir.

Veritabanı işlemlerini görüntü (Image) teknolojisi ile birleştirerek ve sayısal dönüşürülmüş sözleri veri halinde saklayarak oluşturulan deneysel sistemler ile, resim ve sözcük üretilebilmektedir.

Veritabanı teknolojisi, dosya işlemlerinin yetersiz kalması nedeni ile geliştirilmiştir. Bu nedenle ilk uygulamalar, işletme bilgi-işlem konuları olan envanter yönetimi, bordro düzenleme, muhasebe kayıtları, sipariş işlemleri, satış raporları vb. gibi konular üzerine yapılmıştır. 1970'li yılların sonunda, değişik disiplinlerin çeşitli alanlarında da veritabanı uygulamaları başlamıştır. Özellikle, taktik ve stratejik planlara ilişkin verinin, bilişim halinde özetlenerek genelleştirilmesinde yaygın bir biçimde kullanılmaktadır. Güvenliği ve güvenilirliği sebebiyle, veritabanı sistemi birbirinden uzakta bulunan uygulayıcıların ve kullanıcıların etkileştiği bir merkez bilişim kaynağı haline gelmiştir. Böylece, geniş band ve uydu yayınlarını da içeren bir veri iletişim teknolojisi ortaya çıkmıştır. Günümüzde veritabanı sisteminden en çok yararlanılan konu, çoklu işlem (Transactions Processing)'dir. Hava yolu

şirketleri, sigorta şirketleri, bankalar, gibi büyük kuruluşlar, hızlı ve güvenilir şekilde çoklu kayıt işlemlerini ve çevrim-içi güncelleştirmeyi, veritabanı sistemleri ile gerçekleştirmektedirler.

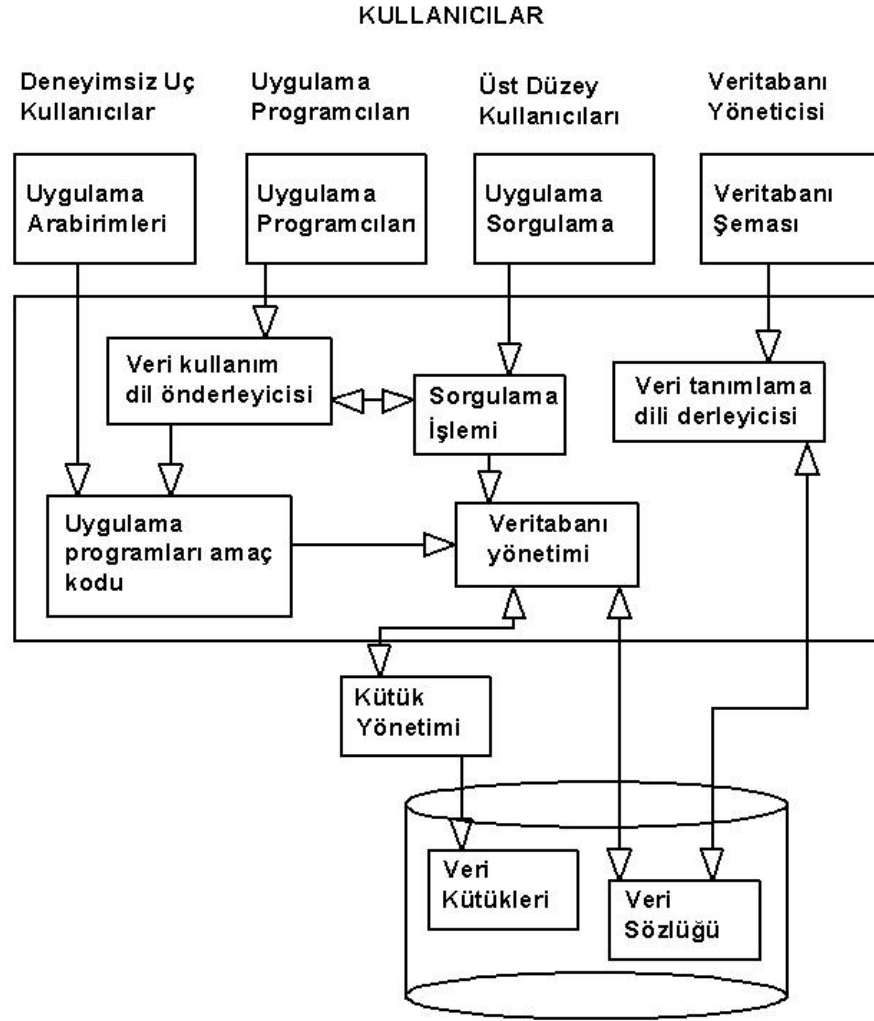
Veritabanı sisteminin diğer bir uygulama alanı da bilgi üretimidir. (Knowledge Processing) Bu uygulama alanında, veritabanında bulunmayan bir görüşü öğrenmek için, kurallar ve yorumlama mekanizması geliştirilmektedir. **(Kalıpsız, 2001)**

2.4. Veritabanı Sistemlerinin Mimari Yapısı

Veritabanı sistemi, modüllere ayrılmış olup, her biri tüm sistemin sorumluluğunu taşımaktadır. Genellikle bilgisayarın işletim sistemi, temel hizmetleri görmekte ve veritabanı sistemi bu temel üzerine kurulmaktadır. Bu durumda, bir veritabanı tasarımında, veritabanı sistemi ile işletim sistemi arasında, ara birimlerin oluşturulması da göz önünde bulundurulmalıdır.

Bir veritabanının işlevsel öğeleri kısaca şu şekilde açıklanabilir. Dosya yönetimi, disk belleğindeki yer paylaşımını ve disk üzerinde saklanan veri yapılarını yönetir. Veritabanı yönetimi, veritabanında saklı veriler, uygulama programları ve sorgular arasında arabirim işlevini görür. Sorgulama işlemcisi, sorgulama dilindeki deyimleri, veritabanı yönetiminin anlayacağı alt düzey komutlara çevirir. Veri kullanım dili ön derleyicisi, bir uygulama programı içerisine gömülü veri kullanım dili deyimlerini, asıl programlama dilinin normal işlem çağrılarına dönüştürür. Ön derleyici, uygun kodu üretebilmek için sorgulama işlemcisi ile bağlantılı olmalıdır. Veri tanım dili derleyicisi, veri tanım dili deyimlerini, değişik verilerden oluşan tablolara dönüştürür.

Fiziksel sistemin kurulması için gerekli veri yapıları ise şu şekilde açıklanabilir. Veri dosyaları, veritabanında saklanır. Veri sözlüğü, veritabanı yapısına ait değişik verileri saklar. Veri sözlüğü çok kullanıldığında tasarımına ve geliştirilmesine özen gösterilmelidir. İndisler (dizinler), özel değerler alan veri maddelerine hızlı erişim sağlar. Bu öğelerin işleyişi Şekil 2.8’de şematize edilmiştir.



Şekil 2.8 : Veritabanı Sisteminin Yapısı (Kalıpsız, 2001)

Veritabanı soyutlaması ile sistemin mimarisi; fiziksel düzey, görünüm düzey, kavramsal düzey olarak üç ayrı grupta tanımlanabilmektedir.

İç (Internal) düzey, verinin fiziksel olarak saklandığı bölümdür. Bu bölüm, verinin sabit olarak yerleştirildiği bir doğrudan erişimli dış bellekte saklanmaktadır. İç düzeye erişim, veritabanı yönetim sistemi aracılığı ile dosya yönetimi ve disk yönetimi tarafından gerçekleştirilmektedir. İç görünüm, veritabanının tamamının alt düzeyde gösterimidir. İç görünüm, sadece saklı kaydın değişik tiplerini tanımlamakla kalmayıp, hangi indislerin bulunduğu, saklı alanların nasıl gösterildiği, kayıtların hangi fiziksel sıralamada olduğu vb. belirtilmektedir.

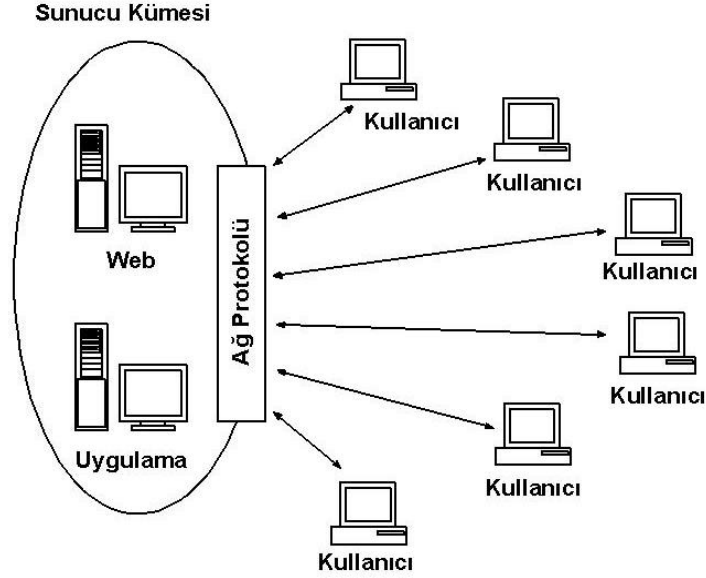
Dış (External) düzey; kullanıcılara aittir. Kullanıcı, veritabanının sadece belirli bir yönden görünümü ile ilgilenabilmektedir. Bir dış kaydın, veritabanındaki iç kaydın

aynı olması gerekmez. Dış şema, belirli bir dış görünüm içerisindeki dış kayıt tiplerinin tanımlarından oluşmaktadır.

Kavramsal (Conceptual) düzey; iç ve dış düzeylerin dolaylı olarak birleştirilmesini sağlayan bir "aracı" düzeydir. Dış düzey bireysel kullanıcı için bir "görünüm" sağladığı halde, kavramsal düzey kullanıcıların tümünü ilgilendirmektedir. Kavramsal görünüm, veritabanındaki tüm bilgileri içermektedir. Fakat fiziksel görünüme oranla büyük ölçüde soyutlanmış haldedir. Kavramsal görünüm, kavramsal şema aracılığı ile tanımlanmaktadır. Kavramsal şema sadece kavramsal görünümün bir tanımı olmayıp, ayrıca güvence denetimleri ve bütünleme denetimleri vb. işlevler için de tasarlanmaktadır.

Yazılımın veya programın işlem gücü gerektiren parçasının, sunucu (Server) olarak adlandırılan bilgisayar sisteminde yürütülmesi, sonuçların da kullanıcı (Client) olarak adlandırılan bilgisayarda gösterilmesine dayanan bir yöntemdir. Kullanıcı ve sunucu mimarisine göre tasarlanan yazılımlar iki ana kısımdan oluşur. Birincisi, kullanıcı sistem üzerinde koşan; diğeri, sunucu sistem üzerinde koşan ve tüm kullanıcılara hizmet veren kısımdır.

Bu mimaride sunucu sistem, performanslı ve güçlü bir bilgisayar sistemi üzerine kurulmalıdır. Çünkü, programın gerektirdiği işlem gücü sunucu bilgisayardan karşılanır. Sunucu sistem, hangi konuda hizmet veriyorsa, o hizmeti tüm kullanıcılara, erişim hakları ölçüsünde sunar. Kullanıcı sistem, bir arayüz gibidir. Genel olarak, kullanıcının isteklerini sunucuya gönderir ve gelen sonucu ekranında veya herhangi bir çıkış birimi aracılığıyla kullanıcıya gönderir. Kullanıcı sistemin performanslı ve güçlü olması beklenmez. Yalnızca program parçasını yürütülebilecek özellikte olması yeterlidir. Örneğin Internet, kullanıcı ve sunucu mimarisine dayalı çalışmaktadır. Internet kullanıcıları, tüm dünyaya yayılmış sunucu sistemler üzerinden hizmet alırlar. Bir Web tarayıcı, Web hizmetinin kullanıcı tarafına ait program parçasıdır. Bir de sunucu tarafında hizmet sunan daha gelişmiş bir yazılım parçası vardır. Bu mimari yapı Şekil 2.9'da gösterilmektedir.



Şekil 2.9 : Veritabanı Sisteminin Mimari Yapısı (Çölkesen, 2003)

Kullanıcı ve sunucu mimarisinin yararı, paylaşımın kolaylaştırılması ve yönetilmesinin sağladığı esnekliktir. Bu mimari yapının sağladığı avantajlar, bilgisayarların sunduğu hesaplama ve veritabanı imkanlarının paylaşılması, ortak verilerin merkezi bir yerde toplanması ve farklı işlere farklı sunucu sistemler yapılmasıyla dağınık bir mimarinin sağlanması olarak belirtilebilir.

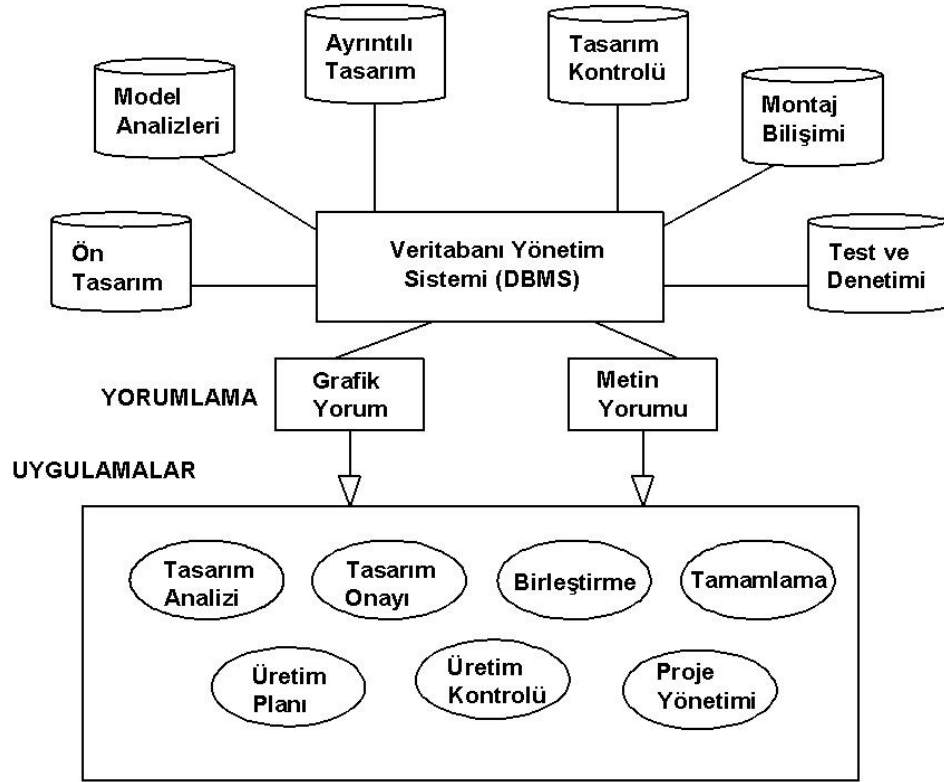
2.5. Veritabanı Sistemlerinin Yeni Kullanım Alanları

Bir işletme veya kuruluşun bilgi-işlem faaliyetleri için geliştirilen veritabanı sisteminin uygulama alanları, günümüzde giderek çoğalmaktadır. Bu alanlardan başlıcaları, mimarlık, mühendislik tasarımı ve üretimi, büro sistemleri ve karar verme sistemleri, istatistiksel ve bilimsel veritabanı yönetimi, uzman sistemler, çoklu ortam veritabanları, web ve veritabanı uygulamalarıdır.

2.5.1. Mimarlık, Mühendislik Tasarımı Ve Üretimi

Bilgisayarla bütünleşik tasarım ve üretimdeki güçlükler, tasarımın ve üretimin etkin olarak gerçekleştirilmesi için gerekli bilişimin sağlanması aşamasında karşımıza çıkmaktadır. Bu alana giren konular, CAD (Computer-Aided Design), CAM (Computer-Aided Manufacturing), CAE (Computer-Aided Engineering) ve CIM (Computer-Integrated Manufacturing) olarak sayılmaktadır.

Veritabanı teknolojisi kullanımı, ürün tahmini, sipariş işlemleri, üretim planı, üretim kontrolü, envanter kontrolü ve maliyet olarak, işletme uygulamaları, materyal gereksinimi için planlama ile birlikte, ürünün tasarımı ve mühendisliği, üretimin kaynak planlaması, uygulanması ve denetlenmesi gibi konuların birleştirilip bütünleştirilmesine yardımcı olmaktadır. Veritabanı yönetim sistemi, bir üretim sürecine ilişkin veriyi işlemekte, yorumlamakta ve bütünleşik olarak uygulayabilmektedir. Bu sistem, Şekil 2.10’da özetlenmektedir.



Şekil 2.10 : Mimarlık, Mühendislik Tasarımında Veritabanı Sistemi

Kullanımı

(Elmasri Navathe, 1989)

Tasarım ve üretim sorunları üzerine yapılan çok sayıdaki araştırma, bilgisayar destekli uygulamalar için özellikle veritabanı sisteminin oluşturulması gereğini ortaya koymuştur. Veritabanı sistemi; analitik araçların (örneğin sonlu eleman analizinin), grafik çizimlerin (özellikle üç boyutlu), simulasyonların ve optimum tasarım algoritmalarının kullanılması için gerekli altyapıyı sağlamaktadır. Mühendislik tasarım verisinin geleneksel veri modelleri ve veritabanı yönetim sistemi ile elde edilmesi ve gösterimi güçtür.

2.5.2. Büro Sistemleri ve Karar Verme Sistemleri

Bilişim sistemleri uygulama konularında en hızlı gelişme, büro işlemlerinin otomasyonunda görülmektedir. Bu konular, olaylar önceden kestirilebildiği ve yanıtları da bilindiği için, programlanabilir işlemlerden sayılmakta ve veritabanı sistemi teknolojisinden büyük ölçüde yararlanılabilmektedir. 1982 yılında, büro iletişim sistemi için IBM tarafından OBE (Office By Example) kısa adıyla bir dil ve veri modeli geliştirilmiştir. OBE dili, QBE (Query-By-Example)'nin genişletilmiş bir şeklidir. OBE modeli de; şekil, rapor, belge, adres listesi, duyuru listesi, menü vb. olarak çeşitli nesneleri temel ilişkisel tablolar halinde göstermektedir. Sistem, bu verileri resim, metin ve zaman olarak yansıtmaktadır. Büro sistemleri bilgi-işlem sistemi; veri analizi, sistem kontrolü, planlama ve karar verme, organizasyon tasarımı ve reorganizasyonu işlemlerini de desteklemektedir. Büro sistemleri veritabanı, büronun organizasyonu ve yönetimi konularında da destek verecek şekilde tasarlanmaktadır.

2.5.3. İstatistiksel ve Bilimsel Veritabanı Yönetimi

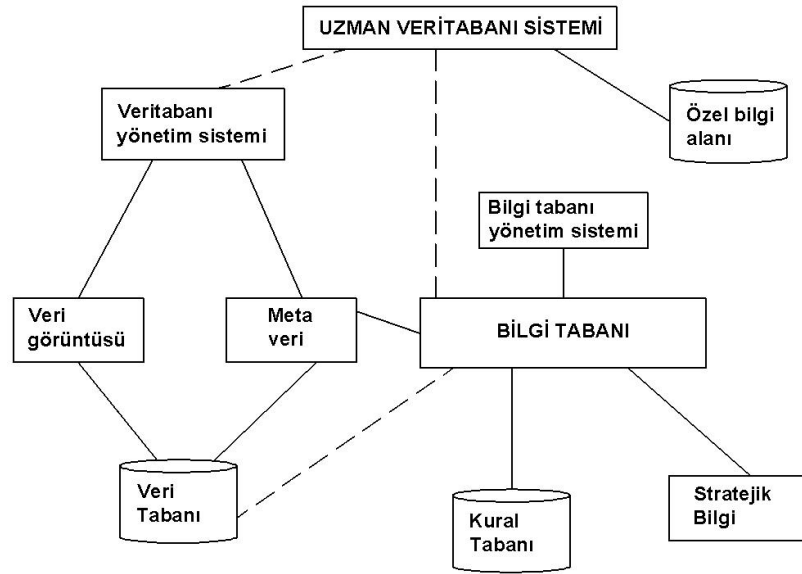
Bilgisayar sistemi, istatistiksel ve bilimsel faaliyetler için gerekli bilişimi sağlamak amacı ile kullanılmaktadır. Bu halde, "İstatistiksel ve Bilimsel Veritabanı" (Scientific And Statistical Database, SSDB) ile desteklenmektedir. Bu veritabanı; mikro ve makro olarak, iki ayrı düzeyde tasarlanmaktadır. Mikro istatistiksel ve bilimsel veritabanı; ham istatistik veri olarak, bireysel varlık ve olaylara ait kayıtları içermektedir. Makro istatistiksel ve bilimsel veritabanında ise, ham istatistik verinin işlenmesi sonucu elde edilmiş olan tablolar veya regresyon analizleri bulunmaktadır.

Veri tipleri; zaman dizileri, vektörler, matrisler vb. olarak da çok çeşitlidir. Veri işlemleri için, tablo karşılaştırmaları, örnekleme ve istatistiksel değerlendirmeler de gerekmektedir. Kullanıcı arabiriminden istatistik simgeler ve kısaltmalar, kayıtların iç yapıları ve anlamları izlenebilmelidir. Makro istatistiksel ve bilimsel veritabanının semantik modelde oluşturulması ve veri karşılaştırmasına olanak sağlaması beklenmektedir.

Bu özellikler nedeni ile, istatistiksel ve bilimsel veritabanının geleneksel veritabanı yönetim sistemleri ile yönetilmesinde güçlükler bulunmaktadır. Fakat yine de, bu amaçla tasarlanmış özel veritabanı yönetim sistemi çok azdır.

2.5.4. Uzman Sistemler

Uzman sistem (Expert System), belirli bir alana ilişkin bilgiyi ve yorum mekanizmasını kullanarak, karmaşık bir sorunu çözen bilgisayar sistemidir. Sorunun çözümü, konuya ilişkin veriye belirli kural ve ilkelerin uygulanması ile gerçekleştirilmektedir. Uzman sistem, yorum çalışmasıyla soruna çözüm üretirken, sürekli olarak bilgiye ihtiyacı olmaktadır. Bu bilgi, bilgi tabanından (Knowledge Base) sağlanmaktadır. Uzman sistemin bir veritabanı ile bütünleşik olması halinde, uzman veritabanı sistemi adı verilmektedir. Bu sistem, Şekil 2.11’de gösterilmektedir.



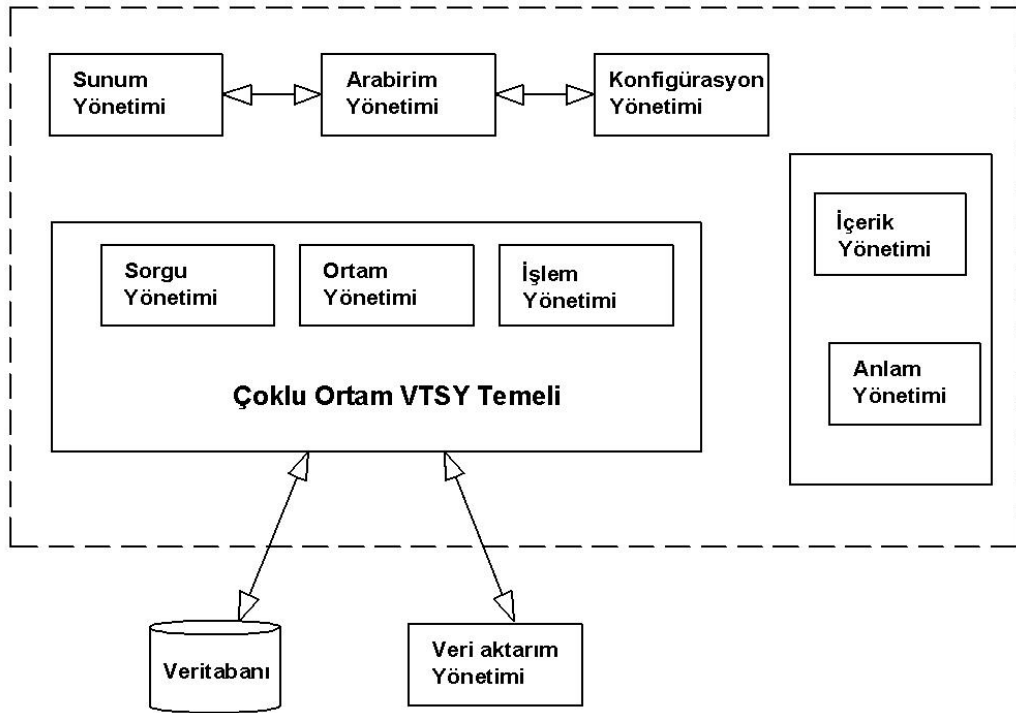
Şekil 2.11 : Bütünleşik Uzman Veritabanı Sistemi Yapısı (Elmasri Navathe, 1989)

Uzman sistemin sorun çözümü işlemini gerçekleştirebilmesi için, sorunla ilgili özel bilgiye ve sorunu yorumlayacak kural ve bilgilere ihtiyaç bulunmaktadır. Uzman sistem; belirli uzmanlık alanındaki bilgi, ilke ve işlemleri düzenleyen ve bu bilgileri sistem kullanıcısı için yararlı yorum haline getiren bir bilgisayar programıdır. Bir yorum makinası olan uzman sistemin sorunu çözüp, sonuçlandırabilmesi için, ayrıca bilgi gerekmektedir. Bu bilgi, bilgi tabanından (Knowledge Base) sağlanmaktadır. Bilgi tabanı yönetim sistemi; meta bilgi ilişkilerini anlamakta ve uzman sistemin

yorumuna sunmak üzere yeni veri üretmektedir. Uzman sistemin, veritabanı yönetim sistemi ile desteklenmesi ile, uzman veritabanı sistemi oluşturulmaktadır. Veri yönetimi ve yorumlama işlevleri tek bir program sistemi içerisinde birleştirilerek “Prolog” gibi tündengelim veritabanı sistemi oluşturulmaktadır.

2.5.5. Çoklu Ortam Veritabanları

Çoklu ortam veritabanları, metin, görüntü, ses, video ve animasyon gibi farklı veri tiplerini içerir. Farklı veri tipleri, saklama, erişim, indeksleme ve geri çağırma için özel yöntemler gerektirmektedir. Çoklu ortam verisi; yapı belirsizliği, geçicilik ve büyük hacim özelliklerine sahiptir. Çoklu ortam veritabanı yönetim sistemleri, Şekil 2.12’te gösterildiği gibi, çoklu ortam veritabanları için veri modelleme, yaratma, saklama, erişim, sorgulama, arabirim oluşturma ve kontrol sağlarlar.

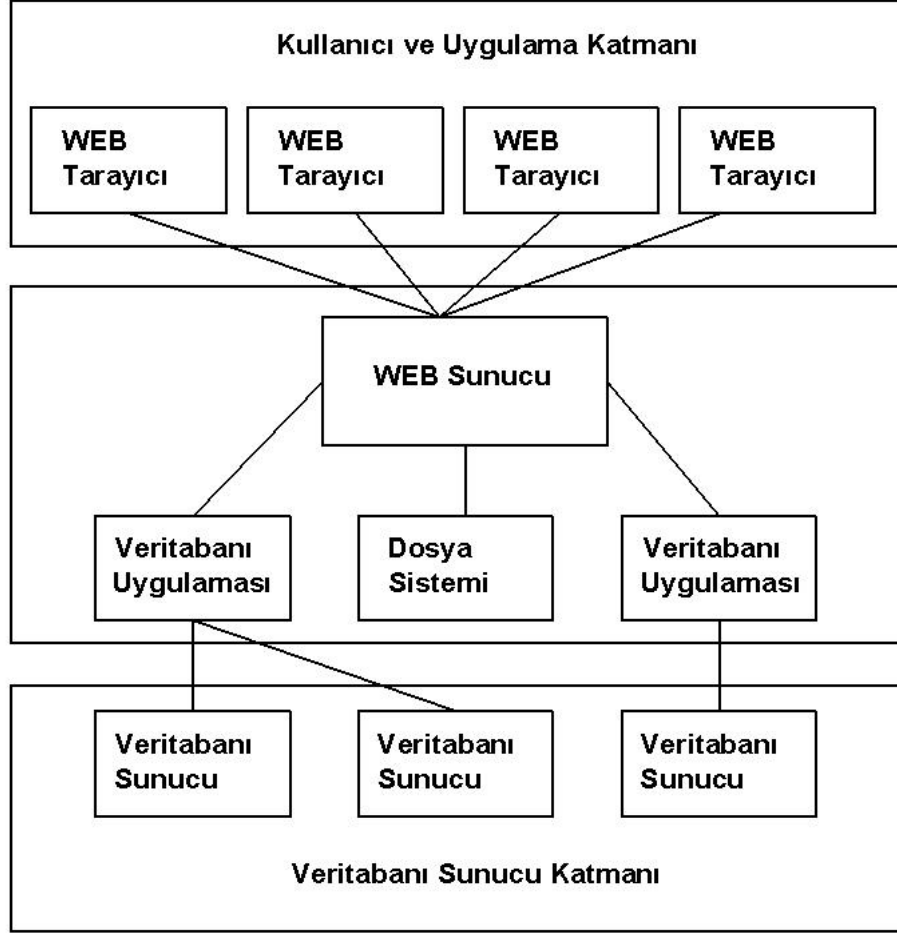


Şekil 2.12 : Çoklu Ortam Veritabanı Sistemi Yapısı (Kalıpsız, 2001)

2.5.6. "Web" ve Veritabanı Uygulamaları

Web, bir belgedeki kelimelerin veya imgelerin diğer belgelere, filmlere, ses dosyalarına veya diğer sunuculara bağlantı oluşturmaya olanak tanır. İnternet üzerindeki web hizmetinden etkili bir şekilde yararlanmak amacıyla, statik olmayan

etkileşimli web sayfaları hazırlanarak, veritabanlarından sorgulamalar gerçekleştiren ve sonuçlarını istemciye tarayıcıya (Browser) gönderen uygulamalar günümüzde kullanılmaktadır. Bu işleyiş, Şekil 2.13'te gösterilmiştir.



Şekil 2.13 : Veritabanı Tarafından Desteklenen Web Yapısı (Richard, 2001)

3. VERİTABANI SİSTEMİ TASARIM AŞAMALARI

Veritabanı bağımsız bir sistem değildir. Bütünleşik bir bilgi-işlem sisteminin bir parçasıdır. Bu nedenle, ait olduğu bilgi-işlem sistemine bağlı olarak tasarlanması gerekmektedir. Bir bilgi-işlem sistemi (veya bilişim sistemi), donanım, yazılım, insan, dosyalar, veritabanı, belgeler, yöntemler olmak üzere, altı ana öğeden oluşmaktadır. Bu öğelerin bir sistem oluşturacak şekilde kaynaştırılması gerekmektedir.

Bilgi-işlem sistemi bağımsız olmayıp, büyük bir sistem olan bir kurumun bilişim ihtiyacını karşılayan bir alt sistemdir. Bu nedenle, bilgi-işlem sisteminin tasarımı ve geliştirilmesi esnasında, ana sistemin amaç ve işlevlerinin göz önünde tutulması gerekmektedir.

Bilgi-işlem sistemi büyüklüğüne ve karmaşıklığına göre, bir kişi ya da örgüt tarafından yönetilmektedir. Ayrıca bir bilgi-işlem sisteminin tasarımı ve geliştirilmesi projenin büyüklüğüne göre, bir sistem analisti tarafından gerçekleştirilebildiği halde, büyük projelerde ancak, oluşturulan sistem analizi donanım, yazılım ve veritabanı çalışma grupları tarafından ortaklaşa olarak, birkaç yılda tamamlanabilmektedir. Bu halde, veritabanı tasarımının oluşturulması ve işletilmesini veri yöneticisi ile veritabanı yöneticisi üstlenmektedirler. Veritabanı yönetimi, gerektiğinde yönetici, mantıksal ve fiziksel tasarımcılar, veri sözlüğü uzmanı, sistem analisti, vb. görevlilerden oluşan bir organizasyon halinde yürütülmektedir.

Bir veritabanı; yukarıdan aşağıya ya da aşağıdan yukarıya doğru, iki ayrı biçimde gerçekleştirebilmektedir. Yukarıdan aşağıya (Top-Down) geliştirmede; genelden özele doğru gidilmektedir. Önce, veritabanı oluşturulacak sistemin stratejik amaçları, bu amaçlara ulaşmak için gerekli araçları ve gereksinimleri, bu bilişimi sağlayacak sistemlerin varlığı incelenmektedir. Bu incelemeye göre, büyük ölçüde soyutlanmış bir veri modeli geliştirilmektedir. Geliştirme ekibi, bu soyut veri modeli üzerinde

işleyerek, ayrıntıları tanımlamakta ve modeller kurmaktadır. Böylece giderek, orta düzey modellere ve nihayet istenilen veritabanlarına ve ilgili uygulamalara ulaşılmaktadır. Bunlardan uygun görülen biri veya birkaçı seçilerek, veritabanı ve uygulama programları geliştirilmiş olmaktadır.

Aşağıdan yukarı (Bottom-Up) geliştirmede ise tersine, belirli bir sistemi geliştirme ihtiyacı ile işe başlanmaktadır. Bilgisayara dayalı sistemlerin form ve raporlarını inceleyerek, kullanıcıların yeni raporlar, formlar, sorgulamalar vb. üzerindeki görüşlerini alarak, gereksinimler saptanmaktadır. Bu gereksinimleri karşılayacak bir bilişim sistemi geliştirilmektedir. Sistemin bir veritabanı olması halinde, gereksinim spesifikasyonları bir veri modeli geliştirmek için kullanılmaktadır. Bu modelden de veritabanı tasarımı ve tamamlayıcı işlemler gerçekleştirilmektedir. Veritabanı sistemi tamamlandıktan sonra, projenin diğer bilişim sistemlerinin geliştirilmesine başlanmaktadır.

Yukarıdan aşağıya doğru geliştirme yöntemini benimseyenler, veri modellerinin genel bir bakış açısı altında geliştirildiğini bildirmektedirler. Böylece arabirimlerinin daha iyi ve tutarlı olduğunu, daha az işlenmesine ve geliştirilmesine gerek duyulduğunu ileri sürmektedirler. Buna karşılık, aşağıdan yukarı doğru geliştirilmesini önerenler, bu yöntemin kısa sürede ve daha az hatalı sonuçlandırılabilirdiğini savunmaktadırlar. Genel olarak, varlık-ilişki modelinde yukarıdan aşağı ve semantik nesneye yönelik modelde aşağıdan yukarı yaklaşımın daha etkili olduğu kabul edilmektedir. **(Kalıpsız, 2001)**

Veritabanı sistemi tasarımı; Kavramsal tasarım, Tasarım için yazılım seçilmesi, Mantıksal tasarım, Fiziksel tasarım, Program Yazımı ve Veritabanının tamamlanması aşamaları ile gerçekleştirilmektedir. Bu aşamaların aynı sırada olması şart değildir. Hatta genellikle, bir sonraki aşamaya gelindiğinde, önceki aşamalara dönülüp, değişiklik yapma ihtiyacı hissedilmektedir.

3.1. Kavramsal Tasarım Aşaması

Kavramsal tasarım aşaması aslında bir hazırlık aşaması gibi de değerlendirilebilir. Kurulacak veritabanı sisteminin temelleri bu aşamada atılır. Gereksinimler belirlenir, veriler gözden geçirilir ve bir model oluşturulur. Kavramsal model, belirli uygulama

programcıları için görünüm (dış model) sağlamayı üstlenmemektedir. Bireysel uygulamalar, veritabanı yönetim sisteminden, veri saklayıcı donanımdan ve dış bellek araçlarındaki verinin fiziksel modelinden tamamen bağımsızdır. Sadece; çok sayıda uygulamalar (kullanıcılar) için, değişik kombinasyonlarda veri işlemekte yararlanılacak bilgi sağlamaktadır. Bu nedenle, kavramsal modelin tasarlanmasında, özellikle sisteme ait verinin ve veriler arasındaki ilişkilerin yapılandırılması üzerinde durulmaktadır. Bu sırada, veritabanının geliştirilmesindeki "tamamlama ve işletme" aşaması ile ilgilenilmemektedir.

Kavramsal modelin tasarımına, veri analizi ile başlanmaktadır. Anket veya görüşme yoluyla, kurumun her basamağında bilgi kullanan ve bilgi üreten görevlilerden, bugünkü ve gelecekteki veri gereksinimleri sorulmaktadır. Halen uygulanan işlemler ve üretilen formlar, makbuzlar, veri dosyaları ve programlar incelenmektedir. Böylece; veri öğelerinin adları ve tanımları, kaynakları, özellikleri, kullanımı, gizliliği, önemi ve değeri, ilişkileri, kullanım süreleri, kurumdaki yöneticiler ve görevliler arasındaki yazışma ve verilerin akış şemaları belirlenmektedir. Bu bilgiler analiz edilerek, verinin ait olduğu bölüm, varlığın adı, ara ve uç kullanım halleri, diyagramlar halinde gösterilmektedir.

Gereksinimlerin belirlenmesi ve veri modelinin değerlendirilmesi aşamalarına ait olan bu çalışma sonunda, normalizasyon işlemi de uygulanarak, veri yinelenmesi, veri kaybı veya veri yetersizliği halleri önlenmiş olmaktadır.

3.1.1. Gereksinimlerin Belirlenmesi ve Analizi

Veritabanı tasarım sürecinde, öncelikle kullanıcıların beklentileri detaylı olarak belirlenmek zorundadır. Tasarımın bu aşamasını, veritabanından beklenen amaçların ve işlevlerin, donanım-yazılım ve personel gereksinimlerinin, kullanılacak veri miktarı ve ilişkilerin saptanması oluşturmaktadır. Bu bilgiler, gereksinim analizi yoluyla elde edilmektedir. Bu incelemenin amacı; bir kullanıcı veri modeli veya gereksinimler veri modeli geliştirmek, veritabanını kullanacak her veritabanı uygulamasının ilişkisel öğelerini belirlemektir. İnceleme için şu yöntemlere başvurulmaktadır;

Varolan dokümanın gözden geçirilmesi yöntemiyle, uygulamalarla ilgili varolan dokümanlar analiz edilmektedir. Formlar, raporlar ve organizasyon şemaları gibi diğer dokümanlar gözden geçirilir.

Uygulama alanı ve işlem ihtiyaçlarının analizi ile bilginin güncel ve planlanmış kullanımı ele alınmaktadır. Bu aşama, sistemdeki bilgi akışı dışında, kesişim tipleri ve ortaya çıkış sıklıklarının analizini içermektedir. Bunun yanı sıra, kesişimler için girdi ve çıktı verileri tanımlanmaktadır. Bütün bunlar, farklı veritabanı oluşumları ve güncelleştirme ihtiyaçlarının miktarları ve sıklıkları hakkında önemli bilgi sağlamaktadır.

Soru sorma yöntemiyle potansiyel veritabanı kullanıcılarından, belirli soru setleri için yazılı cevaplar alınmaktadır. Sorular, kullanıcı öncelikleri ve farklı uygulamalara verdikleri önemi içermektedir. Veritabanı uygulamalarından yararlananlar, kullanıcılarıdır. Bu nedenle, veritabanı gereksinim analizinde, kullanıcıların bilgi, deney ve görüşlerine başvurulmalıdır. Özellikle; girdi, çıktılar ve işlem sınırlamaları bakımından gereksinimleri sorulmalıdır. Veritabanı geliştirme ekibi; formlar, raporlar ve menüler (seçenekler) için prototipler (örnekler) hazırlayarak, bunlar üzerinde kullanıcıların görüşleri alınmalıdır.

İhtiyaçların belirlenmesi için öngörülen bu yöntemler, zayıf yapılanmış ve daha çok formel olmayan durumlar ortaya çıkarmaktadır. Bu ihtiyaçlar, formel ihtiyaç belirleme teknikleri ile daha iyi bir yapıya dönüştürülmektedir. **(Demirdağ, 2001)**

Gereksinim analizi sonunda, veritabanı gereksinimleri belirlenmektedir. Belirleme; metin, veri akış diyagramları, varlık-ilişki diyagramları, nesne diyagramları, prototipler halinde ortaya çıkarılır.

3.1.2. Gereksinimler Veri Modelinin Oluşturulması

Gereksinimler veri modeli, kullanıcıların anlayabileceği şekilde hazırlanan veri modelidir. Bu modeldeki varlıklar ve ilişkiler, sadece kullanıcı gereksinimlerini belirlemekte olup, veritabanında yer alamazlar. Çok kullanıcıli sistemde, kullanıcı gereksinimleri uyumsuz ve çelişkili olabilir. Bu nedenle, gereksinimler veri modeli, veritabanı uzmanları tarafından yeniden yorumlanarak ve değerlendirilerek, daha

sonraki tasarım aşamalarında, veritabanı tasarımında kullanabilecek bir veri modeline dönüştürülmektedir.

Veritabanı uzmanı, kullanıcı veri modellerini (raporlar ve formlar) yorumlayarak, yeniden işleyerek ve canlandırarak, veritabanında saklanabilecek yapıya ve ilişkilere, gerçek bir veri modeline dönüştürmektedir. Veri modelinin oluşturulmasında, tasarım aşamasında yararlanabilecek standart bir dil ve diyagram kullanılmaktadır. **(Kalıpsız, 2001)**

Veri modelinin oluşturulması, çok riskli bir işlemdir. Kullanıcı gereksinimlerini tam karşılayamayacağı gibi, çeşitli hatalar yapılabilir. Hata ve eksiklerin bu aşamada bulunup düzeltilmesi, kolay ve ucuzdur. Oysa, daha sonraki aşamalarda fark edilmesi halinde, geriye dönülmesi büyük bir iş kaybına ve gidere yol açmaktadır. Bu nedenle, veri modelinin tekrar tekrar gözden geçirilmesi, bulunan hata ve eksiklerin anında giderilmesi gerekmektedir.

Veri modelini değerlendirme aşaması sonunda, yapılabilir ve maliyeti uygun gereksinimler ve geliştirilmiş bir veri modeli elde edilmektedir. Bu sonuçlar, veritabanı sistemi tasarım aşamasına girdi olarak aktarılmaktadır.

3.1.3. Normalleştirme İşlemi

Tablo, veritabanının kalbidir. Veritabanı çalışmalarının çoğu, doğrudan veya dolaylı olarak tablolarda yoğunlaşmaktadır. Veritabanı yapısında birbiriyle ilişkili birkaç tablonun bulundurulması, veritabanının işlevselliği için hazır olduğu anlamına gelmemektedir.

Normalleştirme işlemi ya da normalizasyon, bir veritabanında tekrarlanan verilerin ve onlarla ilgili başka hataların bulunması ve ortadan kaldırılması süreci olarak tanımlandığı gibi, normalizasyon, optimizasyon için tabloların daha küçük bileşenlere bölünmesi olarak da ifade edilebilir. **(Villareal, 2002)**

İyi bir veritabanı tasarımını ve doğru bir tablo yapısını garanti etmek için, iki tasarım kusurundan kaçınılması gerekir; tekrar ve tutarsızlık. Eğer aşırı sayıda alana veya çok miktarda gereksiz ve tekrarlanan veriye sahip bir tablo mevcutsa, normalizasyon

işlemi şarttır. Tablolar daha küçük tablolara bölünerek ve ilişkili tablolar arasında bir bağlantı sağlanarak, şu avantajlar elde edinilebilir;

- Hem tablonun kendi içinde, hem de ilişkili tablolar arasında tekrarlanan verilerden kaçınılması,
- Veritabanının hız ve veriminin maksimum düzeye getirilmesi,
- Aynı verinin birden fazla yere girilmesinin gerekmemesinin garanti edilmesi,
- Veri tekrarından kaçınılmasıyla disk alanından tasarruf edilmesi,
- İster girdi, ister çıktı verisi olsun, verilerin tutarlı ve bir örnek olmasının garanti edilmesi,
- Normalize edilmemiş verilere göre, muhafaza edilmesi ve kullanılması çok daha kolay bir tasarım yaratılması,
- Sorgular, formlar ve raporlarda koşullu aramalar için bir mekanizma yaratılması

Veriler (aynı tabloda veya iki ya da daha fazla tabloda) birden fazla yerde saklandığında veri tekrarı yaratılmış olur. Veri aktarımına, Güncelleme Anomalisi (Update Anomaly) adı verilir. Gereksiz tekrar anomalileri ise tutarsızlık yaratmaktadır. Veritabanı içerisinde herhangi bir yapıya ait olmayan resim numaraları olduğu varsayılırsa, tablolar bağlantılı değilse ve bir resim numarası değiştirilmişse, bu durum gerçekleşmektedir. Bu yapının resimleri, boş olarak dolaşp, bir bağlantı olmaksızın yer kaplayan kayıp verilere benzemektedir.

Başka bir tutarsızlık tipi ise, birden fazla yere veri girildiğinde oluşmaktadır. Bir kişinin ismi bir yerde doğru, ancak başka bir yerde yanlış yazılmış olabilir. Bu şekilde, normalizasyon, kullanıcı hatasını minimuma indirmektedir.

3.1.4. Sistem Kataloğu (Veri Sözlüğü) Oluşturulması

Veritabanı içerisinde bir veri sözlüğü de saklanmaktadır. Veri sözlüğü; veritabanının tasarımı, kullanımı ve yönetiminde yararlanılan çok önemli bir araçtır. Veri adları standardizasyonunu, veri, kayıt ve işlemlerin arasında bağlantı kurulmasını sağlamaktadır. Ayrıca, tablo ve veri maddeleri standart hale getirilerek, uygulama programları arasında bir uyum oluşturulabilmektedir.

Veri Sözlüğü, veritabanı kullanıcıları ve veritabanı yönetim sistemi için bir kaynak ve dayanak oluşturmaktadır. Veritabanı yöneticisi, tasarım sırasında, verinin organize edilmesinde, veritabanı kurulduktan sonra, veritabanının değiştirilmesinde ve yönetiminde veri sözlüğünden faydalanmaktadır. Veri sözlüğündeki verinin oluşturulması ve yönetilmesinde, veri tanım ve kullanım dillerinden yararlanılmaktadır.

Veri sözlüğünde her ilişki için; ilişkinin ve içerdiği özelliklerin adları, özelliklerin etki alanları ile oluşturulan görünüm adları ve tanımları, her ilişkinin anahtarı ve bütünlük sınırları saklanmaktadır. Çoğu sistem, yetkili kullanıcıların adlarını ve durumlarını belirtmektedir. Bazı veritabanı sistemlerinde bu bilgiler, özel amaçlı yapılar ve kodlar kullanılarak saklanmaktadır. Genellikle, veritabanı hakkındaki veriyi, başka bir veritabanı içerisinde saklamak yoluna gidilmektedir.

Veri sözlüğü, aktif veya pasif olmak üzere iki çeşittir. Aktif veri sözlüğü; veritabanı yapısındaki bir değişikliği, yazılım aracılığı ile otomatik olarak gerçekleştirmektedir. Pasif veri sözlüğünde ise, ayrıca güncelleştirme işlemi gerekmektedir. Daha ucuz olmasına karşın, güncelleştirilmesi çok zaman alıcıdır.

Veritabanından yararlanan kişilerin, verinin ne olduğunu ve ne anlama geldiğini bilmesi gerekmektedir. Veritabanı tasarımının ilk aşamasında, her veri maddesi doğal bir dil ile tanımlanmaktadır. Bu tanımlar, veriyi kullanacak kişilerin ortak olarak anlayabilecekleri ve kabul edecekleri bir netlikte yapılmaktadır. Bu aşamada her veri parçası yazılı olarak tanımlanıp, gerçek fiziksel gösterimi üzerinde durulmamaktadır. Tasarımcı, verinin paket halinde mi saklanacağı, yoksa karakter formu ya da başka bir format mı oluşturacağı konusu ile ilgilenmemektedir. Buna karşın, projeye başlandığı gün, veri hakkındaki bilginin bir veri sözlüğü içerisinde toplanması gerekmektedir. Bu sözlük sayesinde, tasarımcılar ve kullanıcılar birbiri ile anlaşabilmektedir.

Kavramsal modelin tasarlanmasında; varlıklara, veri öğelerine ya da varlıkları betimleyen özelliklere ve veri öğeleri arasındaki ilişkilere gereksinim duyulmaktadır. Veri öğelerinin kullanıcılar tarafından hangi sıklıkta kullanılacağını da bilinmesi gibi gereksinimler, amaçları ile birlikte, veri sözlüğüne saklanmaktadır.

Mantıksal model için veri sözlüğünde saklanması gereken bilgiler, anahtara bağlı halde veri elemanı grupları ve bunlara dayalı veri modeli, gruplar arasındaki ilişkiler, mantıksal model tarafından desteklenen dış modeller işlemler, programlar ve modüller olarak sayılabilir. Ayrıca; işlemlerin, programların ve modüllerin birbirine bağlantılarına ve aralarındaki ilişkilerine de veri sözlüğünde yer verilmelidir.

Fiziksel model için veri sözlüğünde saklanması istenen bilgiler, veri öğelerinin uzunluğu, modu, sayısal öğelerin doğruluğu, kanıtlama yöntemi, ekran görüntüsü, düzenleme kuralları, hesap işlemlerinde algoritma çıkarımları, ardışıklık, okuma ve güncelleştirmede güvence, kullanılacak araçlar, veritabanının saklanacağı birimler, erişim denetimi vb. olarak sayılabilir.

Veritabanı tasarımının başlangıcında, veritabanı yönetim yazılımı, veri sözlüğü ile etkileşim halinde işlev görmektedir. Veri sözlüğünün bütünleşik bir parçası olabilen bir rapor üreticisinin desteği ile, yönetici ve kullanıcılara ihtiyaçları olan raporlar hazırlanabilmektedir. **(Kalıpsız, 2001)**

3.2. İlişkisel Veritabanı Yönetim Sisteminin Seçimi

Veritabanı yönetim sistemi (DBMS) seçimi teknik, ekonomik ve organizasyon politikası gibi birçok faktörlere bağlıdır. DBMS seçiminde dikkat edilmesi gereken maliyetler, yazılım, bakım, donanım, veritabanı oluşturma, personel ve eğitim maliyetleri olarak sıralanmaktadır.

Yazılımın satın alınması bir maliyettir ve bu noktada dil seçimleri, farklı arayüzler, özel erişim metotları gibi faktörlerin maliyeti dikkate alınmalıdır. Bakım maliyeti ise, DBMS versiyonunun güncelleştirilmesi için standart bir bakım servisinin sağlanmasına ait maliyettir. Donanım maliyeti, ek bellek, terminaller, disk üniteleri gibi yeni donanıma ihtiyaç duyulması halinde ortaya çıkacak maliyettir.

Veritabanı oluşturma ve dönüştürme maliyeti, DBMS yazılımının yeni bir veritabanı oluşturulması veya eski veritabanının yeniye dönüştürülmesi için kullanılmasına ait maliyettir. Bu maliyetin belirlenmesi oldukça zor bir işlem olduğundan, tahmini olarak hesaplanmaktadır.

DBMS'in bir organizasyonda ilk kez kullanılması ile, veriyi yönlendiren bölümlerde bir yenilenmeye ihtiyaç duyulmaktadır. Burada yeni yapıların veritabanına girilmesi, ek bir maliyet unsuru olarak karşımıza çıkmaktadır. Bunun yanı sıra, DBMS'ler çoğunlukla karmaşık sistemler olduklarından, mevcut personelin de bu sistemlerden faydalanabilmesi için eğitilmeleri gerekir ki, bu da yine bir maliyeti artıran unsurdur. **(Demirdağ, 2001)**

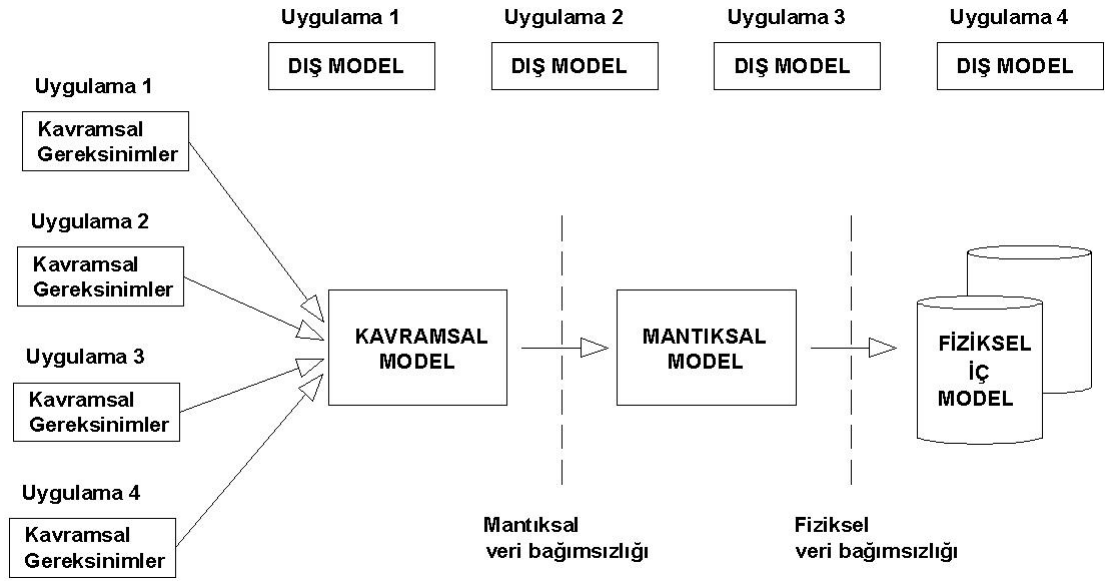
İlişkisel veritabanı yönetim sistemi; 1970 yılında geliştirilmiş olup, günümüzde en yaygın olarak kullanılan sistemdir. Bu sistemlerde, SQL veya QUEL sorgu dilleri kullanılmaktadır. Veri, tablolar halinde düzenlenmekte ve sıralar arası ilişki kavramsal düzeyde görülebilmektedir.

3.3. Mantıksal Tasarım Aşaması

Kavramsal tasarım aşamasında oluşturulan gereksinimler modeli, bugünün ve yakın geleceğin ortak gereksinimlerine dayanarak, veritabanı yönetim sisteminden, tüm yazılım ve donanımlardan bağımsız olarak, tasarlanan bir modeldir. Bu modelin veritabanı yönetim sistemine uyarlanması ile yeni bir mantıksal model oluşturulmuş olur. Mantıksal model, kavramsal modelin aksine veritabanı sistemine, veri modellerine, tüm yazılım ve donanımlara bağımlı olarak geliştirilmektedir.

Mantıksal model, kavramsal modelin seçilen veritabanı yönetim sistemi ile bağdaşacak şekilde dönüştürülmüş halidir. Kullanıcılar, mantıksal modelin alt setleri olan, dış model ya da alt şema adı verilen görünümlemler ile karşılaşmaktadırlar. Dış modeller, kullanıcının aldığı görünümlemler olup, mantıksal modele dayanmaktadır. Kavramsal gereksinimler ise kullanıcı istekleri olup, kavramsal modele temel oluşturmaktadırlar. Şekil 3.1'de bu durum şematize edilerek gösterilmektedir.

Mantıksal modelin kurulmasından önce, düzenlenmiş olan kavramsal model için en uygun veri modelinin kararlaştırılması gerekmektedir. Piyasadaki veritabanı yönetim sistemleri, ilişkisel, hiyerarşik veya ağ veri modellerini ya da bunların bileşimini kullanmaktadırlar.



Şekil 3.1 : Kavramsal, mantıksal ve fiziksel modeller arasındaki ilişki. (Kalıpsız, 2001)

3.3.1. İlişkisel Veri Modeline Uyarlama

Günümüzde en çok kullanılan mantıksal veri modeli, ilişkisel (Relational) veri modelidir. Bu model, çok sayıda ilişkilerden ve tablolardan oluşmaktadır. Kavramsal modeli, ilişkisel veri modeli halinde, mantıksal modele dönüştürmek oldukça kolaydır. Kavramsal modeldeki her kare, bir ilişki veya tablo haline getirilmektedir. Kullanıcı için de tablo formatında görünüm (Userview) oluşturulmaktadır.

3.3.2. Hiyerarşik Veri Modeline Uyarlama

Kavramsal modelin, mantıksal bir hiyerarşik modele dönüştürülmesi, doğrudan gerçekleştirilememektedir. Bazı kişisel seçimler yapılabilse bile, "doğru" sonuca ulaşıldığını söylemek olanaksızdır. Ancak, izlenecek yollar gruplandırılarak, her grupta yapılacak seçimin ilke ve kuralları belirlenebilir. Bu amaç ile beş ayrı grup oluşturulmuştur.

- Belirli bir veritabanı yönetim sistemini göz önünde bulundurmadan, bir hiyerarşik veri modeli oluşturmak.
- Kullanılacak veritabanı yönetim sisteminin, kurallarına aykırılığını oluşunun giderilmesi, veri modelini değiştirmek.
- Değiştirilmiş veri modelinin performansının yükseltilmesi amacı ile işlenmesi olanakları konusunda yapılabilecek işlemler; sadece bir alt basamağı bulunan

varlığı alt basamak ile birleştirmek veya veritabanı yönetim sisteminde her soyağacı için yeni bir veritabanı oluşturmak olarak sayılabilir. Bu külfeti azaltmak üzere, bazı soy ağaçlarını diğeri ile birleştirerek, eleme yoluna gidilebilmektedir.

- Anahtar adları basitleştirmek.

3.3.3. Ağ Veri Modeline Uyarlama

Hiyerarşik modelde olduğu gibi, bir kavramsal modelin ağ veri modeline dönüştürülmesi doğrudan gerçekleştirilememektedir. Burada işlemler beş basamak halinde yürütülmektedir.

- Belirli bir veritabanı yönetim sistemini dikkate almaksızın bir ağ veri modelinin oluşturulması. Bu yöntem, sahip-üye ilişkilerinin kurulması ve tek sahipliğin zorlanması halinde çözümü olmak üzere iki basamakta gerçekleştirilmektedir. Ağ veri modelinde, aynı setteki üyeler birden çok sahibe bağlanamamaktadır. Zorunlu halde, araya veri içermeyen bir veri kaydı eklenerek, çözüm bulunabilmektedir. Her set tipinin bir tek sahibi bulunduğu için, bu yolda bir çözüm gerekmemektedir.
- Kullanılacak veritabanı yönetim sisteminin kuralları ile uyumsuzluk halinde, veri modelinin değiştirilmesi.
- Dönüştürülmüş mantıksal modelin performansının yükseltmesi amacı ile işlenmesi. Bu konuda yapılabilecek işlemler, bazı kayıt tiplerinin birleştirilerek, set tipi sayısının azaltılması veya güvencenin arttırılması için kayıt tiplerini bölerek, set tipi sayısının çoğaltılması vb. olabilmektedir.
- Anahtar adlarının basitleştirilmesi.
- Veriler arasında gerçekte var olan fakat mantıksal modelde ortaya çıkmayan ilişkilerin eklenmesi. Örneğin yapı bileşimine erişimde yapı no ile yetinmeyip, yapı adı ve yapım tarihi de kullanılabilir. Böylece, mimarlık ve yapı arasında, her birinin kendi anahtarı olan üç ayrı set oluşturulur.

3.4. Fiziksel Tasarım Aşaması

Fiziksel veritabanı tasarımından beklenen, veritabanı içeriğinin ve ilişkilerinin fiziksel olarak düzenlenmesi için, uygun tekniklerin kararlaştırılması ve

gerçekleştirilmesidir. Bu amaçla dosyaların ve kayıtların organizasyonu kararlaştırılmakta, dosya kayıt düzenleri hazırlanmakta ve bunlara bağlı olarak yeni kayıtlar oluşturulmaktadır. Bunun için, önce bir plan yapılması, sonra da bu planın uygulanması yoluna gidilmektedir. Fiziksel model tasarlandıktan sonra, bir prototip (model) oluşturularak, gereksinimlere uygunluğunun ve performans yeterliliğinin denetlenmesi gerekmektedir.

Fiziksel veritabanı tasarımının amacı, en az giderle en uygun performansın sağlanmasıdır. Fiziksel tasarıma başlarken, mantıksal tasarımın tamamladığı, mantıksal şeması doğru ve gerçeklere uygun olduğu, kullanıcı isteklerine uygun bulunduğu varsayılmaktadır. Fiziksel tasarımcı, mantıksal tasarımı en etkin biçimde, en az giderle ve en yüksek performans ile gerçekleştirmeye çaba gösterir.

Fiziksel veritabanı tasarımı;

- Veri gösteriminin belirlenmesi,
- Erişim yöntemlerinin seçimi,
- Verinin dış belleklere atanması,
- Veritabanının yüklenmesi

olarak, dört basamakta tamamlanmaktadır.

3.4.1. Veri Gösteriminin Belirlenmesi

Mantıksal veri tasarımı sırasında şema oluşturulurken, aynı zamanda fiziksel tasarım olarak da, her veri ögesi kaydının ve dosyasının nasıl gösterileceği kararlaştırılmaktadır. Her veri ögesi için, veri tipi ve uzunluğu belirlenmelidir. Veri tipleri ve uzunluklarının belirlenmesi için gerekli bilgiler, mantıksal tasarımın başında derlenmektedir. Kayıt ve dosya büyüklüğü tahmininde, kullanılacak veritabanı yönetim sisteminin özelliği dikkate alınmaktadır

Veritabanının içeriği kesin olarak belirlendikten sonra tasarımcı, veritabanı yönetim sisteminin veri tanım dilini (DDL) kullanarak, veri gösterimini ayrıntılı olarak belgelendirmektedir. Veri tanım dilleri genellikle, veritabanı içerisindeki bütün veri öğelerini, kayıtlarını ve dosyalarını biçimsel yoldan adlandırmakta ve ayrıntılı olarak göstermektedir.

3.4.2. Eriřim Yöntemlerinin Seçimi

Eriřim yöntemleri, veritabanı yönetim sistemine baėlı olduėu için önceden belirlenmek durumundadır. Buna karřın yine de veritabanındaki her bir kayda eriřilecek yol saptanmalıdır. Veritabanına kendi anahtarı ile doğrudan eriřilebilen kayıt tipleri, başka kayıtların göstergeleri ya da indeksler aracılığı ile girilebilenlerden ayırt edilmektedir. Mantıksal tasarımcının kullandığı veritabanı yönetim sisteminde belirli erişim kısıtları söz konusudur.

Veritabanındaki her bir işlemin tanımı, giriş noktaları ve erişim yolları, mantıksal tasarım aşamasında gerçekleştirilmemiş ise bu aşamada betimlenmelidir. Bu şekilde kayıtların yerleştirme veya geri çağırma sırası belirlenmiş olmaktadır. Böylece her işlemin değişik seçenekleri için performans tahmini yapılabilmekte ve gider performans karşılaştırması değerlendirilebilmektedir.

Giriş noktaları ve erişim yöntemleri seçildikten sonra, bu seçim sonuçlarına göre veritabanı şeması gözden geçirilmekte ve gereken düzeltmeler yapılmaktadır. Genel giderleri azaltmak amacı ile, kayıt veya kütük boyutlarında da bazı değişiklikler yapılması gerekebilmektedir.

3.4.3. Verilerin Dış Belleklere Atanması

Giriş noktalarının ve erişim yöntemlerinin belirlenmesi, veritabanının fiziksel yapısının tam olarak şekillendirilmesine yetmemektedir. Erişim yöntemi ile tanımlanmış olan her kaydın ve kütüğün fiziksel araçlarda, yani dış belleklerde saklanacağı yerlerin de belirlenmesi gerekmektedir. Bu atama ile, fiziksel tasarım işlemi tamamlanmış olmaktadır. Atamada, çok kullanılan veriye öncelik tanınması ya da en büyük olasılığı sağlayan veri düzeninin bellekte bir araya getirilmesi, performansı yükseltmektedir. Bu işleme, “Kümeleme” (Clustering) adı verilmektedir.

Kümeleme, üç şekilde yapılabilmektedir. Birinci şekilde, çok sayıda özelliklerden oluşan kayıtlar bölünmekte ve özellik alt setleri bir arada saklanmaktadır. Böylece, her defa kaydın tamamı değil, sadece ilgili kısımların çağırılması yeterli olmaktadır. İkinci şekil, eşzamanlı erişilmesi gereken farklı kayıtları bir arada saklamaktır. Üçüncü şekil, bütün veri kütüklerini ya da veri kütükleri bölümlerini, performans

nitelikleri farklı olan dış belleklere dağıtmaktır. Burada amaç, veritabanının çok sık kullanılan kısımlarını daha hızlı ya da daha değerli belleklerde saklamaktır.

3.4.4. Veritabanının Yüklenmesi

Yükleme, bir veritabanı yükleme programı oluşturularak gerçekleştirilir. Çoğu veritabanı yönetim sistemlerinde veri yüklemede örneğin, özel bir kayıt sıralaması gibi, bazı sınırlamalar bulunduğu için, uzman bir kişinin denetiminde yürütülmesi gerekmektedir.

Verinin reorganizasyonu, içeriğinin, yapısının, erişim yöntemlerinin ya da dış bellek atamalarının değiştirilmesi işlemleridir. Bu değişikliğin nedenleri, yeni veri öğelerinin veya kayıt tiplerinin getirilmesi, yeni işlemlerin gerekmesi ya da bellekteki bozuklukların düzeltilmesi, veritabanını değiştirmek yoluyla işlem etkinliğinin artırılması olarak sayılabilir. Yeni veri öğeleri veya kayıt tipleri için, normal olarak mantıksal ve fiziksel düzeylerde yeniden tasarım başlatılmalıdır. Yeni işlem gereksinimleri ve performansın yükseltilmesi için, genellikle sadece yeni bir fiziksel tasarım yeterli olmaktadır. Yapılacak işlem, fiziksel tasarım aşamasının baştan aşağı tekrarlanması şeklindedir. Fakat yeni gereksinimlerin amaçları ve kısıtları dikkate alınarak, fiziksel tasarımın yeniden değerlendirilmesi de gerekmektedir.

3.4.5. Fiziksel Modelin Değerlendirilmesi

Veritabanının oluşturulmasından önce, fiziksel modelin performans özelliklerinin ölçülmesi zorunludur. Performansın tahmini ana değişkeni fiziksel model olan çok değişkenli bir analizi gerektirmektedir.

Fiziksel modelin değerlendirilmesinde, karşılaştırma ve sayısal değerlendirme olarak iki yöntem kullanılmaktadır.

Karşılaştırma (Trade-Offs), veritabanının tasarımında oluşturulan birden çok seçenek arasından en uygununun karşılaştırılmasında, her birinin olumlu ve olumsuz yanları not edilerek, bir karşılaştırma yapılmasıdır. Bu karşılaştırma, birbirine karşıt iki özellikten birinin tercih edilmesi şeklinde gerçekleştirilmektedir. Karşıt özelliklerden başlıcaları şu şekilde özetlenebilir;

Karmaşıklığa karşıt fazlalık, İlişkili varlıkların birleştirilmesinde karmaşık bir veri yapısının uygulanması, veri fazlalığını azaltmaktadır. Buna karşılık, aralarındaki bağları korumak için, yine çok sayıda veriye veya göstergeye ve ek işlemlere gerek duyulacaktır.

Genellemeye karşıt özelleştirme, uygulama programlarının özelleştirilmesi, sorgulamayı kolaylaştırmaktadır. Buna karşılık güncelleştirmede sorun yaratmaktadır

Denetime karşıt etkinlik, verinin ortak bir veritabanında birleştirilmesi denetim bakımından yeni olanaklar sağlamaktadır. Buna karşılık, veri saklama ve işlem giderlerini artırarak, maliyet etkinliğini düşürmektedir.

Esnekliğe karşıt etkinlik, veri bağımsızlığı, kullanıcılara ve veritabanının değiştirilmesine büyük bir esneklik ve tasarruf olanağı sağlamaktadır. Buna karşılık, karmaşık veri tasarımına ve karmaşık bir veritabanı yönetim sisteminin fiziksel desteğine ihtiyaç duyulmaktadır. Ayrıca, erişim ve arabirimlerinden yararlanma, makine kullanım süresini artırmaktadır.

Bellek alanına karşıt işlem süresinde, fazlalık verinin bulunması halinde, birçok fiziksel girdi/çıkıı işlemi oluşabilecektir. Fazlalık veri yok ise, bellek alanında tasarruf sağlanacak fakat daha çok işlem süresi gerekecektir. **(Weldon, 1983)**

Ancak bu karşılaştırmalarda, öncelikle veritabanının bağlı olduğu sistemin veya kurumun gereksinimleri dikkate alınmalıdır. Gereksinimleri karşılamayan bir yöntem, seçenek olarak ortaya konulmamalıdır.

Fiziksel modelin sayısal analizinde, seçenek modellerinin, beklenen dış bellek ihtiyacı ile verinin saklanması ve geri alınmasında, geçmesi beklenen süreler tahmin edilmektedir. Bu tahmin sonuçlarına göre de seçenekler karşılaştırılmaktadır.

Bellek alanının tahmini için, fiziksel veritabanındaki kayıt tiplerinin bit olarak ortalama uzunluklarına ve beklenen yinleme sayısına dayanarak hesaplanmaktadır.

Süre tahmini için, veri kayıt tiplerinin ortalama uzunluklarına ve aralarındaki ilişki düzeylerine göre, özel formüllerle giriş/çıkış (I/O) işlemi olasılık oranları hesaplanmaktadır.**(Kalıpsız, 2001)**

3.5. Veritabanı Sistemi Yazılım Geliştirme (Kodlama) Aşaması

3.5.1. Yazılım Geliştirme Süreçleri

Yazılım geliştirme süreci belirli bir gereksinimi karşılayacak veya bir problemi çözecek işlemlerin sayısal ortamlarda gerçekleştirilmesi için gerekli yazılım tasarımı aşamalarıdır. Yazılım tasarımında en önemli unsur, işin bitebileceğini, işe başlamadan önce görülebilmesidir.

Büyükçe sayılabilecek bir yazılım projesi, ancak adım adım tasarım aşamasından geçerek sonuca ulaşabilmektedir; aksi durumda, proje ya yarım kalmak ya da zaman aşımından dolayı proje sonucu olarak ortaya çıkacak ürüne ihtiyaç kalmamaktadır. Tasarım aşamasında sonu görülmeyen bir program veya yazılım projesi bitmez denilebilir. Çünkü, gereksinimler sürekli değişir, istenenlerden bazılarının yapılması çok kolay iken, çok kolay görünen bazı kısımların gerçekleştirilmesi zor veya maliyetli olabilir. Bu nedenle proje takımı, söz konusu projeden beklenenleri iyice analiz etmeli, gerçekleştirilebilirliğini öngörmeli ve zorluk çıkaracak, maliyeti çok arttıracak unsurları projeye başlanmadan önce ortaya koymalıdır.

Tablo 3.1 : Yazılımın tasarım süreci adımları. (Çölkesen, 2003)

ADIMLAR	AÇIKLAMA
Tanımlama	Kullanıcı isteklerini belirleme, gereksinimleri öğrenme süreci
Analiz	Bilişimci/programcı tarafından teknik analiz ve tanımlama süreci
Bölme	Yapılacakların, ayrık işler olarak takım elemanlarını paylaşılması
Kodlama	Program parçalarının seçilen bir dil veya yardımcı araçla kodlanması
Birleştirme	Ayrık program parçalarının bir araya getirilmesi
Sinama	Ürünün gerçek ortamda sınanması; alfa, beta testlerinin yapılması
Dokümantasyon	Kullanıcı ve teknik açıdan iki ayrı dokümantasyon hazırlanması

Orta ölçekli veya büyükçe bir yazılım projesi belirli aşamalardan geçilerek ortaya çıkarılmaktadır. Bu aşamalar adım adım olup, her bir adıma ait işlemler tamamlandıktan sonra bir diğerine geçilmektedir. Adımlar arasında geriye gidilmesi gerektiğinde, gidilen adım bütünüyle yeniden değerlendirilmeli, yapılacak bir

değişikliğin etkisi o adım için genel olarak göz önüne alınmalıdır. Tablo 3.1'de yazılım projesi ve analizinde kullanılabilecek tasarım adımları verilmektedir.

Tanımlama, kısaca, kullanıcı isteklerini ve gereksinimleri öğrenme sürecidir. Proje sonrası ortaya çıkacak ürünü kullanacak kullanıcının veya kurumun isteklerinin ortaya çıkarılması, programın sahip olacağı fonksiyonların ve davranışın tam olarak yazılı bir şekilde belirlenmesi ve taraflarca onaylanması yazılım projelerinde önemli bir süreçtir.

Analiz, ilk adımda tanımlanan yazılımın gerçekleştirilebilir olması için teknik açıdan gereksinimlerini ortaya koyma sürecidir. Yazılım veya sistem mühendisliği açısından bilgisi olan bilişimci tarafından yapılmalıdır. Proje sürecinde kullanılacak yardımcı araçlar, modelleme yöntemleri, ağ altyapısı gibi unsurlar değerlendirilir.

Bölme, analizi yapılan yazılımın her biri, bağımsız kişi veya kurumlarca yapılabilecek şekilde parçalara ayrılmasıdır. Bölme işlemi, proje takımındaki bireylerin veya destek alınacak kurumların özelliklerine göre yapılır. Belirli bir parça üzerinde uğraşanlar, genel olarak, bütünün ne iş yaptığını bilemeyebilirler. Dolayısıyla bölme işlemi sonucu ortaya çıkacak parçaların işlevi, ilgili global değişkenler ve anlamları, zaman ve bellek kısıtlaması gibi unsurlar açıkça tanımlanmalıdır. Her bir parçanın, gerçekleştiren tarafından testinin yapılabilmesi için de bir sınama bilgisi veya yöntemi verilmeli veya önerilmelidir.

Kodlama, parçaların bir programlama diliyle veya bir yazılım geliştirme ortamında kodlanması sürecidir. Her parçaya ait çalışabilir fonksiyon benzeri program parçaları elde edilir. Kodlanması gereken parçalar birbirinden bağımsız kişi veya gruplar tarafından gerçekleştirilebilir; çünkü her bir parçanın tanımı açıkça yapılmıştır. Bazı kritik durumlarda parçalardan bir veya birkaç tanesi birden çok gruba verilebilir. Böyle durumlarda bellek alanı ve yürütme zamanı kısıtlamalarına en iyi uyan seçilmektedir. Dolayısıyla kodlama yapan kişi ve gruplardan, veya ilgili parça için algoritmik ifadeyi geliştiren kişilerden bellek alanı ve yürütme zamanı bağıntılarının hesaplanması beklenir.

3.5.2. Program/Yazılım Tasarımı

Program veya daha genel kapsamda yazılım tasarımında, belirli aşamaların, daha kodlama işlemine geçilmeden önce göz önünde tutulması gerekmektedir. Yapılacak işlerin gereksinimlerinin, dipnotlarla ortaya koyulması ve sonunun önceden görülebilmesi gerekmektedir.

Program genel anlamda yazılımın bir alt kümesi olmaktadır. Hiyerarşinin en üstünde yazılım, daha altında program, onunda altında belirli kısmi bir işi yapacak algoritma gelmektedir. Daha sonra, program parçası, komut ve veri gelmektedir. En alt düzeyde bilgisayar içerisindeki işlemlerin yürütebileceği komutlar ve onların verileri gelir.

Algoritma, genel olarak, tek bir problemi çözecek davranışın, temel işleri yapan komutların veya deyimlerin adım adım ortaya koyulması olarak tanımlanmaktadır. Örneğin bir küme içerisindeki bilgilerin sıralanması, yalnızca sıralama işlemi, bir algoritmadır, benzer şekilde küme içerisinde bir elemanı arama işlemi de bir algoritmadır. Ancak, tüm bu algoritmaları içerisinde barındırıp, günlük veya iş yaşamındaki gereksinimi karşılayacak çözüm ise programdır. Örneğin, küme elemanlarının okunması, onların sıralanması, diskte saklanması, gerektiğinde bir elemanın varlığının aranması gibi işlemlerin bütünü programı oluşturmaktadır. Bir algoritmanın kısa bir parçası, örneğin bir karşılaştırma sonucu bir sayacın arttırılması program parçası iken, karşılaştırma veya sayacın arttırılması ise komut olarak tanımlanmaktadır.

Program tasarımı veya algoritma tasarımı yapılırken, en çok görsel ve şekilsel yaklaşımlardan yararlanılmaktadır. Bu amaçla standart olmuş, birçok yaklaşım şekli bulunmaktadır. Örneğin, akış şeması algoritma tasarımında çok bilinen şekilsel yaklaşım yöntemidir. Genel olarak, program tasarımı için, Problem Analiz Şeması (Problem Analysis Chart), Yapısal Şema (The Structure Chart), IPO (Input Processing Output) Şeması gibi yöntemler bulunmaktadır. Her biri, çözüme farklı açılardan bakmasına karşın, hepsinin ortak yanı çözüme sistematik yaklaşım sağlamaktır.

3.5.3. Algoritma Tasarımı

Algoritma, belirli bir işi veya görevi adım adım ortaya koymaktadır. Bu tanımdan da görüldüğü gibi, algoritma tasarımı genel olarak, tek bir işin yürütülmesi üzerine yoğunlaşmıştır. Çok özel durumlarda birkaç işi birden yürütebilecek algoritmalar da tasarlanabilir. Ancak, tasarım verimliliği ve etkinliği açısından, en iyi algoritmayı elde edebilmek için tek bir iş üzerine odaklanmaktadır. Örneğin, bir kümenin elemanlarını sıralama, sıralı küme içerisinde eleman arama, iki matrisin çarpılması, bir matrisin determinantının alınması, bir graf üzerinde en kısa yol ağacı bulunması, bir ağaç üzerinde dolaşma yapılması gibi, verinin daha az yer işgal etmesi için sıkıştırılması gibi, algoritmalar, tek bir amaca yönelik oluşturulmaktadırlar.

Bir problem çözülürken biri “algoritmik”, diğeri "heuristic" olarak adlandırılan iki yaklaşım şekli bulunmaktadır. Algoritmik yaklaşımda, çözüm için olası yöntemlerden en uygun olanı seçilmekte ve yapılması gerekenler adım adım ortaya konulmaktadır. "Heuristic" yaklaşımda ise, çözüm açıkça ortada değildir. Tasarımcının deneyimi, birikimi ve o andaki düşüncesine göre, problemi çözecek birşeylerin şekillendirmesiyle yapılmaktadır. Program tasarımcısı, algoritmik yaklaşımla çözemediği, ancak çözmek zorunda olduğu problemler için farkında olmadan "heuristic" yaklaşımla çözdüğü görülmektedir. Problemlerin algoritmik yaklaşımla çözülebilmesi için veri yapıları, veri modelleri ve bunlar üzerine geliştirilmiş algoritmaların bilinmesine bağlıdır.

Kaba kod, bir algoritmanın yarı programlama dili kuralına, yarı konuşma diline dönük olarak ortaya koyulması ve tanımlanmasıdır. Kaba kod çoğunlukla, bir veri yapısına dayandırılmadan algoritmayı genel olarak tasarlanır. Gerçek kod ise, algoritmanın herhangi bir programlama diliyle, belirli bir veri yapısı üzerinde gerçekleştirilmiş halidir. Bir algoritmanın gerçek kodu yalnızca, tasarlandığı veri yapısı üzerinde koşar. Veri yapısı değiştirildiğinde, algoritmanın gerçek kodu üzerinde oynamalar yapılmalıdır. Bir algoritma kaba kod ile verilirse, algoritmanın davranışı daha kolay anlatılabilir ve her türlü veri yapısına fazla zahmet edilmeden uyarlanabilir. Bu nedenle algoritma tasarımcıları geliştirdikleri algoritmaları kaba kod şeklinde göstermektedirler. (Çölkesen, 2003)

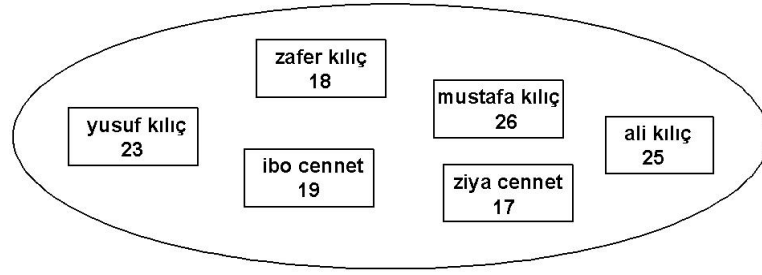
3.5.3.1. Sıralama Algoritmaları

Sıralama, sayısal ortamdaki bilgilerin veya verilerin belirli bir anahtar sözcüğe göre belirli bir anlamda sıralı erişilmesini sağlayan düzenlemedir. Bilgisayar yazılım uygulamalarında ve donanım tabanlı sayısal sistem çözümlerinde bilgilerin ve verilerin sıralı olması, bilgiye erişimi sağlayacak, bilgiyi kullanacak programlara ait algoritmaların gerçekleştirilmesini kolaylaştırır, sadeleştirir ve işlemin daha hızlı yapılmasını sağlar. Bu nedenle, ister donanım olsun, ister yazılım olsun rasgele sırada gelen ve oluşan bilgilerin sıralanması bilgisayar ve benzeri sayısal sistemlerde gereksinim duyulan önemli bir olgudur. Çok değişik sıralama algoritmaları vardır; her biri uygulamaya bağlı olarak en iyi çözüm olabilmektedir. Seçilen veri modeli, kümedeki toplam veri sayısı, bilgilerin geliş sırası gibi özellikler kullanılacak sıralama algoritmasını belirler.

Bellek üzerinde tutulan verilerin birbiriyle ilişkisini veri modeli belirlemektedir. Dolayısıyla, bir küme bilgi içerisinde belirli bir bilginin aranması veya bilgi parçalarının belirli bir anahtar sözcüğe göre sıralanması veri modeline çok bağlıdır. Bazı veri modelleri sıralamayı doğal olarak sağlar. Eğer bu tür veri modellerinden birisi seçilmişse, sıralama işlemi için ayrı bir algoritmaya gerek duyulmaz. Sadece o algoritmaların listeleme veya dolaşma algoritmaları kullanılır.

Sıralama algoritmaları, elemanları rasgele yerleştirilmiş veya dağınık şekilde bulunan bilgi kümelerini belirli bir anlamda düzenleme işi yapar. Anahtar sözcük, bazen verinin tamamı olacağı gibi belirli bir parçası da olabilmektedir. Örneğin; öğrenci bilgilerinin tutulduğu bir veri kümesinde öğrenci adlarına, soyadlarına, yaşlarına veya aldıkları not değerlerine göre sıralanabilir veya bunlardan birkaç tanesi bir arada da kullanılabilir.

Sıralama işlemi belirli bir bilgi parçasına dayanılarak yapılır. Bu, bir sayı olabileceği gibi bir sözcük olabilir veya bunların bir araya gelmesinden oluşabilir. Şekil 3.2’de görüldüğü gibi sıralama işleminde kullanılan bu bilgi parçası, genel olarak anahtar sözcük (Keyword) olarak adlandırılır. Bu anahtar sözcüğün, eğer uygulama için yaşamsal değilse, çok büyük olmaması arzu edilir. Çünkü gerekmediği halde çok uzun tutulan anahtar sözcük, hem programın çalışma zamanını etkiler hem de gerekli bellek alanı büyüklüğünü artırır.



anahtar sözcük: ad

ali kılıç 25	ibo cennet 19	mustafa kılıç 26	yusuf kılıç 23	zafer kılıç 18	ziya cennet 17
-----------------	------------------	---------------------	-------------------	-------------------	-------------------

anahtar sözcük: yaş

ziya cennet 17	zafer kılıç 18	ibo cennet 19	yusuf kılıç 23	ali kılıç 25	mustafa kılıç 26
-------------------	-------------------	------------------	-------------------	-----------------	---------------------

Şekil 3.2 : Kavramsal, mantıksal ve fiziksel modeller arasındaki ilişkiler. (Çölkesen, 2003)

Eğer sıralama işlemi disk, disket gibi saklama birimlerinde tutulan dosyalar üzerinde yapılırsa, Harici Sıralama (External Sorting) olarak adlandırılır. RAM gibi, saklama birimlerine erişimi daha hızlı, ancak sığası daha düşük olan bellek üzerinde yapılırsa Dahili Sıralama (Internal Sorting) olarak adlandırılır. Seçilecek sıralama algoritmasının hangi anlamda kullanılacağı seçimi doğrudan etkiler. Örneğin, harici anlamda işlem yapacak sıralama algoritmalarının, verilerin yerini pek değiştirmemesi istenir. Çünkü, saklama birimlerine erişim göreceli olarak daha yavaştır ve iki veri parçasının yerini değiştirmek zaman almaktadır. Dolayısıyla harici sıralama algoritmalarında en önemli unsur, daha az sayıda yer değiştirme yapmasıdır. Dahili sıralama algoritmalarında ise, bellek sığası kısıtlaması vardır. Uygulamada çok büyük boyutlu verilerin saklanması ve sıralanması için Dizinli Dosya Yapısı (Indexed File Structure) kullanılır.

Dizinli dosya yapısı, harici ve dahili sıralama algoritmalarının olumlu yanlarını bir araya getiren bir yöntemdir. Dizinli dosya yapısında veriler, geldiği sırada saklama birimindeki dosyada tutulur ve her birine bir kayıt numarası verilir. Yani veriler rasgele yerleştirilmişlerdir. Bunların sıralanması, belirli bir anahtar sözcüğe göre iç bellekte yapılır. Yani iç bellekte sıralamayı gösteren bir dizin tablosu oluşturulur. Sıralı bilgilere erişileceği sırada, dizin tablosundan bilginin kayıt numarası öğrenilir ve o numaraya dayanılarak, dosya içerisindeki bilgiye doğrudan erişilir.

Bir sıralama algoritması için önemli unsurlar yürütme zamanı ve gerekli bellek alanıdır. Yürütme zamanı doğrudan algoritma için yazılan koda ve işlemlerin hangi ortamda (iç bellek, disk, disket vs. gibi) yapıldığına bağlı iken, gerekli bellek alanını algoritma belirler.

Sıralama algoritmaları, veri modellerine göre farklılık göstermektedir, bazı algoritmalar yalnızca bir veri modelinde uygulanabilirken bazıları birkaç veri modeli üzerinde uygulanabilmektedir. Birkaç veri modelinde uygulansa bile, gerekli program kodları farklı olur.

Tablo 3.2 : Araya sokma algoritması davranışı. (Çölkesen, 2003)

Sıralanacak dizi : [7, 3, 5, 8, 2, 9, 4, 15, 6]		
Başlangıç durumu : [7] [3, 5, 8, 2, 9, 4, 15, 6]		
	İşlem Öncesi	İşlem Sonrası
1.Adım	[7, 3] [5, 8, 2, 9, 4, 15, 6]	[3, 7] [5, 8, 2, 9, 4, 15, 6]
2.Adım	[3, 7, 5] [8, 2, 9, 4, 15, 6]	[3, 5, 7] [8, 2, 9, 4, 15, 6]
3.Adım	[3, 5, 7, 8] [2, 9, 4, 15, 6]	[3, 5, 7, 8] [2, 9, 4, 15, 6]
4.Adım	[3, 5, 7, 8, 2] [9, 4, 15, 6]	[2, 3, 5, 7, 8] [9, 4, 15, 6]
5.Adım	[2, 3, 5, 7, 8, 9] [4, 15, 6]	[2, 3, 5, 7, 8, 9] [4, 15, 6]
6.Adım	[2, 3, 5, 7, 8, 9, 4] [15, 6]	[2, 3, 4, 5, 7, 8, 9] [15, 6]
7.Adım	[2, 3, 4, 5, 7, 8, 9, 15] [6]	[2, 3, 4, 5, 7, 8, 9, 15] [6]
8.Adım	[2, 3, 4, 5, 8, 9, 15, 6] []	[2, 3, 4, 5, 6, 7, 8, 9, 15] [] bitti.

- Araya Sokma Algoritması (Insertion): Sıralanacak dizinin ilk elemanını yerine bırakarak, ondan sonraki elemanları sırayla alarak, sıraya uygun olan yere sokar. İlk eleman yerinde bırakılır, ikinci eleman alındığında, dizi 2 elemanlı gibi kendi içerisinde sıralanır, 3. eleman alındığında dizi 3 elemanlı gibi kendi içerisinde sıralanır. Bu algoritmada, sıralama süreci boyunca dizi iki parça gibi düşünülür, sıralı olan ön taraf ve henüz sıralanmamış arka taraf gibi. Tablo 3.2'de algoritmanın sıralı olmayan bir dizi üzerindeki davranışı gösterilmeye çalışılmıştır, 9 elemanlı sıralı olmayan bir diziyi 8 adımda sıraladığı görülmektedir. Görüldüğü gibi dizi, sol tarafı sıralı sağ tarafı sırasız olarak iki parça gibi düşünülmektedir ve ilerleyen adımlarda sağ parçadan yeni bir eleman sol tarafa alınıp, sıralı özelliği bozulmadan eklenmekte ve

sokulmaktadır. Burada, araya sokulacak elemanın yerini bulma ve geride kalanları kaydırma işlemleri gerekir. Eğer iki işlem aynı anda yapılırsa yürütme zamanı iyileştirilmiş olunur.

- **Seçmeli Sıralama Algoritması (Selection Sort):** Bu algoritmada dizinin bir ucundan sıralamaya başlanır. Başından veya sonundan olabilir. Örneğin, dizinin başından başladığı ve küçükten büyüğe doğru sıralama yapılacağı varsayılırsa, önce ilk eleman alınır ve daha sonra küme içerisindeki en küçük eleman aranır, bulunduğu zaman ilk eleman ile yer değiştirilir, daha sonra ikinci eleman alınır ve bu eleman dahil kalan elemanlar arasında ikinci küçük eleman aranır ve ikinci elemanla yer değiştirilir. Bu işlem dizinin son elemanına kadar tekrarlanırsa, dizi sıralanmış olur. Seçmeli sıralama algoritmasının davranışı Tablo 3.3’de gösterilmektedir.

Tablo 3.3 : Seçmeli sıralama algoritması davranışı. (Çölkesen, 2003)

Sıralanacak dizi : [7, 3, 5, 8, 2, 9, 4, 15, 6]		
Başlangıç durumu : [7, 3, 5, 8, 2, 9, 4, 15, 6]		
İşlem Öncesi		İşlem Sonrası
1.Adım	[] [7, 3, 5, 8, 2, 9, 4, 15, 6]	[2] [3, 5, 8, 7, 9, 4, 15, 6] 2 ile 7'yi yer değiştir
2.Adım	[2] [3 , 5, 8, 7, 9, 4, 15, 6]	[2,3] [5, 8, 7, 9, 4, 15, 6] -
3.Adım	[2,3] [5 , 8, 7, 9, 4, 15, 6]	[2, 3, 4] [8, 7, 9, 5, 15, 6] 5 ile 4'ü yer değiştir
4.Adım	[2, 3, 4] [8 , 7, 9, 5, 15, 6]	[2, 3, 4, 5] [7, 9, 8, 15, 6] 8 ile 5'i yer değiştir
5.Adım	[2, 3, 4, 5] [7 , 9, 8, 15, 6]	[2, 3, 4, 5, 6] [9, 8, 15, 7] 7 ile 6'yı yer değiştir
6.Adım	[2, 3, 4, 5, 6] [9 , 8, 15, 7]	[2, 3, 4, 5, 6, 7] [8, 15, 9] 9 ile 7'yi yer değiştir
7.Adım	[2, 3, 4, 5, 6, 7] [8 , 15, 9]	[2, 3, 4, 5, 8, 7, 8] [15, 9] -
8.Adım	[2, 3, 4, 5, 6, 7, 8] [15 , 9]	[2, 3, 4, 5, 6, 7, 8, 9] [15] 15 ile 9'u yer değiştir
9.Adım	[2, 3, 4, 5, 6, 7, 8, 9] [15]	[2, 3, 4, 5, 6, 7, 8, 9, 15] [] bitti.

- **Kabarcık Sıralaması (Bubble Sort):** Sıralanacak elemanlar üzerinde bir yönden diğer yöne doğru ilerlenirken, komşu iki elemanın yer değiştirilmesi işlemine dayanır. Yer değiştirme işleminde, komşu iki eleman karşılaştırılır, kendi aralarında uygun yerlerde değillerse yer değiştirilir. Bu işlemler sıralama bitene kadar tekrarlanır. Elemanlar üzerindeki her bir dolaşma bir tane elemanın gerçek yerine oturmasını sağlar. Dolayısıyla, işleme, yerine oturmayan elemanlar üzerinde devam edilir. Kabarcık sıralaması iki çevrim kullanılarak gerçekleştirilir, biri elemanların üzerinde hareket edilmesi, diğeri

yerine oturmayan elemanlardan devam edilmesi için. Her çevrimde bir eleman sıralı durumdaki yerine yerleştiği için ikinci çevrim her seferinde bir eksik yapılır.

- Birleşmeli Sıralama (Merge Sort): Böl ve yönet yaklaşımına dayanır. Sıralanması istenen küme elemanları önce iki alt kümeye parçalanır ve fonksiyon kendisini sol alt küme ve sağ alt küme için iki kez çağırır. Parçalama işlemine alt kümedeki eleman sayısı bir tane olana kadar devam edilir. Alt kümelerde bir tane eleman kalmışsa, geriye doğru elemanlar çağırılır. Geri dönülürken alt kümeler elemanları sıralı olacak biçimde birleştirilir.
- Kümeleme Sıralama (Heap Sort): Bu sıralamanın temeli, ikili kümeleme ağacı kurulmasına dayanır. Sıralanacak elemanlar ile önce bir kümeleme ağacı oluşturulur, bu durumda kök en büyük değere sahiptir. Daha sonra sıralı diziyi elde etmenin iki yolu vardır. Birisi bir başka dizi gerektirmeksizin sırasız elemanların bulunduğu dizi üzerinde çalışır, diğeri sıralı elemanların tutulacağı yeni bir dizi gerektirir. Genel olarak, uygulamalarda bir başka bellek alanı gerektirmeyen çözüm kullanılmaktadır. Çünkü bilgisayar sisteminin sınırlı iç bellek alanı, fazla bir zaman maliyeti iyileştirmesi sağlamıyorsa, tasarruflu olarak kullanılmalıdır. Kümeleme sıralamasının ortalama başarımı, hızlı olanlara göre çok iyi değildir. Ancak sıralama yapılırken, ek bellek alanına gereksinim duymamasından dolayı, iç bellek üzerinde sıralama algoritması olarak seçilebilir. Sıralanacak elemanların başlangıç durumundaki sıraları algoritmanın yürütme zamanını pek fazla değiştirmez. Dizi, başlangıçta sıralı olsun veya olmasın ortalama başarımı pek sapmaz. Bazı sıralama algoritmaları için yürütme zamanı, sıralanacak elemanların halihazırda sıralı olması veya tersi yönde sıralı olmasından çok etkilenir.
- Hızlı Sıralama (Quick Sort): Böl ve yönet (Divide and Conquer) politikasına dayanır, sıralanması istenen dizi belirli bir sınır değerine göre iki alt dizeye ayrılır. İkiye ayırma işlemi, bu sınır sayısından küçük olan elemanlar bir tarafa, büyük olan elemanlar diğer tarafa toplanarak yapılır.

3.5.3.2. Arama Algoritmaları

Arama, bilgi kümesi içerisinde belirli bir anahtar sözcüğe dayanılarak onunla ilgili diğer bilgilere erişme işlemidir. Bilgisayar ve benzeri sayısal ortamlarda tutulan bilgilerin anlamlı olarak kullanılması ve onlar üzerinde işlemler yapılması için herşeyden önce bilginin olup olmadığı ve varsa belleğin neresinde olduğunun belirlenmesi gerekir. Bu işlem arama algoritmalarıyla gerçekleştirilir. Arama işleminin yapılış şekli, bilgiye ait verilerin düzenlenmesi ve bellekte tutulmasına göre farklılık göstermektedir. En yalın arama şekli, bilgiye ait verilere baştan sona kadar tek tek bakılarak yapılmasıdır. Ardışıl Arama (Sequential Search) olarak adlandırılan bu yöntemde, arama programının yazılması kolay ancak programın çalışma hızı düşük olmaktadır. Ortalama hızı arttıracak birçok yöntem kullanılsa da, uygulamalarda çoğunlukla, İkili Arama olarak adlandırılan algoritma tercih edilmektedir. En hızlı arama algoritması, Çırpı Fonksiyonu (Hash Function) kullanılarak gerçekleştirilmektedir. Ancak çatışmaların olmadığı ideal bir çırpı fonksiyonu bulunmasını gerektirir.

Çok çeşitli arama algoritması vardır. Bellek, Zaman ve Okunurluk açısından en uygun seçim yapılmaktadır. Hangisinin en uygun olduğunu, uygulamada seçilen veri modeli ve o uygulamadaki verinin boyutu belirlemektedir. Eğer küme içerisindeki eleman sayısı 50, 100, 900 gibi onlar, yüzler mertebesinde ise, arama işleminde seçilecek algoritmanın ne olduğu o kadar önemli değildir. Kodlama açısından en kolay olanı seçilebilir. Ancak, eleman sayısı onbinler, yüzbinler veya milyonlar mertebesinde ise, seçilecek arama algoritması oldukça önem kazanmaktadır.

Arama işleminin hangi ortamda yapılacağı, seçilecek algoritmayı belirleyebilir. Bellek üzerinde işlem yapmak için gerekli unsurlarla disk, yedekleme birimi üzerinde işlem yapacak gerekli unsurlar farklı olmaktadır.

Arama işlemi belirli bir veya birkaç sözcüğe dayanılarak gerçekleştirilir. Bu sözcükler Anahtar Sözcük (Keyword) olarak adlandırılmaktadır. Eğer veri, anahtar sözcüğe göre önceden sıralanmış durumda ve dizi üzerinde tutuluyorsa, arama işlemi oldukça hızlı yapılabilir, bu durumda İkili Arama (Binary Search) denilen yöntem kullanılır.

Arama işlemi bellek üzerinde tutulan veriler üzerinde yapılıyorsa Dahili Arama (Internal Search) disk ve CD gibi saklama birimleri üzerinde yapılıyorsa, Harici Arama (Extetal Search) olarak adlandırılır. Dahili aramada, arama işlemi bellek üzerinde tutulan veriler üzerinde yapılır. Veriye erişim hızlı olduğu için, verilerin yer değiştirilmesi, araya ekleme, aradan çıkartma gibi işlemler daha hızlı gerçekleştirilir. Harici aramada ise arama işlemi disk, yedekleme birimi gibi belleğe göre daha yavaş olan saklama birimleri üzerinde yapılır. Dolayısıyla verilerin yer değiştirilmesi, bir veriden diğerine atlanması, arkada kalanların öne doğru kaydırılarak bir verinin silinmesi veya arkaya doğru kaydırılarak yeni bir veriye yer açılması gibi işlemler zaman alır.

Arama işleminde kullanılan anahtar sözcükler farklı anlamlarda olabilmektedirler. Dolayısıyla değişik durumlar ortaya koyarlar. Eğer anahtar sözcüğün aldığı değer kayıt için farklı ise, bu Birincil Anahtar Sözcük (Primary Keyword), farklı kayıtlar aynı anahtar sözcüğe sahip olabiliyorsa İkincil Anahtar Sözcük (Secondary Keyword) olarak adlandırılır. Birincil anahtar sözcüğe göre arama yapılırsa, yalnızca bir tane sonuç bulunabilir. Dolayısıyla böyle bir arama algoritması tarafımızca tasarlanıyorsa, aranan bulununca geride kalan kayıtlara bakılmaksızın arama işlemi sonlandırılır. İkincil anahtar sözcüğe göre arama yapılırsa, aranan sözcüğe uygun birden çok sonuç elde edilebilir. Dolayısıyla ikincil anahtar sözcüğe göre arama yapan algoritmalar var olan kayıtların hepsini sınamış olmalıdır. Örneğin, bir öğrenci otomasyon sisteminde öğrenci numaraları birincil, öğrenci adları ikincil anahtar sözcük olabilir. Çünkü, öğrenciler farklı numaralara sahip olurken aynı isme sahip olabilir. Eğer birincil anahtar sözcüğe göre arama yapılırsa bir tane sonuç olur, ikincil anahtar sözcüğe göre yapılırsa birden çok sonuç olabilir. Birincil anahtar sözcüğün bir parçası üzerinde arama yapılırsa ikincilde olduğu gibi birden çok sonuç çıkabilir. Ancak bunların belirli parçaları aynı olur, bu tür anahtar sözcüklerin kullanıldığı arama şekline, kısmi anahtar sözcüklü arama denilir.

Arama işlemi için çeşitli şekillerde düzenlemeler yapılabilir. Veritabanı uygulamalarında en çok kullanılanı dizinli (Index) çalışma şeklidir. Bu yöntemde veriler ve kayıtlar geldiği sırada saklama birimine kaydedilir ve ayrı bir dizin tablosu oluşturularak arama işlemi bu dizin tablosu üzerinden yapılır. Dizin tablosu anahtar sözcükleri ve ilgili bilginin tamamının bulunduğu yeri gösteren işaretçi bilgilerini

içerir. Dizin tablosu, saklama biriminde veya bellekte tutulabilir. Ancak genel olarak üzerinde işlem yapılmadığı zaman dilimlerinde, saklama birimini üzerinde işlem yapıldığı anlarda bellekte tutulur. Eğer dizin tablosu üzerinde çok fazla işlem gerçekleştiriliyorsa, yani çok sık arama işlemi yapılıyorsa, her zaman bellekte tutulur. En son program sonlandırılmadan önce, kalıcı olması amacıyla saklama biriminde saklanır.

Arama işlemi, uygulamada birçok gereksinimden dolayı sık sık kullanılır. Bu gereksinimler, bilginin bir kısmından gidilerek, bütünü bulunması, bilginin bir küme veri içerisinde olup olmadığının belirlenmesi, ekleme yapılırken daha önce olup olmadığının araştırılması, sıralı bilgilerin özelliği bozulmadan ekleme yapılacak yerin bulunması, silme yapılırken varlığının araştırılması ve yerinin belirlenmesi ve bir metin içerisinde bir sözcüğün aranması şeklinde söylenebilir. Bu gereksinimler, tasarlanacak arama algoritmasının özelliklerini ortaya koymaktadır. Eğer bilginin sadece varlığı araştırılıyorsa veya belirli bir anahtar sözcükten yola çıkılarak bütününe erişilmek isteniyorsa, gerekli proses miktarı daha azdır denilebilir. Ancak ekleme amaçlı arama yapılıyorsa ve bilgiler sıralı özellikte ise, diğer bilgileri yerinden oynatmak gerekeceğinden gerekli proses miktarı yüksek olur. Dolayısıyla harici aramalarda saklama birimi üzerinde fazla miktarda proses gerektiren algoritmalarından kaçınılır.

Çok çeşitli arama algoritması tasarlanmıştır, bazıları birbirlerinin türevleri veya biraz değiştirilmiş şekilleri olsa da, 3 temel sınıf altına toplanabilir. Doğrusal veya Ardışıl Arama, İkili Arama ve Bir Çırpıda Arama.

- Ardışıl (Doğrusal) Arama, diğer bir deyişle doğrusal arama en yalın arama algoritmasıdır. Yapılan iş, ilk kayıttan başlanıp en son kayıta kadar tek tek bakmaktır. Eğer arama işlemi birincil anahtar sözcüğe dayanılarak yapılıyorsa, aranan bulununca işlem sonlandırılır. İkincile göre yapılıyorsa son kayıta kadar gidilir. Ardışıl arama algoritması kayıt sayısı onbinler, yüzbinler olan ve sık sık arama yapılan uygulamalar da seçilmemelidir. Arama işleminin seyrek yapıldığı, kayıt sayısının küçük ölçekli olduğu veya anahtar sözcükler dışında verilere dayanılarak arama yapılacağı durumlarda kullanılabilir. İkincil anahtar sözcüğe dayanılarak yapılan arama işlemlerinde,

ki uygulamada bunun daha sıklıkla kullanıldığı görülmektedir, ardışıl arama algoritması kullanılabilir.

Ardışıl aramanın çalışma hızı ortalama olarak azaltılabilir. Bunun için hemen akla gelen yöntemlerden birisi, sıklıkta yöreselliğe dayanılarak en son aranıp, bulunanları bilgi kümesinin başına taşımaktır. Böylece yakın zamanda tekrar aranması olası kayıtlar başa getirilmiş olur ve bir sonraki aramada hızlı bir şekilde erişilir. Bu şekilde bir yaklaşımla ortalama arama zamanı azaltılmış olur. Ancak arananın başa getirilmesi ek işlem maliyeti getirir. Eğer bağlantılı liste kullanılıyorsa, bu ek işlem maliyeti fazla olmaz, kayıt yerinden koparılır ve başa eklenir. Ancak kayıt dizi veya disk üzerindeyse, bu ek işlem ciddi bir işlem maliyetin doğmasına neden olur.

- İkili Arama (Binary Search), sıralı veriler üzerinde çalışan bir algoritmadır. Verilerin önceden, belirli anahtar sözcüğe göre sıralı olması gerekir. Eğer veriler bir dizi üzerinde ard arda sıralı bir şekilde tutuluyorsa veya ikili ağaç veri modeli kullanılıyorsa ikili arama yapılabilir. İkili ağaç üzerinde ikili arama ikili ağaç veri modelinin doğal yöntemi olup en etkin yöntemlerden biridir. Dolayısıyla birçok uygulamada kendisine yer bulmaktadır. 1 milyon tane kaydın bulunduğu bir veri kümesinde arama işlemi yaklaşık 20 çevrimde bulunabilir. Aynı işlem ardışıl arama ile yapılmış olsaydı, en kötü durumda 1 milyon çevrim gerekirdi.

İkili arama bir çeşit köşeye sıkıştırma yöntemidir. Aranan, belirli bir düzende sorgulama yapılarak köşeye sıkıştırılır. Örneğin, veriler dizi üzerinde sıralı olarak tutuluyorsa, ilk önce dizinin ortasına bakılır. Aranan ortadaki eleman ise aranan bulundu demektir. Hayır ise, aranan ya aşağı ya yukarı kısımdadır ya da yoktur. Böylece daha ilk adımda bile arananın, eğer varsa dizinin aşağısında veya yukarısında olduğu belirlenmiş olur. Benzer işlem ilgili parça üzerinde de tekrarlanırsa, aranana gittikçe yaklaşmış olunur. İkili arama yöntemi sıralı olsa bile, bağlantılı listeler üzerinde uygulanamaz. Çünkü bağlantılı listelerde herhangi bir düğüme, ilk düğüm hariç doğrudan erişilmez. Herhangi bir düğüme ulaşmak için, o düğüme kadar olan tüm düğümlerin üzerinden geçilmesi gerekir. Bu da, zaman karmaşıklığını artırır.

İkili ağaç veri modeli, doğrudan ikili arama yapılmasını sağlamaktadır. Çünkü, model yapısı gereği, veriler köşeye sıkışacak bir düzende tutulmaktadır. Arama için ilk önce ağacın köküne bakılır. Eğer aranan, köke eşitse bulunmuştur. Eşit değilse, bu sefer aranan ile kök değeri karşılaştırılır veya sol alt ağaca ya da sağ alt ağaca dallanılır.

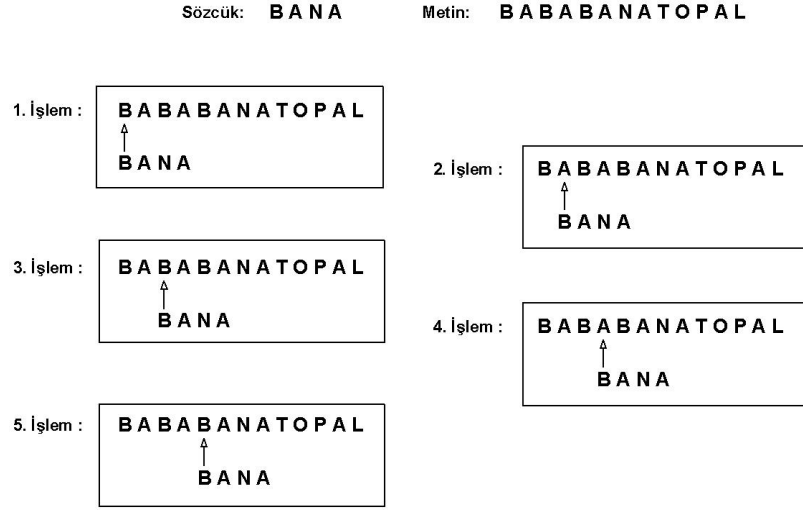
- Çırpı Fonksiyonu (Hash Function) arama işlemini bir çırpıda yapmak ve aranana doğrudan ulaşmak için kullanılan bir fonksiyondur. Fonksiyona arama işleminde kullanılacak anahtar sözcük değeri girilir ve karşılığında bir tamsayı alınır. Bu tamsayı indis gibi kullanılarak, dizi düzeninde tutulan veriye erişmek için kullanılır. Veriler üzerinde arama yapmak için en hızlı yöntem çırpı fonksiyonu kullanılmasıdır denilebilir. İdealde arama işlemi bir çırpıda yapıldığı için aramada sonuca bir anda ulaşılır. Ancak uygulamada, her zaman için ideal çırpı fonksiyonunu bulmak veya üretmek mümkün olmadığı için ideal durumdan sapmalar olmaktadır.

Çırpı fonksiyonu idealde, farklı anahtar sözcük değerleri için farklı tamsayı değerleri üretmelidir. Aynı değeri üretirse, Çatışma (Collision) oluşur. Bu durumda eğer çatışma oluşturmeyen bir çırpı fonksiyonu bulunamazsa, işin çatışmalarla beraber uygulanması gerekir. Bu işleme Çatışmanın Çözümlemesi (Collision Resolution) denilir.

- Metin içerisinde sözcük arama veya desen tanıma işlemleri de birer arama problemidir. Ancak önceki ayrıklarda verilen arama algoritmalarında farklı bir durum vardır. Metin içerisinde sözcük arama ve geçtiği yerlerin belirlenmesi, bütünüyle bir karakter dizisi içerisinde sözcüğün tüm olası durumlar göz önüne alınarak varlığının araştırılmasıdır. Önceki arama algoritmaları, daha çok verinin bir parçası olan anahtar sözcük üzerinden arama yapılması şeklindedir.

Metin içerisinde sözcük arama için en klasik yöntem, metnin ilk karakterinden başlanarak son karakterine kadar aranan sözcüğün ilk karakterin aranması ve bulunduğu zaman ardından gelen karakterin uyuşup uyuşmadığının sınanmasıdır. Eğer uyuşuyor ise bulunmuştur, uyuşmuyor ise

metin üzerindeki bir sonraki karaktere geçilir. Şekil 3.3’de bu işlem gösterilmektedir.



Şekil 3.3 : Metin içerisinde sözcük arama algoritmasının işleyişi. (Çölkesen, 2003)

3.5.3.3. Graf Algoritmaları

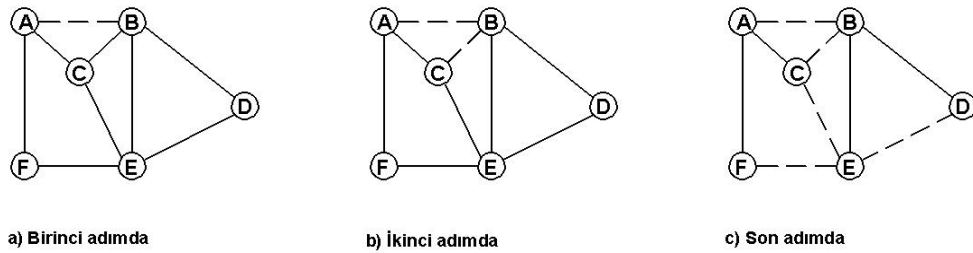
Graflar temel bilimlerde, mühendislik dallarında ve sosyal bilimlerde birçok probleme algoritmik çözüm olanağı sunar. Bir sistemin davranışı veya bir problemin tanımı, düğümler ve düğümler arasındaki ilişkiler, çizgilerle yapılabiliyorsa her türlü problemin çözümü graflara yaklaştırılabilir. Dolayısıyla graflar üzerine daha önceden yapılmış tanımlar, teoremler ve algoritmalar söz konusu problemin çözümünde yardımcı birer araç olurlar. Örneğin, karayolu haritası üzerinde iki şehir arasındaki olası en kısa yolun bulunması, bir şebeke üzerinde en yüksek akışın sağlanması, harita üzerindeki tüm şehirlere elektrik taşınması için en az maliyetin belirlenmesi, eşleşme probleminde en uygun eşleşmenin veya oturumun yapılabilmesi gibi sayılamayacak problemin çözümü graf algoritmalarına dayandırılarak, algoritmik çözümler üretilir.

Graf üzerinde dolaşma yöntemleri ve Greedy yaklaşımı, graf algoritmaları üzerinde önemli bir yer tutar. Hem temel işlemleri yürütürler, hem de diğer algoritmaların anlaşılması için teorik ve davranışsal model oluştururlar. En kısa yol problemi, düğümler arası yollardan en kısa olanın bulunması işini yürütürken, yol ağacı algoritması tüm düğümleri kapsayan yolun belirlenmesi işini yürütmektedir.

Graf üzerinde dolaşma, grafin düğümleri ve kenarları üzerinde istenen bir işi yapacak veya bir problemi çözecek biçimde hareket etmektir. Tüm düğümleri en az maliyetle dolaşma, iki düğüm arasında en kısa yolu bulma, tüm düğümler arasında en kısa yollar bulma, iki düğüm arasında en yüksek akış kapasitesini hesaplama, düğümler arasında n uzunlukta yol olup olmadığını belirleme, dolaşı yapma, gezi yapma ve buna benzer birçok işlem graf üzerinde dolaşma konusuna girer.

Mühendislikte, temel bilimlerde ve sosyal bilimlerde, birçok problemin çözümü graf veri modeline yaklaşır ve çözümleri, çoğunlukla graf üzerinde dolaşma algoritmalarıyla sağlanır. Dolayısıyla graf üzerinde dolaşma algoritmaları bilgisayar biliminde önemli bir yer tutar. Graf üzerinde dolaşma yapan birçok yaklaşım yöntemi vardır. Önemli olan iki tanesi, kısaca, DFS (Depth First Search) ve BFS (Breadth First Search) olarak adlandırılmıştır.

- DFS (Depth-First Search), graf üzerinde dolaşma yöntemlerinden birisidir. Önce derinlik araması olarak adlandırılabilir. Graf üzerinde dolaşmaya, başlangıç düğümünün bir ayrıtından başlayıp, o ayrıt üzerinden gidilebilecek en uzak (derin) düğüme kadar sürdürür. Ara düğümlerde de benzer davranışta bulunur. O düğümün bir ayrıtından gidilebilecek en uzak düğüme kadar gidilir, gidilemeyen düğüm varsa o düğümün bir başka ayrıtından denenir.

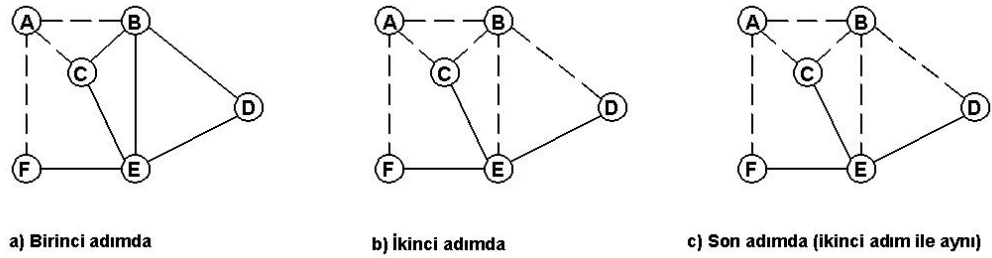


Şekil 3.4 : DFS yönteminin örnek graf üzerinde açıklanması. (Çölkesen, 2003)

Şekil 3.4.'de DFS yönteminin çalışma şekli gösterilmiştir. Dolaşmaya A'dan başlandığı varsayılırsa, önce A ile bağlantısı olan B'ye gidilir, oradan da B ile bağlantısı olan C ve sonra sırasıyla E ve D'ye ulaşılır. Bir düğümden kendisine bağlı birçok düğüm arasında önce birine gidilir, oradan da bağlı olan düğümlerden birine gidilir. Ancak bu şekilde ilerlerken gidilemeyen düğümler varsa bir geriye dönülür ve o düğümden sıradaki bir sonraki düğüme gidilir. Buradaki sıralama, düğümlerin herhangi bir düzende

sıralanmasına dayanır. Verilen örnekte düğüm adları, karakterlerle belirtildiği için düğümler A, B, C şeklinde sıralanmıştır. Bu nedenle bir düğümden bir sonrakine gidilirken, önce bu sıraya göre gidilmemiş olan düğüme gidilir.

- Graf üzerinde, diğer bir dolaşma yöntemi olan BFS (Breadth First Search), önce genişlik araması olarak adlandırılabilir. Bu yöntemin DFS'den en önemli farkı, dolaşmaya, başlangıç düğümünün bir ayrıtı üzerinden en uzağa gidilmesiyle değil, başlangıç düğümünden gidilebilecek tüm komşu düğümlere gidilmesiyle başlanır. Ara düğümlerde de başlangıç düğümü gibi davranılır. Bir ara düğüme gelindiğinde, o düğümden komşu ve daha önce ziyaret edilmemiş olan tüm düğümlere gidilir. Bu yöntemin örnek bir graf üzerinde açıklaması Şekil 3.5'de verilmiştir. Şekilden görüleceği gibi, önce A düğümüne komşu tüm düğümlere gidilir. Daha sonra, bir sonraki düğüme geçilir ve oradan da kendisine komşu olan ve daha önce gidilmemiş olan düğümlere gidilir.

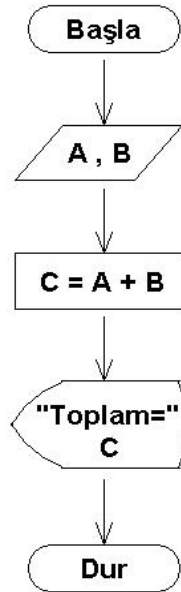


Şekil 3.5 : BFS yönteminin örnek graf üzerinde açıklanması. (Çölkesen, 2003)

- Greedy yaklaşımı, graf üzerinde belirli bir konuda optimum sonucu veya en iyi sonucu bulabilmek amacıyla dolaşma yapılırken, bir sonraki düğümü belirlemek için kullanılan bir karar verme ve seçme yöntemidir. Greedy yaklaşımında, o andaki seçenekler içerisinde en iyi olarak görüneni seçilir. Kriteri bölgesel veya yerel değerlendirmelere göre yapılmakta olup, seçilenin global olarak tüm sistem için en iyi seçim olacağı öngörülür. Greedy yaklaşımı graf üzerine geliştirilmiş olan birçok problemin çözümünde, bir sonraki düğümün belirlenmesinde, seçme unsuru olarak kullanılmaktadır. Greedy yaklaşımında anahtar sözcük bir sonraki adıma, bir sonraki düğüme geçiş için yerel karar verilmesidir. Yani o anda bulunan düğümden görüldüğü kadar en iyisi seçilir.

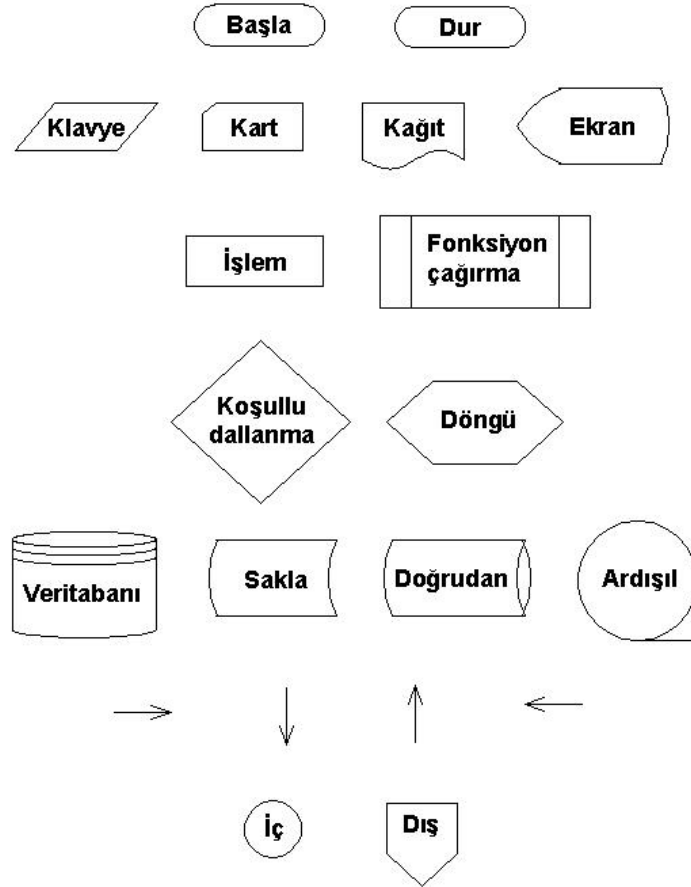
3.5.4. Akış Şemaları

Akış Şeması (Flow Charts), algoritmanın görsel ya da şekilsel olarak ortaya koyulmasıdır. Problemin çözümü için yapılması gerekenler, başından sonuna kadar, geometrik şekillerden oluşan simgelerle gösterilmektedir. Her simge, genel olarak, yapılacak bir işi veya komutu göstermektedir. Şekil 3.6' de çok basit bir akış şeması örneği gösterilmektedir. Akış şemaları, özellikle görsel öğrenme veya anlamaya yatkın program tasarımcıları için veya algoritmik düşünceyi tam oturtamamış veya algoritmik düşünce tarzına geçiş yapmamış tasarımcılar için iyi bir yöntemdir. Program tasarımcılarının deneyim kazandıkça, daha çok kaba kod ile tasarım yaptıkları görülmektedir. Ancak akış şemasının beyninde canlandıramayan programcılarında, doğrudan kaba kod tasarımı yapmaları oldukça güçtür.



Şekil 3.6 : Akış şeması örneği

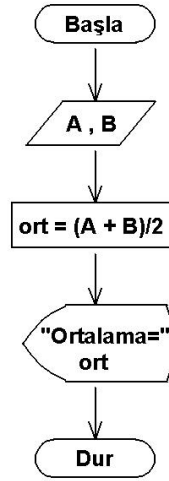
Akış şeması, algoritmanın metinsel değil de, aşağıda Şekil 3.7'de verilen geometrik simgelerle şekilsel olarak ifade edilmesidir. Akış şeması, kodlanacak dilin kontrol deyimleri göz önüne alınarak çizilirse, daha sonra kodlama kolayca gerçekleştirilebilir.



Şekil 3.7 : Akış şeması simgeleri (Çölkesen, 2003)

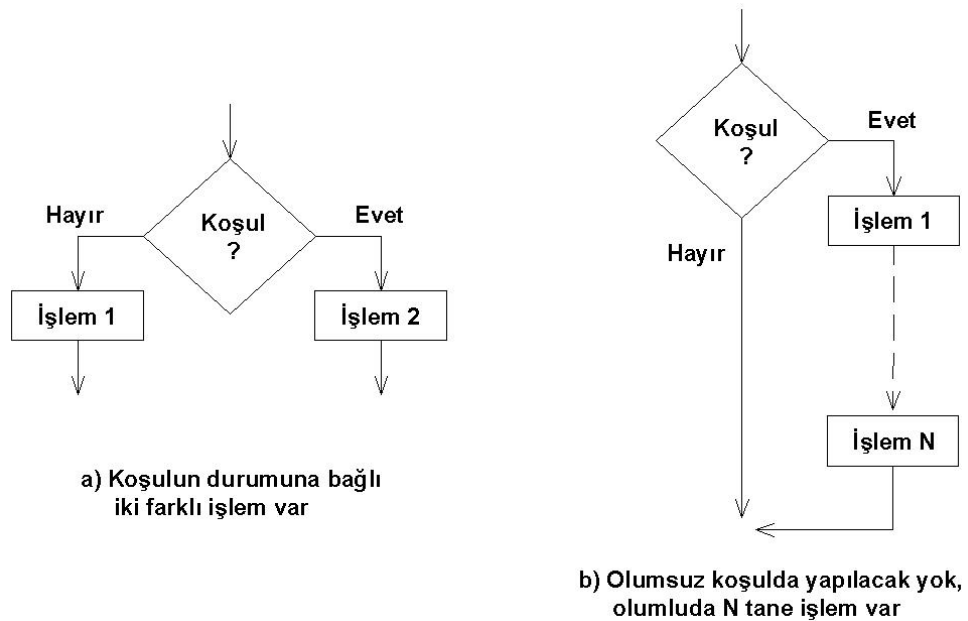
Yukarıdaki şekilde, en üstte verilen oval simgeler başlama ve bitiş için kullanılır ve ne amaçla kullanıldığı içerisine yazılır. Hemen altında dört tane G/Ç (Giriş / Çıkış) simgesi verilmiştir. Kullanılırken içerisinde değişken adı veya mesaj yazılır. Üçüncü sırada, işlem ve fonksiyon çağırma simgeleri verilmiştir. Bir dikdörtgen şeklinde olan işlem simgesi içerisine yapılacak işlemin ne olduğu açıkça yazılmaktadır. Fonksiyon çağırmada ise, simgenin içerisinde fonksiyonun adı ve formal parametreleri belirtilmektedir.

Örneğin klavyeden girilen iki sayının aritmetik ortalamasının hesaplayan algoritmanın akış şeması Şekil 3.8'de verildiği gibi gerçekleşmektedir. Bu örnekte gerekmediği için, karşılaştırma ve döngü simgeleri ile saklama veya veritabanı simgeleri kullanılmamıştır.



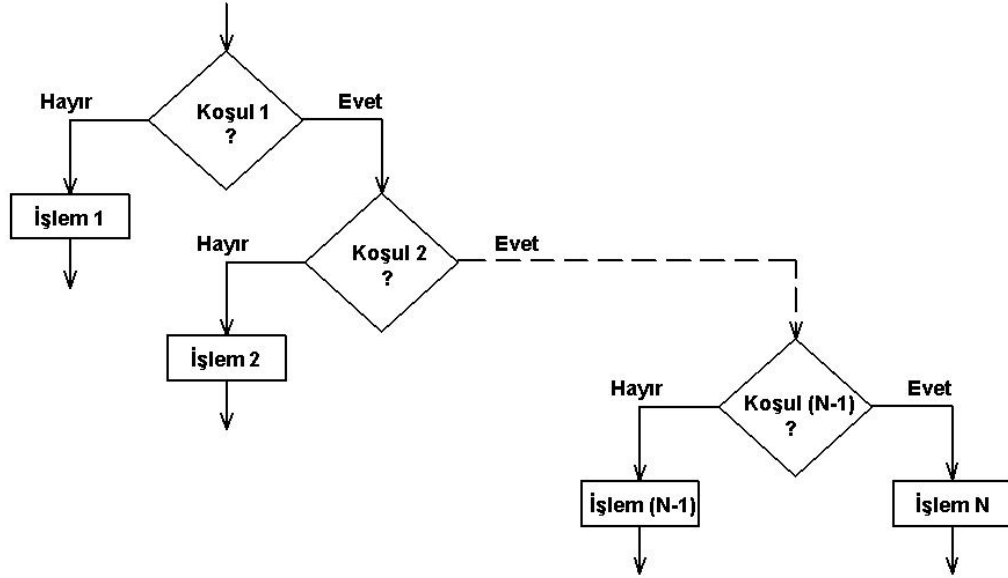
Şekil 3.8 : İki sayının aritmetik ortalaması akış şeması (Çölkesen, 2003)

Koşulla dallanma simgesi yalın olarak baklava dilimi şeklindedir. Ancak akış şeması üzerindeki kullanımı Şekil 3.9'daki gibi olmaktadır. Simgenin içerisine koşul yazılmakta ve en fazla iki yöne dallanma yapılabilir. Simgenin bir tane giriş noktası vardır ve koşulun sonucuna göre iki tane çıkış vardır. Koşul olumlu ise Evet, olumsuz ise Hayır olarak etiketlenen yöne dallanılır. Aşağıda, a) seçeneğinde, koşulun olumlu (evet) ya da olumsuz (hayır) olması durumunda ayrı ayrı yapılacak işlem olması, b) seçeneğinde olumlu olduğunda N tane işlem, olumsuz olduğunda ise işlem olmaması durumları gösterilmektedir.



Şekil 3.9 : Koşullu dallanma simgesi örnekleri (Çölkesen, 2003)

Şekil 3.10’da verilenler, koşullu dallanmanın tipik olarak uygulama şekilleridir. Tasarımcı, simgenin bir tane girişi, en fazla iki tane çıkışı olacak şekilde iç içe geçmiş bir şekilde de düzenleyebilir. Şekil 3.10’da, bu durum için bir örnek olarak verilmektedir.



Şekil 3.10 : İç içe koşullu dallanma simgesi örnekleri (Çölkesen, 2003)

Döngü simgesi yalın olarak basık altıgen gibidir. Ancak akış şeması üzerinde kullanılırken, genel olarak döngü sayacı, dönme koşulu ve sayacın artım bilgisi verilmelidir. Eğer döngü sayacı kullanılmıyorsa, sayacın arttırılması diye bir durum da yoktur, bu durumda yalnızca koşul bilgisi verilir.

3.6. Veritabanı Sisteminin Tamamlanması ve Test Edilmesi

3.6.1. Veritabanı Sistemi İçin Kullanıcı Arayüzü Hazırlanması

Veritabanında varlıklar, ilişkiler haline dönüştürülmüş olarak saklanmaktadır. İşlenebilmesi için, bulunduğu ilişki ve tablolardan uygulanabilir hale getirilmesi gerekir (Materyalizasyon, Somutlaştırma). Ancak somut hale getirildikten sonra, verinin üzerinde güncelleştirme, silme, görüntüleme, çıktı alma işlemleri yapılabilir. Bu işlemler sürdürülürken, kısıtlamalara uyulması ve hataların önlenmesi gerekir. SQL sadece ilişkileri sorgulayabildiği için, varlıkların sorgulaması da ancak uygulama programı ile gerçekleştirilebilir.

Veritabanı kullanıcı arayüzü, bir arabirim işlevi görmektedir. İşlevi ve yapısı tek tip olmayıp, kullanıcının beklentisine göre, basit bir liste oluşturmaktan karmaşık mimarlık çizimleri ve hesaplarına kadar değişik hizmetler verebilecek düzeyler için tasarlanmaktadır. Tasarım konuları; Kontrol mekanizması tasarımı, Form tasarımı, Rapor tasarımı olarak gruplandırılabilir.

Kullanıcı arabirimi olarak uygulama; komut verme, menü kullanımı, grafik çizme işlemleri söz konusudur. Komutlar, veritabanına girdi olarak verilen ve belirli bir işlemin yapılmasını gerektiren deyimlerdir. Menü, veritabanı kullanıcılarına sunulan bir seçenekler listesidir. Kullanıcı, bu listeden yapacağı işlemi seçmektedir. İşlemler, bu liste ile sınırlıdır. Menü hiyerarşinin kurulmasında, varlık ve işlem ya da işlem ve varlık şeklinde iki ayrı yol izlenebilmektedir. Varlık ve işlem sıralamasında önce işlenecek varlıkları gösteren bir liste sunulmaktadır. Arkasından da bu varlık için uygulanabilecek işlemler sıralanmaktadır. İşlem ve varlık şeklinde ise, tersine önce işlemler verilmekte, sonra da bu işlemin uygulanacağı varlık seçilmektedir. Uç kullanıcıların düşünce biçimine, birinci hal daha uygun düşmektedir. Grafik Arabirimi (Graphical User Interface; GUI), pencere, grafik simgeleri, Şekil Listesi (Pop-Down List Box) vb. yapılar olup, genellikle bir fare (mouse) gösterge ile yönetilirler. Bir Diyalog Kutusu (Dialogue Boxes) aracılığı ile ve menü seçimi ile birlikte işlem komutu da yerine getirilebilir.

Form bir ekran görüntüsü olup, genellikle veri girişi ve veri düzenlemesi için kullanılmaktadır. Ayrıca, rapor hazırlamakta da yararlanılmaktadır. Formlar değişik niteliktedir. Bazıları basit ve kullanışlı olup, sadece giriş hataları yapılabilmektedir. Bir kısmı ise karmaşıktır. Karmaşıklığı önlemek için form yapısı basit ve kolay kullanılır olmalıdır. Varlığın yapısını tam olarak yansıtmalıdır. Formun başlığında ve birinci bölümünde varlık tanımlanmalıdır.

Rapor veritabanındaki verinin çıkarılmasıdır. Raporlar kağıda basılabilmekte, ekranda gösterilmekte ya da başka çıktı araçlarında yayınlanmaktadır. Raporlar süreli olarak, özel durumlarda ya da isteğe bağlı olarak düzenlenmektedir. Raporlar da formlar gibi, uygulama programı yardımı ile düzenlenmekte ya da veritabanı yönetim sistemi tarafından üretilmektedir. Çoğu veritabanı yönetim sistemi ürünleri veritabanındaki her kütük için otomatik şekilde tanımlanmış hazır bir rapor da içermektedir. (Kalıpsız, 2001)

Hazır raporlar genellikle basittir. Uygulama programları ile daha karmaşık raporlar düzenlenebilmektedir. Böylece, kullanıcının istekleri tam olarak yerine getirilebilmektedir. Ancak bir uzmanlık işi olduğu gibi, düzenlenmesi de çok zaman almaktadır.

Rapor tasarımında uyulacak ilkeler, form tasarımına benzerdir. Burada, rapor yapısının dayandığı varlığın yapısını aynen yansıtması gerekmektedir. Raporun adı, varlık içerisindeki ilişki girişini göstermelidir. Ayrıca varlığın anahtarı da ilişkiye giriş noktasının anahtarı olmalıdır. Rapordaki Satır Maddeleri (Line Item) sıra numarasına, tarih sırasına ya da başka bir özelliğine göre sıralanabilmektedir.

3.6.2. Veritabanı Sisteminin Yüklenmesi

Veritabanı tasarımı tamamlandıktan, dış bellek sistemi belirlendikten ve gerekli donanım sağlandıktan sonra, tasarlanan veritabanının dış belleklere yükleme işlemi yerine getirilmelidir. İlk Yükleme (Initial Loading) adı verilen bu işlem ile, yeni veritabanına veri yerleştirilmektedir. Yerleştirme, verinin girilmesi, veri gruplarının oluşturulması ve indekslerin kurulması yoluyla gerçekleştirilmektedir.

İşletilmekte olan bir veritabanında, yeni bir tablo veya indeks oluşturulması, güncelleştirilmesi ya da bir sıra eklenmesi, veri tanımlama dilleri ile kolayca gerçekleştirilmektedir. Ancak bu yoldan sınırlı sayıdaki veri üzerinde işlem yapılabilmektedir. Oysa veritabanının ilk yüklenmesinde, milyonlarca verinin kaydedilmesi gerekmektedir. Bu verinin birer “Ekle” (Insert) komutu ile veritabanına aktarılması düşünülemez.

İlk yükleme, bu duruma göre bazen kolayca gerçekleştirilebildiği gibi, bazı hallerde de veritabanının tasarımı ve oluşturulmasının en güç ve zaman alıcı işi olabilmektedir. En büyük sorunlar özellikle, verinin bir sistemden başka birine aktarılmasında baş göstermektedir.

İlk yükleme sırasındaki zorluklar ve zaman kaybı, yüklenecek verinin kaynağına ve formatına, kayıt sayısına, oluşturulacak indeks sayısına ve kullanılacak veritabanı yönetim sistemine göre çok değişmektedir. Örneğin, kaynak verinin bulunduğu araçların yeni sisteme aktarmaya uygunluğu ve okuma olanakları, formatların ve kodlamanın farklılığı çok etkili olmaktadır. Kaynak verinin makine tarafından

okunamaması, büyük zorluk çıkarmaktadır. Kodlamanın farklı olması halinde, birinin diğerine dönüştürülmesi gerekmektedir, iç veri formatlarının ayrılığı halinde, veri ve sayısal alanların iç formatlarının değiştirilmesi yoluna gidilecektir. Bu değişiklikler için, kullanıcıların dönüştürme programları yazması gerekmektedir.

Kayıt ve indeks sayılarının çokluğu yükleme süresini olumsuz etkilemektedir. Yükleme süresi veritabanının oluşturulmasında önemli bir konudur. Küçük bir veritabanının ilk yüklemesi saatlerce, milyonlarca kayıttan oluşan büyük bir veritabanında ise günlerce sürebilmektedir. Genel olarak, onbin kayıt ve beş indeksten oluşan bir tablonun yüklenmesi için, ortalama 60.000 kez disk erişimi gerekmektedir. Seçilen veritabanı yönetim sistemine göre de, ilk yükleme işleminin süresi ve sorunları farklılık göstermektedir. İlişkisel, hiyerarşik ve ağ biçimi sistemlerden yüklenmesi en kolay olanı, ilişkisel veritabanı yönetim sistemi olarak görülmektedir. **(Kalıpsız, 2001)**

3.6.3. Dönüştürme ve Bütünleştirme İşlemleri

Dönüştürme (Converting), bir veri gösteriminin, bilişim içeriğinden kaybetmeksizin, başka bir veri gösterimine çevrilmesidir. Bütünleştirme (Integration), komutların paylaşımı ve bilişimin bir programdan diğerine akışının sağlanması anlamındadır. Veritabanını ilk yükleme aşamasında, dönüştürme ve bütünleştirme işlemleri gerekmektedir. Veritabanı ortamında en çok zaman alıcı işlem, eldeki veri kütüklerinin verisini veritabanına dönüştürmektir. Çoğu ortamda, dönüştürülecek veri çok sayıda kütüğe dağıtılmış olarak ve farklı formatlar halinde bulunmaktadır. Bu veri içerisinde de önemli ölçüde gereksiz ve yinelemeli bilgi olabilmektedir. Dağınık veriyi veritabanı merkezinde toplamak için, çok sayıda uygulama programı yazılması gerekmektedir. Bu programları yazmak çok zaman alıcı bir işlemdir.

Dönüştürme programlarının yazımı, gereksiz veri içeren çok sayıda kütüğe erişilmesini gerektirmektedir. Bazı hallerde, veri kütüklerinin sadece bir kısmı alınmaktadır. Dönüştürmenin nereden nereye yapılacağının kararlaştırılması, uygulama kodlarının yazılması, test edilmesi ve gözden geçirilmesi, çok sayıdaki diğer programlarla ayarlanması zaman alıcı bir işlemdir.

Dönüştürme ve bütünleştirme işlemi sırasında, önce verinin geçerliğinin sağlanması ve sonra yerleştirilmesi gerekmektedir. Veri dönüştürmede yapılacak hatalar,

sonradan kolayca giderilememektedir. Dönüştürme işlemi hemen bitirilememekte, aylarca sürmektedir. Bu arada, bir yandan kurumun işlevleri ve görevlilerin işleri de sürdürülecektir.

Bazı yükleme programları ve yardımcı araçlar olmasına karşın, çoğu veritabanı yönetim sisteminde eldeki veri kütüklerini veritabanına dönüştüren genel amaçlı yazılım bulunmamaktadır. Bu konuda en önemli engel, değişik ortamlardaki veri kütüklerinin birbirine benzememesidir. Genellikle her kurum, veri kütüklerini kendi ihtiyacını karşılayacak şekilde, özel olarak geliştirmektedir.

3.6.4. Veritabanı Sisteminin Test Edilmesi

Veritabanı yüklendikten sonra, sistemin test edilmesi gereklidir. Bu testin amaçları; Veri yüklemede bütünlüğün sağlandığını, Uygulama programları arabirimlerinin veritabanı ile uygunluğunu, Sistem performansının gereksinimleri karşılamasının denetlenmesidir. Bu denetimler, veritabanı tasarımının diğer aşamalarında da uygulanmaktadır. Fakat bu kez veritabanı sisteminin veri yüklenmiş halde ve gerçek ortamında topluca denetimi ve onayı sağlanmış olmaktadır.

Veritabanının denetimi ve onaylanması, üç aşamada tamamlanmaktadır. İlki; veritabanı yüklenmesinin onayıdır. Küçük bir veritabanında bile, verinin tümünün ve ilişkilerinin doğruluğunu sağlamak çok güç ve zaman alıcıdır. Ancak her kayıt tipinden ve ilişkiden alınan örnekler üzerinde hata aramak yoluna gidilmektedir (Örnekleme Yöntemi). Veritabanının onaylanması için, ilk yükleme sorunlarını ortaya çıkarmak üzere, programlar yazılmaktadır. Bu programların başlıca işlevleri;

- Kayıt anahtarlarının işlevliklerini denetlemek,
- Kayıt tipleri arasındaki ilişkilerin doğruluğunu sınamak,
- Özel formatlar için veriyi denetleyerek, yükleme sırasında hiçbir alanın atlanmadığını belirlemek olarak özetlenebilir.

Bu programlarla çok sayıda veri tutarsızlığı ve diğer hatalar bulunmasına karşın, yine de geride bulunamayan bazı hatalar kalmaktadır. Bu testler çok zaman alıcıdır. Fakat yükleme sonundaki denetim ile, ileride çıkabilecek çok önemli sorunlar önlenilmektedir.

Veritabanının denetimi ve onaylanması işleminin ikinci aşamasını, bütünlük sistem testleri oluşturmaktadır. Bütünlük testi ya da sistem testi olarak adlandırılan bu denetimde, uygulama programlarının ve bunların veritabanı ile oluşturulan arabirimlerinin doğruluğu aranmaktadır.

Bütünlük testten önce, uygulama mantığı işlevsel olarak sınanmaktadır. İşlevsel test, bireysel programların veriyi doğru işlediğini göstermek üzere, bir test veritabanı üzerinde uygulanmaktadır. Bu test, veritabanının küçük bir örneği üzerinde yapıldığı için, her zaman bir "örnekleme hatası" söz konusudur. Veritabanının bütününde sonuç biraz daha farklı olabilir. Ancak bu aşamada, veritabanını bir bütün halinde test etmek olanaksızdır.

Bütünlük test, tüm işlemlerin ve raporların işletilerek denetlenmesidir. Bu test ile veri formatlarında tutarsızlıklar ve programlar ile veri arasındaki uyumsuzluklar büyük ölçüde ortaya çıkarılabilmektedir. Yeni sistemde gerçek değerlerin işlenmesinden elde edilen sonuç, eski sistem ile bulunmuş olan değerle karşılaştırılmakta ve doğruluğu onaylanmaktadır.

Veritabanının denetimi ve onaylanması işleminin son aşamasını dayanıklılık testleri oluşturmaktadır. Genel olarak, veritabanı sistemi faaliyete geçirilmeden önce, son olarak dayanıklılık testi uygulanmaktadır. Dayanıklılık testin amacı, yanıt süresi, işlem hızı, eşzamanlı kullanıcı sayısı vb. olarak sayılan sistem performans özelliklerinin gereksinimleri karşıladığının belirlenmesidir. Dayanıklılık testi, daha önce modeller veya prototipler kullanarak yapılmış olan performans analizlerini doğrulamakta ya da çürütmektedir. Dikkatli planlanması halinde, bütünlük ve dayanıklılık testleri eşzamanlı olarak da uygulanabilmektedir.

Dayanıklılık testi, sistem kullanıcılarının normal ve yoğun iş koşullarında yapacakları girişleri simüle etmek yoluyla gerçekleştirilebilmektedir. Değişik yörelerde kullanıcıları olan büyük bir sistemde, böyle bir uygulama çok güç ve pahalıdır. Testte görevlilerden yararlanılması halinde, iş gününde işlemlerin durması, tatilde ek ödeme nedenleri ile gider artmaktadır. Bu yaklaşımın olumlu bir yanı, kullanıcıların eğitiminde son basamak olmasıdır.

Dayanıklılık testinin uygulanmasında en ekonomik yöntem, işlemleri simüle eden bir "işlem sürücü program" (Transaction Driver Program) kullanmaktır. İşlem sürücü,

makinenin okuyabileceği formatta bulunan bir dizi işlemleri sunmaktadır. İyi bir işlem sürücü, sisteme değişen hızda ve değişik işlem karışımında işlemleri sağlayabilmektedir. Bu yöntem, işlem yükünün doğru biçimde kestirilmesi yanında, yedek kapasite ve dar boğazlar bakımından da önemli bilgiler sağlamaktadır. Sistem, ancak bu yükleme koşullarının değerlendirilmesiyle, en yüksek performans için düzenlenebilir. Test sonunda işlem hızının düşük bulunması, işlem sırasında bellek ve disk kapasitesinde vb. konularında bazı sorunların saklı olduğunu gösterecektir.

Dayanıklılık testi, yinelemeli olarak da uygulanmaktadır. Her test, işlem hızları ve yanıt süreleri gibi performans verisini vermektedir. Her testin sonunda bulunan performans verisi incelenmekte ve yeterli görülmemesi halinde, performansı yükseltmek için gerekli işlem yapılmaktadır. Bellek alışverişi aşırı ise, programı değiştirmek ya da bellek eklemek yoluyla performans yükseltilebilmektedir. Eğer bir disk sürücüsü diğerlerinden daha meşgul ise, dengelemek için bazı kütüklerin dağılımı değiştirilmelidir. Bir işlem çok uzun sürüyorsa, yükünü paylaşmak üzere yeni bir işlem eklenmelidir. Performans kriterlerinin yeterli bulunması halinde, sistemi faaliyete geçirmeden önce, performansın daha da yükseltilmesi olanakları üzerinde durulmalıdır. Sistemin çalışmaya başlatılmasından önce, performansı yükseltme olanaklarını gerçekleştirmek, daha sonra alınabilecek önlemlere oranla çok daha kolaydır. **(Kalıpsız, 2001)**

Test işlemleri tamamlandıktan sonra, veritabanı güncelleştirilerek, aktif duruma getirilmektedir. Böylece sistem çalışmaya başlatılmış olmaktadır.

4. VERİTABANI SİSTEMİ YÖNETİMİ

4.1. Veritabanı Sistemi Yönetim Fonksiyonu

Veri yönetimi, istenilen bilgiye, istenilen noktadan istenildiği anda erişimi gerektirmektedir. Burada önemli olan veritabanlarında tutulan yapısal verileri ve çek, ekstre, faks, imaj, elektronik imza, e-posta gibi yapısal olmayan verileri bir arada yönetebilmek ve entegre edebilmektir. Gerek yapısal olan, gerek yapısal olmayan verilerin, dökümanların ve içeriklerin yönetiminde ve tüm bu bilgilerden Kurumsal Zeka (Business Intelligence) olarak adlandırılan bir değer yaratmada, veritabanı yönetimi için bulunan çözümler, çok önemli bir detay olarak karşımıza çıkmaktadır. (www-5.ibm.com, 2004)

Veritabanı yönetiminin amacı, belirli gider ve olanakların sınırı içerisinde kalarak, kullanıcıların ihtiyacını karşılayacak şekilde, veritabanının sürekli işletiminin sağlanmasıdır. Yönetim fonksiyonları, izleme, sistem performansı düzenleme, iyileştirme ve genişletme, veritabanının yeniden düzenlenmesi olarak tanımlanabilir.

4.1.1. İzleme (Monitoring)

Veritabanı durağan değildir. Zaman içerisinde yeni veri ve uygulamalar eklenmekte, değiştirilmekte ve çıkarılmaktadır. Bu değişiklikler, veritabanının performans özelliklerini etkilemektedir. Bu nedenle, değişikliklerin etkilerini belirlemek ve gerekli önlemleri zamanında alabilmek için, veritabanı sürekli olarak izlenmekte ve gözlenmektedir. Gerektiğinde kullanabilmek için, değişiklik öncesinde ve belirli zamanlarda veritabanının kopyası bir yedek belleğe alınarak, güvenli bir yerde saklanmaktadır. (Kopyalama, Back-Up) Gerektiğinde yedek olarak alınan bu kopya, tekrar veritabanına aktarılmaktadır .(Restoring)

Veritabanının işlenmesinde izlenen yol, veritabanının ileri süreçteki büyümesini karşılamak üzere, veritabanı kapasitesini denetim altında tutmaktır. Bunun için, diskler, tablolar ve indeksler izlenerek, ek bellek alanı ihtiyacının karşılanabilirliği

güvence altına alınmaktadır. Bellek alanının yetersiz olması halinde, yeni kayıt ve indeks eklenemeyeceğinden, uygulama işlemi gerçekleşmemektedir. Bellek alanının takip edilmesi halinde, atama sorunu önceden sezilerek, ek alan sağlamak üzere düzeltici ayarlamalar yapılabilmektedir.

Bellek alanı kapasitesinin izlemesi yanında,

- İşlemlerin giriş hızları,
- Yanıt süreleri,
- İşlem tiplerinin karışımı,
- Disk oku/yaz hızları,
- Disk ve iç bellek arasındaki değişim hızı,
- istenilen verilerin diskte bulunma olasılığı,
- Ara belleğin kullanımı vb. üzerine de bilgi toplanmalıdır.

4.1.2. Sistem Performansını Düzenleme (Tuning)

Bir sistemin performansının düzenlemesi, bu sistemin performansını yükseltmek amacı ile parametrelerin ya da sistem konfigürasyonunun değiştirilmesi işlemidir. Ayarlama, sistemin tümü üzerinde uygulanmaktadır. Bilgi-işlem sistemlerinde performans, "hız" olarak belirlenmektedir. Çevrimiçi (On-line) uygulamalarda Yanıt Süresi (Response Time) olarak ölçülmekte ve saniye ölçeği ile belirlenmektedir. Toplu İşlemlerde (Batch), İş Süresi (Job-Time) olarak ölçülmekte ve dakikalar, hatta saatler sürebilmektedir. Sistem içinde yazılım ve donanım olarak sayısız bileşen bulunmaktadır ve bu bileşenlerin hepsi sistem performansını (hızını) etkilemektedir. Sistem yanıt verme süresinin ölçümü esnasında, tüm bileşenlerin tek tek gözden geçirilmesi gerektiğinden, zor ve zahmetli bir işlemdir. **(Curtis, 2000)**

Çevrimiçi işlemlerde yanıt süresi, işlem süresi, veri iletişim süresi, disk erişim süresi öğelerinin toplamı olarak, Toplam İş Süresi (Total Transaction Time) olarak belirlenmektedir. Performansı yükseltmek için, bu üç süreyi kısaltabilmek amacıyla önlemler almak gerekmektedir.

Donanımın yetersiz olması ya da tam kullanılamaması, veritabanı performansını olumsuz etkilemektedir. Teyp ve disk gibi dış bellek araçlarının yetersiz ya da aşırı

yüklü olması, bellekte boş yer arama nedeniyle, sistemin başarısızlığına yol açmaktadır.

Sistem yazılımı ve veritabanı yönetim sistemi de performansı etkilemektedir. Örneğin, işletim sisteminin bir bölümü olan bellek yönetiminin uygun olmaması, girdi ve çıktı faaliyetinin artmasına yol açabilmektedir. Sistemin girdi ve çıktı arabelleklerinin büyüklüğü, dış bellek erişim sayısı isteğini etkilemektedir.

Veritabanı uygulama programı tasarımının zayıf olması ya da verimsiz kodlanması, performansı etkilemektedir. Düşük performansla aşırı işlem yapılması, veritabanı yönetim sisteminin hatalı sonuç vermesine neden olmaktadır.

Bir sistem kuruluşu tamamlanınca, beklenen iş yüküne göre en iyi performansı sağlayacak şekilde ayarlanmaktadır. Sonraki ayarlamalar, herhangi bir kullanıcı sistemin yeniden ayarlanması gerektiğini hissetmeden önce gerçekleştirilmelidir. Bunun için, sistemin performans özelliklerinin sürekli izlenmesi, doğru zamanın saptanması gerekmektedir. Böylece, performansın düşme eğiliminden, kritik noktaya gelmeden önceki anı kestirilebilmektedir.

Ayarlama işleminde, sistem öğeleri arasında bir denge kurulmaya çalışılmaktadır. Tek bir hamlede, büyük ölçüde bir değişiklik yapmak, dengeyi çok fazla bozabilir ve performansın daha da düşmesine neden olabilir. Ayarlama yönetimi olarak, birden çok seçenek oluşturulması halinde, uygun olan seçenek seçilmeli ve bunun üzerinde çalışma sürdürülmelidir. Değişikliğin sonucunu görebilmek için, tek bir hamlede, sadece bir değişiklik yapılmalıdır. Aynı zamanda birden çok değişiklik yapılması halinde, sonucu hangisinin etkilediği kestirilememektedir. Ayar değişikliği yapıldıktan sonra, performans yeniden değerlendirilmelidir. Genel olarak, ayarlama işlemi azar azar artırılarak sürdürülmelidir. Performans yükselişi görüldükçe, değiştirme yinelenerek, önceden kararlaştırılan düzeye ulaşınca kadar ayarlamaya devam edilmelidir.(Kalıpsız, 2001)

4.1.3. Sistemde İyileştirme ve Genişletme (Upgrading)

Özellikle çevrimiçi sistemlerde, zamanla kullanıcıların gereksinimlerine göre, veritabanına erişim yolunda değişiklikler yapılabilmektedir. Bu değişikliklerde,

bazen tablolara yeni veriler eklenmesi, yeni tablolar, görünüm ve indeksler oluşturulması da gerekmektedir. Veritabanında ve diğer sistem öğelerinde, herhangi bir uyumsuzluk oluşmaması için, değişiklik kontrollü olarak yönetilmelidir. Bu işlem, küçük ölçüde bir tasarım şeklinde yürütülmektedir. Bir istek formu doldurulmakta, bu isteğin etkileri ve yararları analiz edilmekte, analiz sonucunun olumlu olması halinde uygulamaya geçilmektedir. Değişiklik gerçekleşikten sonra, test edilmekte ve sisteme yerleştirilmektedir. Bundan sonra da, sistem yakından izlenmekte ve görülen anormallikler düzeltilmektedir.

4.1.4. Veritabanını Yeniden Düzenleme (Reorganization)

Çoğu veritabanı ortamında zaman zaman, veritabanının temizlenmesi gerekmektedir. Veritabanının yüklenmesinden bir süre sonra, bazı kayıtların çıkarılması sonucu, geçerli kayıtların arasında kullanılmayan boş alanlar oluşmakta, ayrıca alanda parçalanmalar olmaktadır. Eklenen kayıtlar, mantıksal sıralı olarak saklanamamakta ve bu nedenle uzun zincirler oluşmaktadır. Kayıtların alınıp getirilmesi ya da ardışık işlem süresi uzamaktadır. Çoğu veritabanında silinmiş verinin bulunduğu alanı geri kazanmak ve taşma olan alandaki kayıtları asıl yerine getirmek için hazır programlar bulunmaktadır. Veritabanının reorganizasyonu adı verilen bu işlemde, eski veritabanından silinenler dışındaki kayıtlar, başka bir dış belleğe kopya edilmekte ve geçerli kayıtlar tekrar bloklar halinde düzenlenerek, yeniden veritabanına yüklenmektedir.

Reorganizasyon zamanında kayıtlar, çoğunun fiziksel sırası mantıksal sıralamanın aynı olması şeklinde düzenlenebilir. Ya da çok kullanılan kayıtlar hızlı bir birime, az kullanılanlar da daha yavaş bir birime yerleştirilebilir. İdeal şekil, reorganizasyonun veritabanı yönetim sistemi tarafından sürekli olarak kontrol altında bulundurulmasıdır. Bu durumda, veritabanındaki bir kayıt silinince, otomatik olarak yeri geri alınmaktadır.

4.2. Sistemin Çalışma Hızı Ve Bellek Gereksinimi

Programın çalışma hızı, belirlenen bir problemin çözümü için tasarlanan program kodunun görevini yerine getirmesi için gerekli zaman bilgisini verir. Bu bir sayı olabileceği gibi, bir matematiksel ifade de olabilmektedir. Bellek gereksinimi ise,

programın yürütülmesi sırasında, program kodu ve kodun yürütülmesinde kullanılan verilerin işgal ettiği ya da gereksinim duyduğu toplam bellek alanıdır.

Program çalışma hızı, program kodunun zaman açısından başarısını; karmaşıklık ise, ilgili program kodunun parametre sayısına bağlı olarak çalışma süresini ve gerekli işlem sayısının artış miktarını gösterir. Çalışma hızını belirlemek için, Kıyaslama (Benchmarking) ve Algoritma Analizi (Algorithm Analysis) olarak adlandırılan iki yol vardır. Birisi uygulamanın gerçekleştirilmesini, diğeri ise bağıntı hesabı gerektirmektedir. Kıyaslamada söz konusu algoritma gerçekleştirilir ve örnek veri için çalıştırılır. Algoritma analizinde ise, belirli bir veri yapısı için, söz konusu algoritmanın analizi yapılır ve bir zaman bağıntısı bulunur. Analiz, algoritma gerçekleştirimi karmaşıklaştıkça, ciddi bir matematiksel birikim haline gelir. (Çölkesen, 2003)

4.2.1. Kıyaslama (Benchmarking)

Programlamada benchmark, bir nesnenin gerçek performansına değer biçmek için, standart test ve denemelerden geçirilmesi sonucu olarak karşımıza çıkmaktadır. Üreticiler, daima kendi ürünlerini en iyi gösteren benchmark uygulamalarını tavsiye etseler de, kullanıcılar bu benchmarkları değerlendirmek zorunda değillerdir. Piyasada her türlü yazılım ve donanım için, Benchmark programları bulunmaktadır. (<http://en.wikipedia.org/wiki/Benchmark>, 2004)

Genel olarak benchmark, sistemin performansını belirleyen bir ya da birçok değişkenin tanımlanmasından oluşmaktadır. Sistemin kapasite ölçümü için, standart, kabul edilmiş bir yöntem yoktur. Kapasite, mevcut sistemi, benzer sistemlerle kıyaslama yoluyla ölçülür. (<http://rkb.home.cern.ch/rkb/AN16pp/node14.html>, 2004)

Kıyaslama, geliştirilen programın önceden belirlenmiş örnek giriş değerleri için çalışma hızını gösteren deneysel sınama sonucudur. Programa, giriş değerleri olarak, her olası durumu içeren örnek giriş verileri (driver data) uygulanır ve bunlara karşı düşen başarı skoru elde edilir. Bu skor, çoğu zaman, belirli bir maksimum değere göre orantılanmış bir gerçel veya tamsayıdır. Örneğin en yüksek başarının 1.0 olduğu

skor deęerlendirmesinde bařarım sınaması sonucu 0,5 veya %50 ıkarsa yarı bařarımlı olarak deęerlendirilir.

Kıyaslamada en nemli unsur, giriř verilerinin oluřturulmasıdır. Elde edilen sonuların tartıřılmaz olarak karřılařtırmalarda kullanılabilmesi iin, seilen rnek veriler herkes tarafından kabul edilen evrensel anlamda olmalıdır. Bu tr sınama verileri, eřitli uygulama alanları iin belirlenmiřtir. Uluslararası dzeyde karřılařtırma yapılması isteniyorsa, bu rnek veriler kullanılmalıdır. Eęer programcı tarafından hazırlanmak istenirse, her olası durumu yeteri kadar sayıda sınayacak řekilde olmalı ve en kt durum, en iyi durum gibi zelliklere dikkat edilmelidir.

4.2.2. Algoritma Analizi

Algoritma analizi, tasarlanan program veya fonksiyonun belirli bir iřleme gre matematiksel ifadesinin bulunmasına dayanmaktadır. Temel hesap birimi seilerek, programın grevini yerine getirebilmesi iin, bu iřlemden ka adet yapılması gerektięini bulmaya yarayan bir baęıntı hesaplanır. Eęer bu baęıntı zamanla ilgiliyse, alıřma hızını; bellek gereksinimiyle ilgiliyse, bellek gereksinimini ortaya koyar. Eęer bir programın yrtme zamanı ve bellek gereksinimi hesaplanmışsa, o programın eřitli bilgisayar donanım ortamlarındaki bařarısı ngrlebilir.

4.2.3. Bellek Gereksinimi

Programın bellek gereksinimi, programın veya fonksiyonun yrtlmesi sırasında gerekli bellek miktarıdır. Biri program kodu, ikincisi programın zerinde alıřtıęı veri ve ncs de fonksiyon aęırmalarda kullanılan yıęın (stack) alanı olmak zere  kısımdan oluřmaktadır. Bir programın bellek gereksinimi hesaplanırken, bu  unsur gz nnde bulundurulmaktadır.

Eęer program, alt programlara ayrılmamıřsa, program kodu, yaklařık olarak, doęrudan program kodunun saklandıęı dosya byklęndedir. Ancak, program, prosesler řeklinde tasarlanmışsa, yani alt programların birleřmesinden oluřuyorsa, bu durumda program kodu, bellekte, aynı anda alıřması gereken proseslerin iřgal ettięi alan byklęndedir. Veri ise, program ierisinde kullanılan deęiřken, dizi veya veri yapılarının toplamıdır.

Veri için bellek gereksinimi, doğrudan program içerisinde kullanılan değişken veya sabit sayı gibi veri yapısı türüne bağlıdır. Bellek gereksinimi, kendi içerisinde biri statik, diğeri dinamik bellek olmak üzere iki sınıfa ayrılmaktadır. Statik bellek gereksinimi, program içerisindeki global değişken ve veri yapılarından hesaplanabilmektedir. Dinamik bellek gereksinimi ise, programın yürütülmesi anındaki gereksinime göre belirlenmektedir. Dolayısıyla programın çalışması sırasındaki parametrelere bağlıdır. Ancak, örnek olaylar için, en kötü durumda gerekli bellek gereksinimi hesaplanabilir.

Hemen hemen her sistemde, yığın olarak adlandırılan bellek alanı bulunur. Burası son giren, ilk çıkar (LIFO) davranışında olup, bilgilerin geçici olarak saklandığı bellek alanıdır. Burası işletim sistemde yürütülen her program tarafından bilinmektedir. Dolayısıyla program veya fonksiyonlar bu alan üzerinden birbirlerine parametre aktarabilirler.

Kesme hizmet programları da yığını kullanır. Sisteme bir kesme isteği geldiğinde, işlemci o anda yaptığı işi bırakır ve o andaki durum bilgilerini (saklayıcı içeriklerini, dönüş adresini vs.) yığına atar. Kesme hizmet programından döndüğünde, yığından eski bilgilerini alır ve kaldığı yerden devam eder. (Çölkesen, 2003)

4.3. Veritabanı Sistemin Doğruluğu ve Hata Kaynakları

Veritabanının doğruluğu (Integrity), doğru ve geçerli sonuçlar elde edilebilmesi anlamına gelmektedir. Veritabanında doğruluğu sağlamak ve sürdürmek için, veritabanı içeriğinin niteliğini ve yapılan işlemlerin uygunluğunu güvence altına almak gerekmektedir. Bunun için hata kaynaklarının bilinmesi önemlidir. Sistemde gözlenebilecek hataların başlıcaları şunlardır;

- Kullanıcı hataları: Her düzeyden kullanıcı, kullanım ve yönetim sırasında çeşitli hatalara düşebilmektedir. Örneğin, programcı kodlama işlemi esnasında mantık hatası yapabilmektedir. Derlenen bilgi, anlamsız ya da hatalı olabilmektedir. Ayrıca, operatörler (işletmenler) ve veritabanı yöneticileri de çeşitli hatalara yol açabilmektedir.

- Donanım hataları: En az rastlanan, fakat veritabanını en fazla etkileyen hata tipidir. Bunlardan en önemlisi, bir disk kafasının hasara uğraması ve bu birim üzerindeki bazı izlerin içeriklerinin kaybolmasıdır. Zarar gören alanın veritabanı kayıtları veya indekslerinin bulunması halinde, veritabanının doğruluğu sağlanamaz. Bu halde, veritabanı sağlam bir birime kopyalanmalı ve ancak kayıp veri tamamlandıktan sonra kullanılmalıdır.

Başka bir donanım hatası da, veritabanının güncelleştirilmesi sırasında elektriğin kesilmesi veya merkezi işlem biriminin bozulmasından ileri gelmektedir. Veritabanı işleminin tamamlanmamış oluşu halinde, hata yapılmaktadır. Bu hatayı gidermek için, işlemi tersine döndürmek (rollback, backout) ve donanım hatası giderildikten sonra tekrar başlatmak gerekmektedir. Ayrıca, bilinmeyen veri iletişim hataları da verinin yanlış alınmasına neden olmaktadır. Bu yanlış veri, veritabanına geçmektedir. Donanım hatası nedeniyle oluşan hatalı veri, bellekte ya da diskte saklı veriyi de etkilemektedir.

- Sistem yazılımlarının hataları da veritabanının doğruluğunu bozabilmektedir. Örneğin, veritabanının işletilmesi sırasında işletim sisteminin anormal biçimde duraklaması, veritabanının tutarsız durumda kalmasına neden olabilmektedir. Bu durumda, donanımın zamansız durmasında olduğu gibi, önce veritabanının işlem öncesi durumuna getirilmesi ve yarıda kalan işleme başından başlanması gerekmektedir. Başka bir sorun da, işletim sisteminin veritabanını eşzamanlı olarak kullanması halinde çıkabilmektedir. İki ayrı uygulama programı ile veritabanının belirli bir kaydına güncelleştirmek amacı ile tesadüfen erişilirse, güncelleştirme iki ayrı şekilde ve birbirinden farklı olarak gerçekleştirilmektedir. Bir yazılım olan veritabanı yönetim sistemi, her yazılımda olduğu gibi, mantıksal hata içerebilmektedir. Bu durumda, veritabanındaki veri kaybedilmekte ya da yanlış olabilmektedir. Verinin kaybedilmesi, fiziksel olarak ortadan kalkması ya da fiziksel olarak veritabanında bulunduğu halde erişilememesi gibi sorunlar oluşabilir. Örneğin, bir veritabanı yönetim sisteminin indeks tablosundaki güncelleştirme hatası, bu indeks aracılığı ile bir ya da daha çok sayıdaki kayda erişilememesi ile sonuçlanabilmektedir. **(Kalıpsız, 2001)**

4.4. Veri Sıkıştırma İşlemi

Saklama biriminde tutulan bir metin dosyası, görüntü dosyası, makina kodu içeren bir program dosyası, günümüzde kullanılan kodlama yöntemlerinden dolayı, içerdiği bilgiden çok daha fazla yer işgal etmektedir. Bu dosyalar, içerdiği bilgi değişmeden veya çok az kayba uğratarak, fazlalıkların atılmasıyla, daha az yer işgal edecek biçimde kodlanabilmektedir.

Veri sıkıştırma, aslında bilginin saklama düzeni olan veri yapısının değiştirilmesidir. Bazı metin tabanlı verilerin %60-70 oranında sıkıştırılabildiği görülmektedir. Dolayısıyla saklama birimi sığasından veya veri haberleşmesi kanalı bant genişliğinden önemli ölçüde tasarruf edilmesi sağlanmaktadır.

Sıkıştırma yöntemleri, genel olarak, kayıpsız ve kayıplı teknikler olarak iki sınıf altında toplanmaktadır. Kayıplı teknikler, resim ve ses uygulamalarında, kayıpsız teknikler de metin, program kodu gibi en ufak kayba izin verilmeyecek uygulamalarda kullanılmaktadır. Veri, kayıplı bir teknikle sıkıştırıldığı zaman, bilgiden bir miktar kaybedilir ve sıkıştırılmış veri yeniden açıldığında ilk hali gibi elde edilmeyebilir. Birçok uygulamada bu kabul edilebilse de, bazı uygulamalarda 1 bitlik bir kayıp bile kabul edilememektedir.

Bir algoritmanın sıkıştırma başarısı, ilgili verideki fazlalıkların atılmasına veya azaltılmasına bağlıdır. Veri içinde değişik durumlarda bulunabilen veri fazlalılığı, yöreselliğe dayanarak üç ana grup altında toplanabilir;

- Alanda Yöresellik
- Sıklıkta Yöresellik
- Zamanda Yöresellik

Alanda yöresellik, bazı karakter, katar, hece veya bit dizisinin belirli bir alan içinde tekrarlanmasına dayanır. Örneğin, bir metin dosyası içinde, paragraflar arasında bırakılan boş satırlar, alanda yöreselliğe dayanan fazlalığa neden olur. Bir katar içindeki bazı karakterlerin art arda gelmesi, bir bit dizisi içindeki bazı bit desenlerin art arda bulunması veya belirli aralıklarla bulunması alanda yöreselliğe girer.

Sıklıkta yöresellik, bazı karakterlerin, hecelerın veya katarların diğerklerine göre daha sık kullanılmasına dayanır. Sık kullanılanlara sabit uzunlukta kısa kodlar; seyrek kullanılanlara ise daha uzun kodlar verilerek sıkıştırma yapılır.

Zamanda yöresellik, sıkıştırılması için gerçek zamanda gelen veri içinde, herhangi bir sözcüğün veya hecenin yeniden görölmesine dayanır. Örneğın, her sözcüğe, uzunluđu verinin içinde bulunma sırasına bağılı olan bir kod verilir. Kendisine özel bir kod atanmış olan bu sözcük gelen veride görölmezse, sözcüğe atanmış olan kod alınır ve diğerk bir sözcüğe atanır, görölürse sözcüğe atanmış olan kod olduđu gibi kalır. Genelde, sözlük kullanan ve sözlüđu sürekli güncelleyen sıkıştırma algoritmaları zamanda yöreselliğe dayanan fazlalıkları azaltır.

Veriler, tipine bağılı olarak değışik türde fazlalıklar içerebilir. Evrensel bir sıkıştırma algoritması, verideki her tür fazlalığı bulup azaltırken, bazı algoritmalar yalnızca bir tür fazlalığı bulur ve azaltır. Evrensel olmayan ve farklı türdeki fazlalıkları azaltan algoritmalar art arda kullanılarak sıkıştırma başarımı artırılabilir. Ancak, aynı türden fazlalığı azaltan algoritmaların arka arkaya kullanılmaları sıkıştırma başarısını artırmayacağı gibi, zaman kaybına yol açmaktadır. (Çölkesen, 2003)

4.5. Veri Güvenliğinin Sağlanması

Veri güvenliği, kasıtlı ya da kaza ile veritabanının kötü kullanımının önlenmesi ve böylece verinin güvence altına alınması ile sağlanmaktadır. Bunun için, veritabanındaki verinin yetkisiz kişiler tarafından kullanımına karşı korunması gerekmektedir. Koruma önlemleri, veritabanından yararlanılmasını ve veritabanı kullanımını bir ölçüde sınırlamakta ve güçleştirmektedir. Bu nedenle, ancak gerektiği ölçüde önlem almakla yetinilmelidir. Bu ölçü, verinin gizlilik derecesine ve önemine göre, veritabanı sisteminin tasarımı sırasında kararlaştırılmaktadır. Örneğın, kişisel bilgisayar veritabanında herhangi bir güvenlik önlemine gerek duyulmamasına karşın, yurt savunmasına ya da banka işlemlerine ilişkin bir veritabanının korunması için olağanüstü önlemler alınmaktadır.

Kaza ile veri düzeninin bozulması, işlem sırasında donanım arızası, veritabanına eşzamanlı erişimdeki anormallik ve verinin çok sayıda bilgisayara dağıtımındaki aksaklık, işlemlerin programlanmasında veritabanı dengesini bozucu mantıksal hata

nedenlerinden ileri gelmektedir. Bilerek ve kötü niyetle veritabanına erişimler ise, bilgi hırsızlığı amacı ile yetkisiz olarak veri okuma, veriyi değiştirme veya veriyi bozma eylemleridir.

Veritabanının kaza sonucu bozulmasına karşı korumak, kötü niyetli erişime oranla, daha kolaydır. Kötü niyetli erişimi kesin olarak önlemek mümkün değildir. Ancak, yetkisiz kişilerin veritabanını kullanmasına izin vermemek ve olanak tanımamak yoluyla, bir ölçüde güvenlik sağlanabilmektedir. Bu önlemlere karşın, yine de kötü niyetle veritabanına erişim tam olarak önlenememektedir. Bu konuda literatüre geçmiş çok sayıda olay bilinmektedir. Örneğin; bilgisayarda savaş oyunu oluşturan bir genç, askeri bilgisayar sistemine girerek, neredeyse bir nükleer savaşın çıkmasını gerçekleştirebilecek ortam hazırlamıştır. İki öğrenci tarafından düzenlendiği söylenen "worm" (solucan) adlı bir virüs benzeri programın, 1988 yılında geniş bir bilgisayar ağına yayılarak, 6000 bilgisayar iş istasyonuna bulaştığı saptanmıştır. Bir banka danışmanı, 10 milyon doları önce başka bir bankaya, oradan da elmas satın aldığı bir firmanın hesabına aktarmıştır. Olay ancak, kendisi etrafta öğünerek anlattığı için ortaya çıkmış ve tutuklanmıştır. Bir programcı, ortak veritabanından kasıtlı olarak binlerce kaydı sildiği için, işten çıkarılmıştır. Ayrıca, "Truva Atı" adıyla bir virüs programı geliştirdiği anlaşılarak tutuklanmıştır. Bu örnekler daha da çoğaltılabileceği gibi kuşkusuz, pek çok sayıda olay da yayın organlarına yansımamış ve saptanamamıştır. Bu örnekler, veritabanına hatalı girişlerin yanında, içerden ve dışardan kötü niyetli girişlerin daha da çok olabileceğini göstermektedir.

Fiziksel olarak, bilgisayar sisteminin bulunduğu mekana kötü niyetli kişilerin zorla ya da gizlice girmemesi için gerekli önlemler alınmalıdır. Ayrıca, veritabanı sisteminin kullanımı da belirli ölçülerde sınırlandırılmalıdır. Bu sınırlama, kullanıcılara veritabanının sadece bir bölümüne girme, yalnız sorgulama yapabilme, veritabanını değiştirmesini yasaklama şeklinde uygulanabilmektedir. Bu önlemlerin sadece birinin yetersiz olması, veritabanının güvenliği için harcanan tüm çabaları boşa çıkarabilmektedir. Bu nedenle, her düzeyde güvenliği yeterince sağlayacak önlemler alınmalıdır.

Veri asla tam olarak güvence altına alınamamaktadır. Koruma önlemlerinin artırılmasıyla, kasıtlı ve kasıtsız veri kaybı olasılığı azalmaktadır. Buna karşılık koruma maliyeti de yükselmektedir. Koruma giderlerinin en uygun durumu, korunan

veri öğelerinin değeri ile koruma giderleri arasındaki farkın en büyük olduğu noktadır. Veri öğelerinin koruma değeri, hırsızın buradan sağlayacağı kazanç veya veri sahibinin uğrayacağı zarar ile ölçülmektedir. Ancak, kazanç ve zarar miktarları her zaman para ile ölçülememekte, çoğu kez moral değer kullanım değeri, mekanik değer ya da maliyet değeri halinde belirlenmektedir.

Fiziksel güvenlik, duyarlı ve bozulabilir bilişim içeren bölgeye girişi güvenceye almak ve korumakla sağlanmaktadır. Toplu İşlem (Batch Processing) olarak çalışılması halinde, bilişim sadece bilgisayarın bulunduğu mekanda gerçekleştirildiği için, bu mekanı denetim altına almakla fiziksel güvenlik sağlanabilmektedir. Ancak, bilgisayar ağı sisteminde her terminalden giriş yapıldığı ve çevrimiçi sistemlerde iletişim hatlarına uzaktan iletişim sağlanabildiği için, sadece mekanın denetim altında bulundurulması yeterli olmamaktadır. Bu durumda, veri girişini de güvenceye almak gerekmektedir.

Veriye erişim güvenliği, kimlik denetimi, kullanıcı ve donanım sınırlamaları, donanım güvenlik araçları, mekanik ve program güvenlik işlemleri, güvenlik katmanları vb. önlemlerle sağlanmaktadır.

Kimlik denetimi, kullanıcının, sisteme giriş yapabilmesi için kimliğini kanıtlaması ile gerçekleştirilmektedir. Bu amaçla, günlük kütüğüne kimlik kaydı, kimlik kartı, kart okuyucu vb. yöntem ve araçlar kullanılmaktadır. Ayrıca, bir "Parola Sözcük" (Password) ile kullanıcı kendini kanıtlamaktadır. Yüksek düzeyli güvenlik yöntemi olarak el izi, retina tarama, ses onayı gibi biyolojik yöntemler de kullanılmaktadır.

Kullanıcı sınırlaması, kullanıcıya ancak, kendine izin verildiği ölçüde veritabanından yararlanma olanağının tanınmasını sağlamaktadır. Bu olanak, kullanıcılara veya uygulama programlarına özel olarak düzenlenen dış modeller ile sağlanmaktadır. Böylece, veritabanı yönetim sistemi, kullanıcının hangi görünüme girebileceğini belirleyen bir mekanizma oluşturmaktadır. Veritabanı yöneticisi böylece her kullanıcı için okuma, güncelleştirme, silme, ekleme vb. işlemlerinden sadece birini ya da birkaçını yapmak yetkisi verebilmektedir. Bu yöntemle, veritabanı yöneticisi, kullanıcılara tanıdığı işlem olanaklarını belirlemekte ve "Kullanıcı Profili" adı verilen bir liste haline getirmektedir.

Donanım sınırlaması, sistemin kullanımını sınırlamak ile sistemin denetim altında tutulmasıdır. Sistemin belirli gün ve saatlerde açılması, yetkili kişilerin kullanması, sadece belirli işlemlerin yapılabilmesi vb. sınırlamalar getirebilmektedir. Bu kısıtlamalar genellikle, kullanıcı sınırlamalarını desteklemek için, ek olarak alınan önlemlerdir.

Güvenlik katmanları, bir sistemin güvenliğinin birbirine bağlı çok sayıda önlemlerle sağlanmasıdır. Donanım, işletim sistemi ve uygulama programı, basamaklar halinde birer güvence katmanı oluşturmaktadır. Bir kullanıcı, bu katmanlardan birine geçemez ise, takılmakta ve bir sonraki katmana erişememektedir. Basit bir güvenlik katmanı örneği Şekil 4.1’de gösterilmektedir.



Şekil 4.1 : Güvenlik Katmanları. (Kalıpsız, 2001)

Çok önemli ve gizli nitelikteki verilerin güvenliğini sağlayabilmek için, veri girişinin kısıtlanmasının yanında ek bir önlem olarak, verinin şifrelenmesi yoluna gidilmektedir. Böylece, diğer önlemleri aşarak şifreli veriyi elde edebilen yetkisiz bir kişi şifreyi çözemediğinde, bu veriyi kullanamamaktadır. Şifreleme, bilginin kodlanması, kodların yerlerinin değiştirilmesi, özel bir kod kullanılması şeklinde uygulanmaktadır. Birinci halde ASCII, BCD, EBCDIC gibi standart bir kodlama sistemi kullanılmaktadır. Bu kodlama, sadece verinin makine tarafından okunmasını sağlamakta ve herhangi bir gizlilik içermemektedir. Kodlama, kod yazıcı tarafından gerçekleştirilmektedir. Fiziksel (analog) veri, özel bir transformatör (analog to digital converter, ADC) aracılığı ile, sayısal kodlamaya dönüştürülmektedir. Veri iletişiminde de, kodlama ve kod çözücü (encoder, decoder) araçları ile sinyal ve digital kod dönüşümleri gerçekleştirilmektedir. Şifreleme ise, ikinci ve üçüncü şekillerde, kod sınırlamasının değiştirilmesi ya da özel bir kodlama yöntemi kullanılması hallerinde söz konusudur. Bu hallerde, özel bir kurala göre ve şifre anahtarı kullanarak, kodlama yapılmaktadır.

Kodların yerlerinin değiştirilmesi veya özel bir kodlama oluşturulmasında, şifre anahtarına temel olarak, "Veri Şifre Standardı" (Data Encryption Standard, DES) adı verilen bir standart önerilmiştir. Bu şemaya göre, yetkili kullanıcı kendine özel bir şifre geliştirmektedir. Ancak, şemayı bilen yetkisiz bir kişi, bu şifreyi kolayca çözebilmektedir. Biraz daha etkin bir şema da "Genel Anahtar Şifre" (Public-Key Encryption) olarak adlandırılan bir standarttır. Bu şifre, Genel (Public) ve Özel (Private) nitelikte iki anahtardan oluşmaktadır. **(Kalıpsız, 2001)**

5. VERİTABANI SİSTEMİ VE MİMARLIK İLİŞKİSİ

Mimari tasarım sistemlerinde, yapıların temel özelliklerini tanımlayarak, tipoloji, görsellik ve geometriye ait özellikleri kullanıcıya aktaran görsel veritabanları kullanılmaktadır. Düzenlenmiş veya özel bilgi olarak gruplandırılan tasarım deneyimi bilgisi, bilgisayar destekli tasarım kuralları, çerçeveler ve en son olarak da prototipler biçiminde sunulmaktadır. Bilgisayar destekli tasarım, ele alınan yapıların topolojik ve geometrik özelliklerini içeren veritabanlarının kullanımı ile tanımlanırken, malzeme özellikleri gibi başka sentaktik bilgilerle, nesne sınıflarının tanımlanmasına imkan verir. Temel olarak, görsel veritabanları, sanal kütüphanelerde depolanan görsel nesnelerin farklı sınıf tanımlamalarına göre ayırt edebilme özelliğine sahiptir. Görsel veritabanlarının geliştirilmesi konusunda yapılan araştırmalar, CAD veritabanlarındaki sentaktik bilgiyi, semantik bilgi ile ilişkilendirebilmek amacıyla, yazılı enformasyon sistemlerinin, bilgisayar destekli tasarım sistemleri (CAD) için bütünleştirilmesi yönünde geliştirilmektedir. **(Rosenman, Gero ve Oxman, 1992)**

5.1. Mimarlıkta Veritabanı Uygulamaları

Mimari tasarımcılar, görsel veritabanlarını üç farklı şekilde araştırma yapmak üzere kullandıklarını ifade etmektedirler.

- Kullanıcı, belirli bir resmi aramaktadır. Kullanıcı hangi yapıyı aradığını bildiğinden, indeks sistemini kullanarak, ihtiyaç duyduğu yapı ile ilgili bilgilere kolaylıkla ulaşabilmektedir.
- Kullanıcı, daha az odaklanmış bir arama yapmaktadır. Kullanıcı, belirli bir konu ile ilgili arama yaptığından, veritabanındaki uygun kategori altındaki tüm resimleri görüntülemek istemektedir.
- Kullanıcı, veritabanından rasgele bir resim seçerek, veritabanındaki diğer resim ve yapıların özelliklerini kısmen veya tamamıyla gözden geçirmek istemektedir.

(Gross,1997)

Mimari tasarımın, tasarım aşamasını kolaylaştıran ve destekleyen görsel bilgiye duyduğu ihtiyacı açıkladığı üç tanım aşağıda özetlenmektedir.

- Mimari tasarımın kalitesi, gerçekleştirilmiş veya gerçekleştirilmemiş yapılar için geçerli olduğu gibi, profesyonel ve amatör tasarımlarda da, formların birbiriyle oluşturduğu kompozisyona bağlıdır.
- Mimar, yeni bir tasarımı çalışmaya başlanmadan önce, tasarımcıya verilen bilgi, genellikle form ve kompozisyonlarla ilgili olduğundan, sonuca ulaşılması mümkün olmayan açık uçlu ve hasta tanımlanmış problemle karşılaşmaktadır.
- Bazı mimari tasarımlarda, form ve kompozisyonlarla ilgili kurallar, tasarımcıya sunulan bilgiler arasında verilmektedir. Eğer tüm kompozisyon kuralları ve biçimler belirlenmiş ise, tasarımcı daha az sayıda hasta tanımlanmış tasarım problemini çözmek durumunda kalır. **(Goldschmidt,1997)**

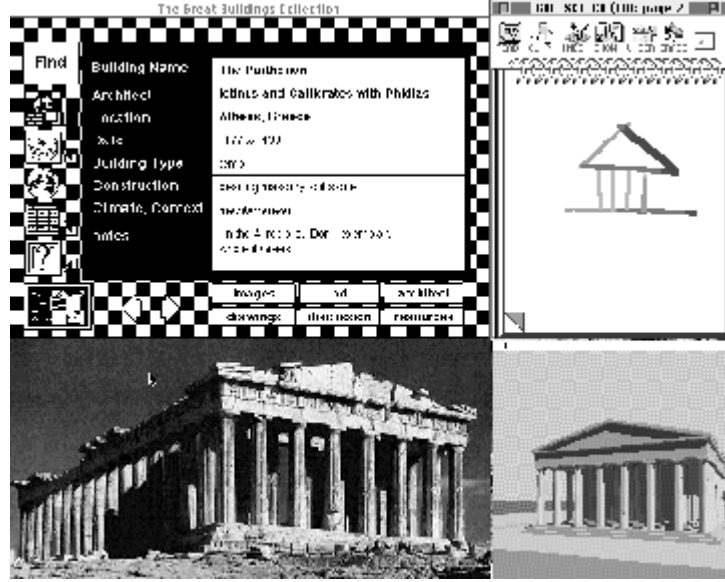
5.2. Tasarımda Veritabanı Uygulamaları

Tasarımcı, yeni ve farklı mimari arayışlara girmeden önce, oluşturulan görsel mimari veritabanından, daha önceki mimarların yaptığı başarılı yapıların araştırılmasında ve yeni mimari tasarımda uygulanabilecek noktaların analiz edilmesinde faydalanmaktadır. Mimarlık bilim dalında, son yıllarda geçmiş mimari örnekleri rasyonel olarak kabullenerek, tasarımcıların mimari verilerden en iyi şekilde faydalanabilmesi için, verilerin farklı kategorilerde ve farklı tasarım amaçlarına uygun biçimde sınıflandırılması tasarımcıya yardımcı olmaktadır.

Yaratıcı ve kavramsal tasarımlarda, mimarlar, kendi tasarımına adapte edebileceği formlar araştırırken, genellikle kitaplara, dergilere ve diğer resim koleksiyonları kaynaklarından faydalanırlar. Bu bağlamda, görsel koleksiyonları tanımlayıcı anahtar kelimelerle indekslenmesi gündeme gelmektedir. Anahtar kelime indekslemesine ilave olarak, tasarım konsepti ve konusuna dayanarak resimlere erişim ve direkt olarak mimari eskizleri anımsatan görsel ve grafik şemalar tasarımcıların farklı alanlardan ilgi çekici referanslar hatırlamasına yardımcı olur.

Eskiz, kavramsal tasarımın temelidir. Mimari tasarımcılar, kavramsal tasarımlarını yaparken, geleneksel kalem ve kağıt kullanmaya alışmışlardır. Çünkü eskiz aktiviteleri bulma, adapte etme, uyum sağlama, daha önce çalışılan şekil ve

yapılardan yeni formlar elde etme gibi tasarım için önemli duyguları içermektedir. CAD yazılımları, mimari çizimler, animasyon, render gibi aşamalarda tercih edilse bile, geleneksel tarzdaki, yaratıcı tasarım için CAD yazılımlarının kullanımı çok fazla yapılandırılmış olarak algılandığından, tasarım sırasında kalem kullanmak temel tasarım aracı haline gelmiştir. Mimarlar, yaratıcı tasarımda benzerlik ve örneklemeyi kullanır.



Şekil 5.1. Great Buildings Arayüzü (Gross, 1995)

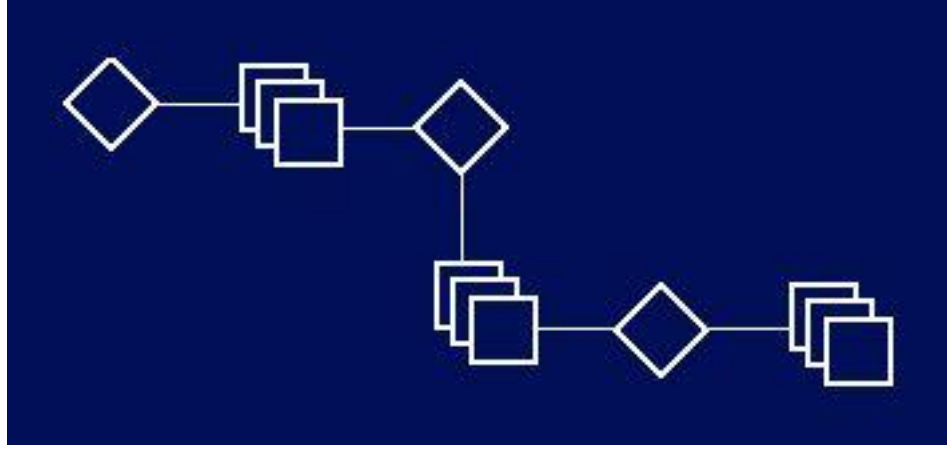
Kayalar ve çiçekler gibi doğal ve yapay form resimlerinin kullanımı mimari tasarımda tartışılan bir konudur. Mimari resimlerin hipermedya ortamındaki veritabanlarının mimarlık tarihi eğitiminde kullanılması, ArchWAIS, DOORS, SPIRO (Şekil 5.2), Great Buildings Collections (Şekil 5.1) gibi mimari slayt arşivi veritabanlarının oluşumunu sağlamıştır. Bu veritabanlarındaki öğeler zor kodlanmış bağlantılar üzerinden ilişkilendirilmiş olmakla birlikte, resimleri tanımlayan, her resim için, isim, tarih, mimar gibi bilgiler veren kelimelerle indekslenmiştir. Bu kurgu, tasarımcının bir veya birkaç anahtar kelimeyle oluşturduğu sorguları kullanarak arama yapmasına imkan vermektedir. Mimari görsel veritabanlarının kullanımı, resim koleksiyonlarının eskiz tabanlı olarak indekslenmesi ve tasarımcının anahtar kelime yerine eskiz veya diyagramlar kullanarak sorgulama yapabilmesiyle genişletilebilmektedir. (Gross, Luen Do, 1995)



Şekil 5.2. SPIRO Arayüzü (Gross, 1995)

Hasta yapılı mimari tasarım problemlerinin çözümünde, tasarımcı analogik stratejileri kullanması gerekliliğinin farkında değildir. Mimari tasarım yöntemlerini, tasarım sürecinin içine katmak üzere yapılan çalışmalar artık durdurulmaya başlanmıştır. Mimarlık bilim dalı, çoğunlukla bilgi amaçlı görsel ifadelerle dayandığı için diğer bilim dallarından ayrıldığından, günümüzde bilgisayar teknolojisinin mimari tasarımcıya verebileceği desteğin anlaşılmasıyla birlikte, tasarım bilgisinin saklanması, geri çağırılması ve yorumlanması gibi çalışmalar daha yaygın olarak kullanılmaya başlanmıştır. Mimari tasarım konseptindeki araştırmaları analiz edebilmek için, bilgisayar sistemlerinin içinde bulunduğu semantik ağdaki "serbest yönetimli tarama" algoritma teknikleri kullanılmaktadır. Semantik ağ, kullanıcı tarafından akıllı taramalarda uygulandığında, araştırmada bulunan sonuç verilerine, akıllı linkler vererek, yeni verilere bağlantı sağlamaktadır. (Tombre, 1997)

Robert Oxman, Şekil 5.3'teki şemada gösterildiği gibi, tasarım çizimleri ile ilgili yöntemlere ve davranışlara ait çalışmalarla görsel sorgulama konulu bir araştırma yürütmektedir. Tasarım eskizlerine örnek çalışmalarını empoze ederek, tasarımda görsel sorgulamaya ait fenomeni tarif etmek için bir taksonomi tanımlamıştır. Görsel sorgulamanın bu yapısı, tasarım davranışının gözlemlenmesi ve kaydedilmesinde tahsis edilebilecek bir grafik kod için temel olacaktır.



Şekil 5.3. Tasarımda Görsel Sorgulama (**Koutamanis, 1997**)

Tasarım süreci, fonksiyonel terimlerle verilen bir tasarım probleminin, yapısal terimlerle verilen bir tasarım tanımına dönüştürülmesini içermektedir. Deneyim, tasarım sürecinde önemli bir yere sahiptir. Deneyimli tasarımcılar güncel durumları çözmek için, deneyimsel bilgiye dayanan sorgulama süreçleri için geliştirilen modelleri kullanarak, daha önceki tasarımlara ulaşmaktadırlar. Tasarımın rolü, belirli problemlere özgün sonuçlar geliştirmektir.

Gerhard Schmitt, mimarının ana işinin tasarımın değiştirilmesi olduğunu savunarak, yeni teknolojilerin tanıtımı ile ilgili gerçekçi bir yaklaşım üzerinde çalışmaktadır. Görsel veritabanları, mimarlık tarihinde kullanılan çizimler, fotoğraflar ve günümüzde bilgisayar tabanlı imgeler uzun bir geçmişe sahiptirler. Geçmişte mimarlar bu veritabanlarının geleneksel kullanımlarına güvendiklerinden, kitap ve bilgisayar uygulamaları fonksiyona hizmet edecek biçimde olmasına rağmen, mimari tasarım üzerinde yoğun etkisi bulunamamaktaydı. Schmitt, görsel veritabanlarının geleneksel pasif rollerinin aktif , bilgi sunumunun mimari örnekleri kapsayacak hale gelmesini ve bu örneklerle olan etkileşimin sanal bir tasarım ortamında gerçekleşmesi gerektiğini savunmaktadır.

Veritabanı kompleks, karmaşık bir gerçeklik modelidir. Tasarım veritabanı, sadece belirtilen modeli saklayan ve bu veritabanına ilkel erişimleri sağlayan bir sistem değil, aynı zamanda gereken tüm duraksama ve gerileme adımlarına da cevap verebilmektedir. Tasarım amacından daha çok özellikle iş amaçlı veritabanlarında bu özellik, birçok problemin zorluğunu aza indirmektedir. Aynı zamanda, veritabanı

kullanımında, anlamsal bütünlük ve sabit devamlılık da çok büyük bir önem taşımaktadır. Anlamsal bütünlük hataları, cisimlerin tanımlanmış sınırlamalarının kullanıcı tarafında ihlalidir. Sabit devamlılık hataları, birden fazla kullanıcının aynı nesneye erişiminde, benzer sınırlamaları ihlalidir. Veritabanı tasarımının doğasında gerekli olan en önemli unsurlardan biri de arayüz tasarımıdır. **(Laufe, 1978)**

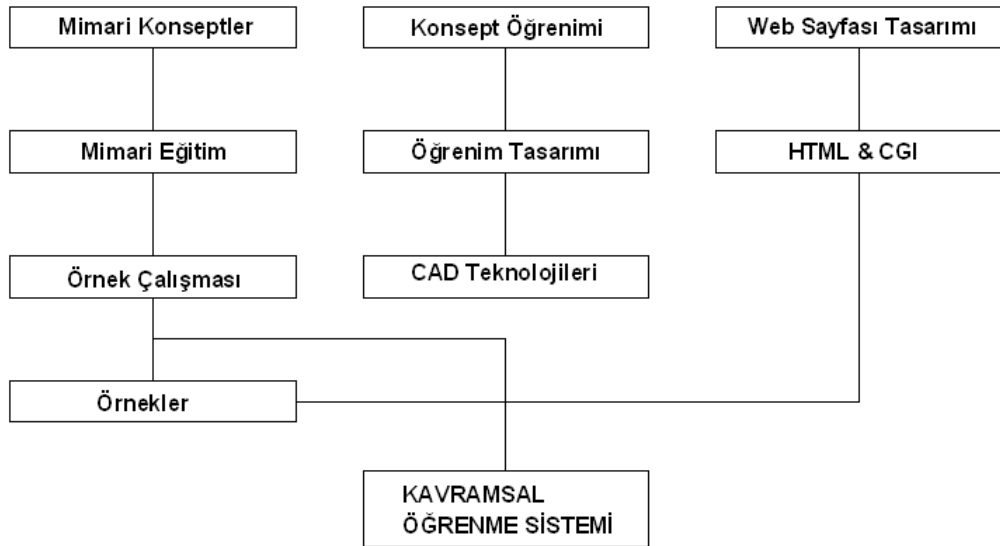
5.3. Tasarım Öğreniminde Veritabanı Uygulamaları

Tasarım gibi iyi tanımlanmış problemleri içeren bir alanda, önceki örneklerden elde edilen bilgilerin yeni durumların anlaşılabilmesi için kullanımı karmaşık bir süreçtir. Hiçbir problem, kendine ait tüm elemanlara çözüm üretecek biçimde ortaya çıkmaz. Dolayısıyla, aynen korunarak, hiçbir homojen bilgi seti yeni bir örnek için kullanılamaz. Önceki bilginin güncel tasarım problemine cevap verebilmesi için uyarlanması gerekmektedir. Uygun sunumlardan transfer yapılırken, her örnek kendisine refakat eden örneklerin farklı kavramlarında ele alınmalıdır. **(Muller and Pasman, 1996)**

Varolan örnekleri inceleyerek, mimari konseptleri öğrenme, tasarım stüdyolarının ortak aktivitelerindendir. Geleneksel olarak, eğitmen bir tasarım konseptini slayt, fotoğraf, çizim ve sözlü analizler kullanarak sunduğu örneklerle anlatır. Bu yöntem, tasarım bilgisinin, doğrusal bir yola aktarımına dayanır ve zamana bağımlıdır. Bilginin anlaşılmasından çok saklanması ve esaslarını ortaya koymaktadır. **(Park and Miranda, 1997)**

Mimari örnek bilgilerinin sunumu dışında, veritabanındaki örnekleri anlamayı sağlayacak şekilde hazırlanan dijital sunumlar, geleneksel yöntemin dezavantajlarını ortadan kaldırarak, bilgisayar teknolojisi kullanımının yeni avantajlarını ortaya çıkarmaktadır. Mimari tasarımdaki konseptlerin geliştirilmesi ve test edilmesinde grafik görüntüler, yazılar ve raporlar kullanan bilgisayar destekli bir ders sistemi geliştirilmesi ihtiyacı ortaya çıkmıştır. Mimari konsept oluşumunda, mevcut örneklerin yorumlanması ve geçmiş örneklerin incelenmesi tasarım bilgisini sunmanın bir yolu olarak kabul edilmektedir. Mimari konseptleri öğrendikten sonra, öğrencilerin veritabanındaki örnekleri yorumlayarak, özgün tasarım fikirleri geliştirmeye başlayabilir duruma gelmesi hedeflenmektedir. Bir tasarım fikri, bir yapı örneğinin tasarım strüktürünü veya gelişimini oluşturan bir form ya da şekilden

çıkartılabilir. Mimari konsept tasarımı öğreniminin bir yolu, konseptleri ait oldukları koşullardan bağımsız olarak değerlendirerek, başka örneklere uygulamalarına imkan verebilmektir. Öğitmen, verimli bir konsept öğrenimi için, öğrencilerine mimari ve benzer konsept durumları ile ilgili farklı örnekler sunmalıdır. Bilgisayar destekli ders sisteminde, mimari konseptlerin grafik görüntüler ve metinler aracılığıyla sunulması amaçlanmıştır. Bilgisayar destekli sistem, internet bağlantısı ve grafik gösterebilme kabiliyeti olan her bilgisayarla çalışabilir olmakla beraber, sistem verimli konsept öğrenimi karakteristiklerine, erişebilirlik, interaktiflik, esneklik, hızlı geri besleme vb. dayanmaktadır. Güncel kavramsal çalışma sonuçları, bilginin sunum şeklinin, mimari konseptlerin öğretilme ve öğrenilme aşamalarında ne kadar önem taşıdığını göstermektedir. Şekil 5.4'te bilgisayar destekli ders sistemi projesi organizasyonu gösterilmektedir.



Şekil 5.4. Öğrenme Sistemi Organizasyonu (Park and Miranda,1997)

Tasarım alanlarında elektronik medyanın kabulünün artmasıyla, elektronik uygulamaları genişletmede, bilgi çıkışının sağlanması önemli bir nokta haline dönüşmüştür. Spiro Pollais, mimari tasarımda üretim teknolojisi hakkında bilgi sağlayabilen bir veritabanı kullanımını oluşturmuş ve veritabanı uygulamaları için dört sınıf belirlemiştir. Bu veritabanı uygulamaları daha önceki tasarımlara ulaşılmasını sağlayarak, bina bölümlerinin geometrik modellerini oluşturmakta ve uzman bir görüş önermektedir. Bu veritabanları, kendi içinde organize olmuş

tasarımlara imkan verirler ve yakın gelecekte mimari tasarımda önemli bir role sahip olacaklardır.

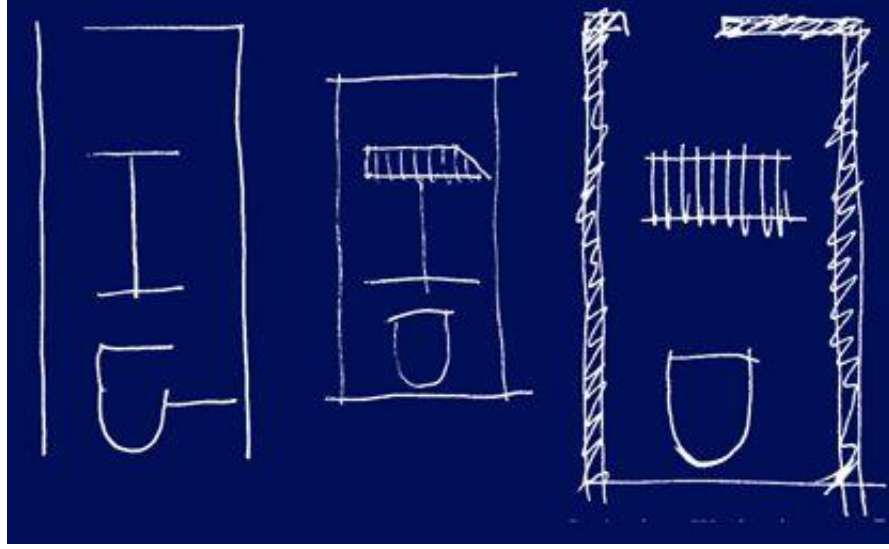
5.4. Mimari Sunumlarda Veritabanı Kullanımı

Bilgisayar destekli mimari tasarımda (CAAD), mimari imgelerin sistematik olarak, bilgisayar ortamına aktarılması, elektronik yayınlarda ve geleneksel kütüphaneler ile slayt koleksiyonlarının online bilgi sistemlerine dönüştürülmesinde yaygın olarak kullanılmaya başlanmıştır. Mimaride görsel veritabanı uygulamaları araştırmasındaki esas, geniş bir spektruma sahip olan yaklaşımlar, amaçlar ve tekniklerin gözden geçirilmesi ve mimari bilgi sistemlerinin gelişim yöntemlerinin incelenmesi olmuştur. Yapma çevrenin tasarımı ve yönetimi yoğun olarak görsel bilgiye ve görsel veritabanlarındaki koleksiyonlarına dayanmaktadır. Görsel veritabanlarının oluşturulması yoğunlaşan araştırma konuları: mimari konseptlerin görsel sunumu ve tasarımın görsel sunumlarda kullanımı olmuştur. Günümüzde ise, CAAD araştırmalarında, tasarım imgesi için tarihsel ve metodolojik açıdan artan bir ilgi oluşmaktadır. Araştırmacılar, belirli mimari fikir ve çözümlerin nasıl ve ne zaman ortaya çıktığını açıklayan dokümantasyona ulaşmak için, tüm mimari arşivleri analiz etmeye başlamışlardır. **(Koutamanis, 1997).**

Görsel mimari sunumların, resmi bir sistem olarak algılanması ve kabul edilmesi, mimari problemlerin anlaşılmasını sağlamaktadır. Birçok mimari tasarım çalışması, mimari imgenin önemini küçümsemektedir. CAAD'in üç boyutlu görsel çalışmaları, mimarların kullandıkları görsel sunumların yaygınlaşmasına katkıda bulunmuştur. İnsanla yapma çevre arasındaki etkileşim genellikle görsel yöntemlerle gerçekleşmektedir. Dolayısıyla, mimari sunumlarımızın temeli de görseldir ve yapma çevre içerisinde algıladığımız imgelerle ve zihinsel simgelerle yakından ilişkilidir. Mimariye ait zihinsel imgeler, mimarın yapma çevreyle ilgili elde ettiği kişisel deneyimden veya yayınlardan bildiği, varolan tasarımların mimarda bıraktığı izleridir. Mimari araştırma için uygun olan, bilinen imgelerin yeni tasarım problemlerinin çözümünde kullanımlarıdır. Mimar, çoğunlukla yeni tasarımlara ait zihinsel imgeleri, eski tasarımların imgeleriyle ilişkilendirmektedir.

Mimari eğitimden bağımsız olarak, her gün kazanılan deneyimler ve özel teknik bilginin mimari bilgi olmak üzere, mimari bellek iki temel kaynağa sahiptir. Mimari

bilgi, mimari belleğin çalışma biçimine ait esaslı bir deneyime sahiptir. Bir binanın resmi, mimari eğitim görmüş ve görmemiş kişilerde farklı tepkiler oluşturmaktadır. Fotoğrafın yorumlanış biçiminden ve farklı yorumların yakaladığı derinlikten dolayı farklı bakış açıları oluşmaktadır. Bellekteki yapma çevreye ait bilginin organizasyonu, bellek stratejileri için mimari bilgiyi kullanmaktadır. Mimari belleğin, mimari eğitim ve öğretim sırasında ve sonrasında gelişimi dikkate alınmaya değer bir konudur. (Koutamanis, 1997)



Şekil 5.5. Mimari Bellekte Diyagramlarla Sorgulama (Koutamanis, 1997)

Mark Gross, online bilgi sistemlerinin sezgisel veya doğrudan çıkış sağlayabilme ve geri kazanma konularına sağladıkları imkanların arttığını gözlemlediğinden, tasarımlara ait görsel veritabanlarındaki verilere ulaşabilmek için, elle çizilmiş diyagramları sorgulama biçimi olarak kullanılan bir şema tanımlamıştır. Gross, denemeler sonucunda, diyagram tabanlı bir indekslemenin çalışması için, tasarımcıların diyagramlarının yeterli olabileceğini kanıtlamıştır. Görsel bir veritabanındaki elle çizilmiş diyagramları algılayacak bir programla bağlantısını kurarak, tasarımın çalışma prensibini de belirlemiştir. Şekil 5.5'te gösterilen uzaysal ilişkilere ait sembolik tanımlamalara dönüştürülen diyagramlar, sorguya uyan imgeleri arayan veritabanına ait diyagram indekslerine uydurulmaktadır.

Varolan bilgileri bilgisayar ortamına transfer ederek, bu dokümanların kompakt bir biçimde korunmaları ve yayılmaları amaçlanmaktadır. Bilgisayar ortamındaki dokümanlar, orijinal dokümanlarda bulunan bilginin kullanılabilirliğini geliştiren ve

yaygınlaştıran, gelişmiş ve tutarlı sistemlerde yapılandırılmıştır. Mimari sunumlardaki oluşumlar ve ilişkiler, yapılmış bir formun birçok özelliğinin tanımlanması için yeterli olabilmektedir.

Kullanıcı ve bilgisayar, yapının mekansal, yapısal ve fonksiyonel özelliklerinin tanımlarını geliştirebilmektedir. Bilgisayar bilgi sistemine, bir sorgulama olarak verilen tanım verisinin uygun yanıtı getirebileceğinin garantisi yoktur. Semantik farklılıklardan oluşabilecek zayıf geri kazanım performansı, kullanılan sözcük ve deyimlerin kontrol edilmesine dayanan bir araştırma ile çözülebilir. Mimari terim ve mimari konseptlerin genel taksonomisine göre bir sorgu tanımlanır ve bu taksonomi bilgi sisteminin içeriği ile bağlanır. Bilgi sistemlerinin herhangi bir alanda kullanımı, varolan doküman arşivlerinin, uygun bir formata dönüştürülerek, bilgisayar sistemine aktarılması sorununu ortaya çıkarmaktadır. Mimarlar tarafından yapılan doküman analizi, farklı türde birçok çizimin kağıt üzerinde bulunması açısından, mimarlık, mühendislik gibi teknik alanlarda güçlü bir araç olarak görülmektedir.

5.5. İnternet ve İnternet Teknolojilerinde Veritabanı Uygulamaları

İnternet teknolojileri, inşaat dokümanlarını, tasarım aşamasından sonra üretimini ve servis hizmetlerini desteklemek amacıyla kullanılmaktadır. Gündeme gelmekte olan bilgi sistemleri bağlamında, sistem dokümanlarına ait bilgi içeriği ve sistem yönetim süreçlerine ait yürütülen çalışmanın, yeni bilgi teknoloji sistemlerini uygulayan sistem yöneticilerine yardımcı olması hedeflenmiştir. Bina sistemlerinin yaşam dönemlerinde kullanılan dokümanların analizi, tasarım çizimleri ve as-built çizimler dışındaki dokümanların karakterize edilmesine götürmektedir. Sistem yönetim süreçlerinin analizinden, bu süreçlerdeki sistem dokümanları için farklı roller tanımlanmıştır. Bina sistem dokümantasyonu kaynak, girdi veya çıktı olarak kullanılabilir. Bilgilerin çizimlerden nasıl elde edilebileceğini, veritabanlarıyla bütünleştirilme biçimlerini ve sanal ortam arayüzleri ile nasıl kullanılacaklarını göstermek için yazılımlar hazırlanmıştır. Uygulama dokümanlarının çok sayıda bilgi tipinden oluştuğu ve veritabanları, CAD çizimleri ve hypertext gibi farklı bilgi araçlarına ihtiyaç duyduğu bilinmelidir. Özellikle, intranet teknolojileri, sistem yöneticileri ve kullanıcılar için bilginin ulaşılabilirliğini sağlamaktadır.

İnternet teknolojileri, ticari uygulamalar olarak hızla yaygınlaşmakta olsa da, bunların mimarlar ve sistem operatörleri tarafından binanın yaşam dönemini desteklemek için nasıl daha efektif olarak kullanılabileceği tam olarak netlik kazanmamıştır. Özellikle, internet teknolojilerinin özel bilgisayar ağlarında intranet olarak kullanılması yaygınlaşmaya başlamıştır. Bu olanakların daha iyi anlaşılması için, güncel yöntemler ve sistem yönetimi için gerekli olan bilgi üzerinde çalışılmış ve internet teknolojilerinin bu ihtiyaçları nasıl karşılayacağı gözden geçirilmiştir intranet yaklaşımı, pratik ve uygun oldukları kanıtlanmış olan tarayıcılar için anahtar teknolojileri tahsis etmektedir. Yaygın olarak kullanılan sanal doku (WEB) teknolojisi, farklı grafik ve yazı bilgisine destek verebilir ve geleneksel bina sistem yönetim sistemlerine entegre edilebilir. Bir intranet, sistem yönetiminde gerekli olan bilginin sunumuna adapte edilebilir.

5.6. Görsel Mimari Tasarımda Veritabanı Projeleri Örnekleri

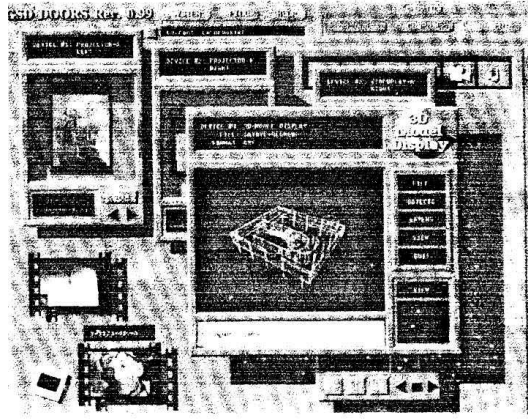
5.6.1. DOORS – Design Oriented Online Resources System Projesi

Harvard Üniversitesi Tasarım Bölümü'nde geliştirilmekte olan DOORS Projesi, tarama, kompozisyon oluşturma ve sunum olmak üzere, üç işlemci model üzerine kurulmuştur. Tarama modülünde; kullanıcı, saklama ve dip not koyma gibi araçlar ile desteklenirken, kompozisyon oluşturma modülünde; ilişkilendirme, materyallerin vurgulanması ve niteliklerin düzenlenmesi gibi çoğul ortam yönetim araçlarıyla desteklenmekte, sunum bölümünde ise; stüdyodan, ofisten veya sınıftan yapılandırılmış sunumlar gösterilebilmektedir. **(Harvard, 1993)**

DOORS Projesi'nin hedefleri şöyle sıralanabilir.

- Tasarıma referans olabilecek, değişik türlerdeki, slayt kütüphaneleri, fotoğraflar, bilgisayar destekli tasarımlar, çoklu ortam projeleri gibi görsel materyallere erişim sağlamak,
- Tasarım çalışmaları ve araştırmalarında oluşturulan elektronik projelerin okul yerel ağı sayesinde alınması, saklanması ve ulaşılabilmesi için bir arayüz oluşturmak,

- Kullanıcılara esnek bir sistem sağlayarak; üzerinde çalışmak ve düzenlemeler yapmak üzere görsel enformasyon sağlamak, ağdaki görüntü ve modellerin analizini yapmak, farklı formlardaki tasarım belgelerini karşılaştırmak ve ilişkilendirmek, stüdyolarda, çalışma alanlarında ve ofislerde, bilgisayar vasıtasıyla, dersler için sunum hazırlamak. (Sklar, 1997)



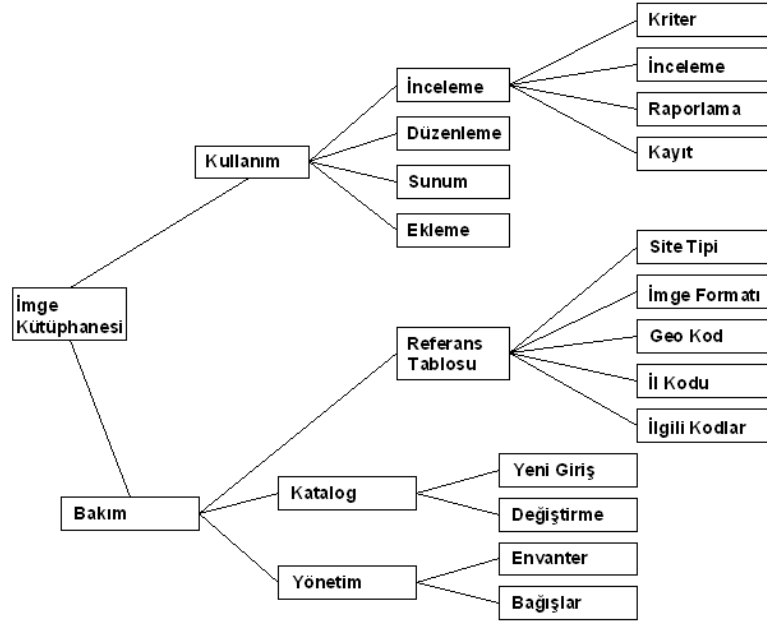
Şekil 5.6. “DOORS” Projesi Oluşturma Modülü Arayüzü. (Altan, 2002)

Hinda Sklar, Şekil 5.6’da arayüzü gösterilen “DOORS” Projesi, Harvard Tasarım Okulu bilgisayar ağı içerisinde kullanılacak olan, tasarım referans materyallerini gösteren bir sistem olarak tanımlamaktadır.

Projenin amaçları:

- Fakülte çalışanları ve öğrencilerin, tasarım pedagojisi içerisinde kullanılan kütüphane slaytları, fotoğraflar, haritalar, coğrafi bilgi sistem verisi, bilgisayarda oluşturulmuş animasyonlar ve sesli dijital video çalışmaları gibi farklı görsel referans malzemelerine ulaşmalarını sağlamak,
- Elektronik ortamda yaratılmış projeleri toparlamak, depolamak ve okulun çoklu-platform ağında ulaşılabilir kılmak,
- Kullanıcılarına görsel bilgiyi çalışma ve değiştirme amaçlı inceleme olanağı sağlayan, esnek bir sistem yaratmak, imge ve modelleri analiz etmek, tasarım dokümantasyonunu farklı formatlarda karşılaştırmak ve aralarında bağlantı kurdurmak, ofisler, sınıflar ve çalışma stüdyolarındaki bilgisayarlardan ders ve sunumlar gerçekleştirmek.

Fakülte, öğrenciler ve kütüphane personeli ile yapılan dikkatli çalışmalar sonucunda, Şekil 5.7’de strüktür şeması gösterilen "DOORS" projesinin gelişimi, kullanıcı arayüzü ve veritabanının indekslenmesi üzerinde yoğunlaşmıştır.



Şekil 5.7. "DOORS" Projesi Strüktürü (Koutamanis, 1997)

5.6.2. AVIDOR Projesi

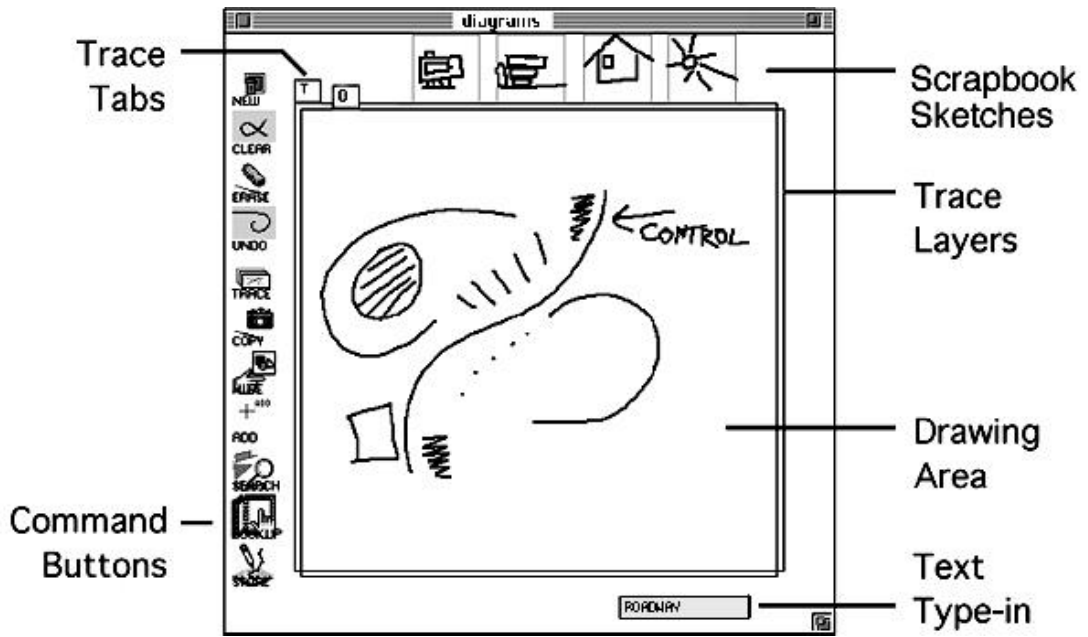
Columbia Üniversitesi'nde yayınlanan, tasarımcılar ve araştırmacılar tarafından oluşturulan AVIDOR Projesi, yaklaşık 41.000 mimari çizim içermektedir. AVIDOR Projesi, kullanıcı etkileşimli bir ortamda çalışmakta ve mimari tasarımcıya seçtiği imajla ilgili, veritabanında eşleşen tüm resimleri gösteren bir arama motoru sunmaktadır. Genişleyen kütüphanesine rağmen, sistem sadece yerel ağ üzerinde çalıştığından, verimli bir şekilde kullanılabilir.

5.6.3. The Electronic Cocktail Napkin Projesi

The Electronic Cocktail Napkin Programı, elle çizilen eskiz ve diyagramların oluşturulduğu ve yönetildiği bir tasarım ortamıdır. Program, elle çizilen figürleri depolama ve tanımlama dışında, grafik arama, katman yönetimi ve çizim alanının birden çok kullanıcıya paylaşılması gibi işlevleri de yerine getirmektedir. Veritabanı şeması tamamen mimari diyagramların gövdesine bağlı olarak

kurgulanmış, Macintosh Common Lisp ile yazılmış, tüm diyagramları tamamen görsel olan The Electronic Cocktail Napkin Projesi, interaktif olarak, elle çizilmiş basit eskiz diyagramları, elle çizilmiş diğer diyagramlarla eşleştirilerek sınıflandırmakta ve sembolik ilişkilere dönüştürmektedir.

Kullanıcı, özel bir bilgisayar kalemi kullanarak, basitçe bir tabletin üzerine eskizler çizdikçe, programın çizim penceresinde şekiller belirlemektedir. Program, kullanıcının şekillerini harf, biçim veya çizgi olarak tanımlamaya çalışmaktadır. Program, aynı zamanda ağaç veya balon diyagramları gibi basit, belirli biçimleri tanımlama özelliği göstermektedir.

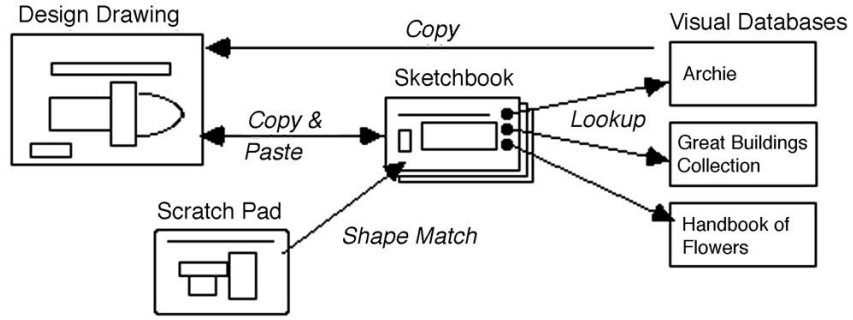


Şekil 5.8. The Electronic Cocktail Napkin Projesi Arayüzü (Gross,1994)

Şekil 5.8’de gösterilen “The Electronic Cocktail Napkin” programı arayüzünde, eskiz penceresi, komut butonları, kullanıcıyla yazı ile etkileşim kurulan mesaj kutusu, çizim penceresindeki yaprakların seçimi için bölümler (tablar) ve daha önce yapılan diyagramları gösteren bir katalog bulunmaktadır.

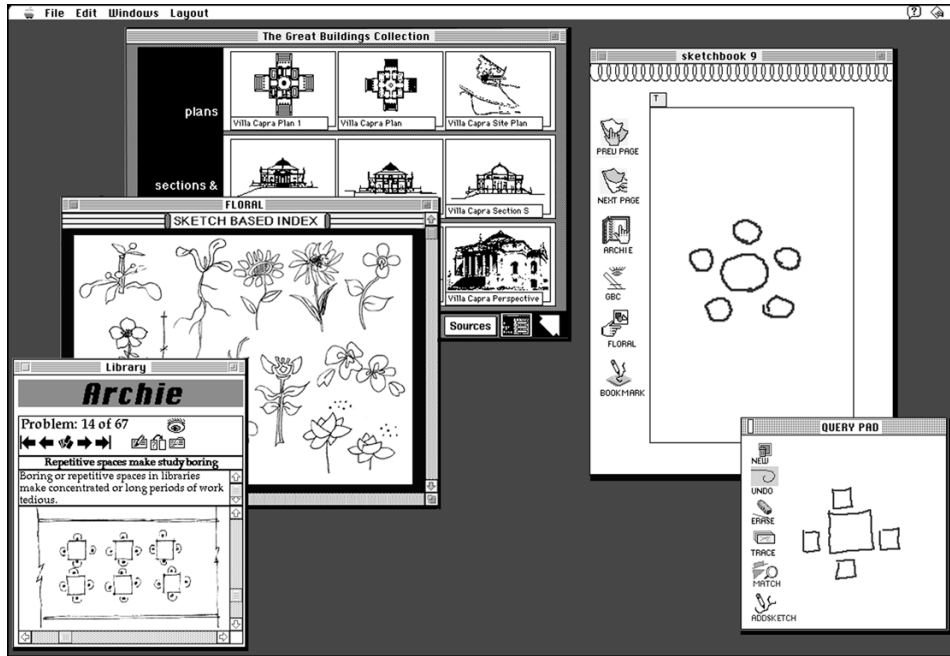
Benzer çizim tanımlama programlarından farklı olarak, “The Electronic Cocktail Napkin” programına, tasarımcının birçok diyagramı bir araya toplayabileceği ve farklı veritabanlarındaki öğelerle ilişkilendirilmesini sağlayan “sketchbook” tasarım arayüzü eklenmiştir. Tasarımcı çizimler ve eskiz kitabı (sketchbook) arasında geçiş yaparak, sayfaları kopyalayabilmektedir. Şekil 5.9’te gösterilen çizim tanımlama

sistemi bölümlerinden biri olan çizim pedi (scratch pad) diyagramlar oluşturarak, sorgulama başlatılabilmektedir.



Şekil 5.9. The Electronic Cocktail Napkin Çizim Tanımlama Sistemi (Gross,1994)

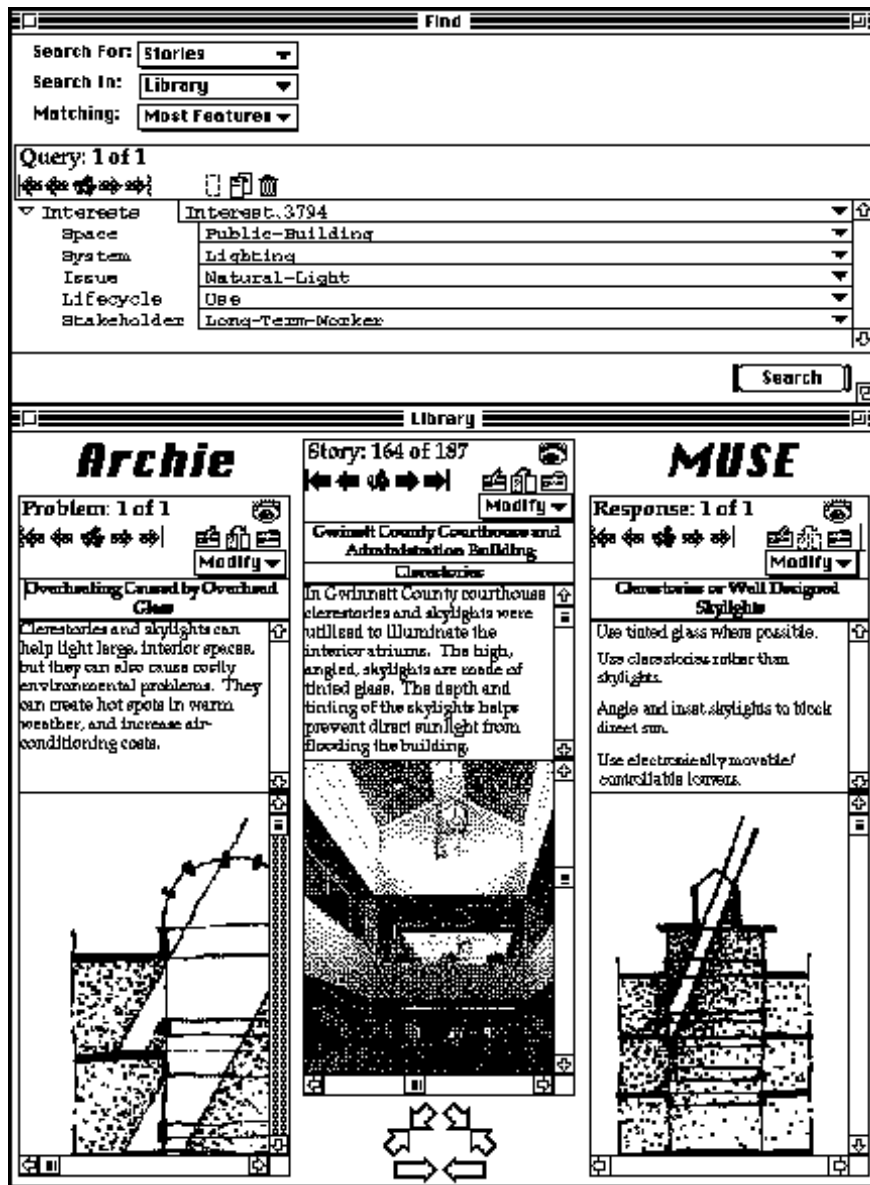
Program, tanımlanan diyagram sayfalarında en uygun ve benzer olanını seçerek, ilgili öğeler için gerekli veritabanı bağlantılarını kurmaktadır. Tek bir sorgulama ile Şekil 5.10'da gösterildiği gibi, Great Buildings Collection programı ile önceki mimari örnekler, Archie programı ile tasarım hikayesi ve Natural Forms Veritabanındaki sonuç sayfası bir arada gösterilmektedir.



Şekil 5.10. The Electronic Cocktail Napkin Sonuç Sayfası Arayüzü (Gross,1994)

5.6.4. Archie Projesi

Archie, tasarıma yardım sağlaması, kavramsal tasarımın öğreniminde desteklemesi amaçlanan bir bilgisayar aracıdır. Macintosh Common Lisp ile yazılmış olup, efektif problem çözümü, uygun geçmiş deneyimler üzerinden adapte etmek veya geri almak üzerine kurulu, durum tabanlı akıl yürütme teknolojisi olarak adlandırılan teoriye dayandırılmıştır. Şekil 5.11’de arayüzü gösterilen Archie projesi, öğrencilerin yapıların değerlendirilmiş durumlarına erişim sağlamaktadır. Archie programındaki durumlar, tanımlar, problemler, tasarım yanıtları ve tasarım hikayeleri olarak yapılandırılmaktadır.



Şekil 5.11. Archie Projesi’nde Tasarım Hikayeleri Arayüzü (Gross,1994)

Tanımlar, mimari kitap ve dergilerdeki, plan, kesit, eskiz gibi genel mimari sunumlardır. Archie Projesi’ndeki problemler, çözülmeyi bekleyen ikilemler olarak tanımlanmaktadır. Öğrencilerin, tasarım kararlarını verirken, çelişkili gereksinimleri çözümlmek için çaba harcaması gerektiğini bilmesinin önemi vurgulanmaktadır. Tasarım yanıtları, tasarımcının amacına ulaşması için değerlendirmesi gereken stratejilerdir. Tasarım hikayeleri, belirli binaların tasarımındaki problemlerin çözümünün kısa tanımıdır.

5.6.5. Greatest Hits Slide Library Projesi

Tasarımcılar için gerekli çoğu bilgi kaynağı, anahtar kelimelerle tanımlanmakta, indekslenmektedir. Görsel Veritabanları (Slayt Kütüphaneleri), veritabanındaki her bir yapı için, yapının ismi, mimarı, tarihi ve yeri gibi başlıklar altında anahtar kelimeler belirlenmektedir.



Şekil 5.12. “Greatest Hits” Slayt Kütüphanesi Arayüzü (Gross,1994)

Şekil 5.12’de arayüzü gösterilen “Greatest Hits” Slayt Kütüphanesi Programı, tanınmış yapıların görsel koleksiyonu olarak, File Maker Pro yazılımı kullanılarak, yapılandırılan bir veritabanı programıdır. Her bir kayıt, yapının resmini, yapıyı tanımlayan yazılı bilgiyi, mimar, tarih ve yapı ile ilgili kısa anlatımı içermektedir.

6. GÖRSEL MİMARİ VERİTABANI İNTERNET UYGULAMALARI

Mimarlık literatüründe iz bırakmış mimarlık veritabanı uygulamalarını inceledikten sonra, yurtdışında hazırlanan internet sayfalarında yayınlanan görsel mimari veritabanı uygulamalarını inceledik. Günümüzde geleneksel araştırma yöntemleri geride bırakılıp, dünyanın farklı yerlerindeki gerekli bilgiye, eş zamanlı erişim önem kazanmıştır. Mimarlık, tasarım, görsellik ve veri ulaşımındaki yenilik ve gelişmeleri irdeleyebilmek için, aşağıda farklı görsel mimari veritabanları, slayt arşivleri, arayüzleri ve arama mantıkları karşılaştırılmaktadır.

6.1. “SPIRO” Projesi / California Üniversitesi

University of California, Berkeley
ARCHITECTURE VISUAL RESOURCES LIBRARY

S * P * I * R * O Query Form

HELP: [Searching SPIRO](#) [Look-up Tables](#) [Input Fields](#) [% Truncation](#) [Sample Searches](#)

[Period or Topic](#)
20th Century
19th Century
17 & 18th C
North American Colonial: General

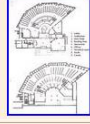
[Personal or Group Name](#) [Subject or Object Type](#)
[Building or Object Title](#) (and subject)
[Location](#) [Source Abbreviation](#)
[Image ID](#)

Result format ☒ Thumbnail w/description
☐ Thumbnail images only

Date Range to
BCE ☐ BCE ☐

• Send comments on images & records in SPIRO to Architecture Visual Resources Librarian Maryly Snow
slides@socrates.berkeley.edu.
• For information on licensing SPIRO images

Şekil 6.1. “SPIRO” Görsel Mimari Veritabanı Sorgulama Arayüzü

SPIRO Query Results		
Your query for [Period:20th Century, Building/Object:library] on Sun Dec 26 20:35:45 US/Pacific 2004 returned 22 records. Page 1 of 2. Click on a thumbnail to view a more detailed record.		
AALTO, Alvar. Library. Mount Angel Benedictine College, Saint Benedict, OR. Floor plan, initial design, vw 1964. Canty, p.24. 98-047-001 © UCB AVRL tW24 A112 Ben1b.1 	AALTO, Alvar. Library. Mount Angel Benedictine College, Saint Benedict, OR, 1965-1970. Floor plan, final version: ground, lower level (t). Canty, p.26. 98-047-002 © UCB AVRL tW24 A112 Ben1b.2 	AALTO, Alvar. Library. Mount Angel Benedictine College, Saint Benedict, OR, 1965-1970. Aerial vw, semicircular library in center. Canty, p.6. 93-109-001 © UCB AVRL tW24A112 Ben3a.1 
AALTO, Alvar. Library. Mount Angel Benedictine College, Saint Benedict, OR, 1970. Exterior, rear. AR, May 1971. 01-305-001 © UCB AVRL tW24 A112 Ben3b.1	AALTO, Alvar. Library. Mount Angel Benedictine College, Saint Benedict, OR, 1970. Interior, book collection & daylight. LBL dupe, . 01-305-002 © UCB AVRL tW24 A112 Ben6.1	AALTO, Alvar. Library. Mount Angel Benedictine College, Saint Benedict, OR, 1970. Interior, reference or circulation desk. Treib slide. 01-305-003 © UCB AVRL tW24 A112 Ben6.2

Şekil 6.2. “SPIRO” Görsel Mimari Veritabanı Sonuç Arayüzü 1

“http://www.mip.berkeley.edu/query_forms/browse_spiro_form.html” web adresinde bulunan, Şekil 6.1’de gösterilen “SPIRO” olarak adlandırılan, California Üniversitesi’nde Maryly Snow yönetiminde yürütülen görsel mimari slayt arşivinin sorgulama sayfası arayüzü HTML kullanılarak, oldukça sade ve işlevsel olarak tasarlanmıştır. Yapı ismi, yapının bulunduğu yer, yapının dönemi ve dia ID numarası gibi kategorilerde istediğiniz kelimeyi yazıp, JavaScript dili yardımıyla arama yapma olanağı sağlayan görsel bir veritabanı kurulmuştur. Kullanıcının aradığı anahtar kelimelerle ilişkilendirilmiş dia ve dia bilgileri Şekil 6.2’de sonuç sayfasındaki gibi gösterilmektedir. Herhangi bir dianın üzerine tıklandığında, o dia ile ilgili detaylı bilgilerin verildiği, Şekil 6.3’te görülen detaylı bir sonuç sayfası açılmaktadır.

SPIRO Image and Record (scroll down for complete record)	
	
Call Number tW24 A112 Ben6.1	
Image ID 01-305-002	
Period 20th Century	
Subperiod Finland	
Location Saint Benedict, Oregon	
Object Name Library, Mount Angel Benedictine College	
Date 1970	
View Description Interior book collection & daylight	
Personal/Group Names Aalto, Alvar -- architect	
Subjects abbeyes libraries (buildings) Modernist Finnish bookshelves	

Şekil 6.3. “SPIRO” Görsel Mimari Veritabanı Sonuç Arayüzü 2

6.2. “AVRL” Projesi / California Üniversitesi



UC BERKELEY LIBRARY | HOME | SEARCH

ARCHITECTURE VISUAL RESOURCES LIBRARY

Revised October 2003 Author: [Marvly Show](#)

College of Environmental Design
Department of Architecture

232 Wurster Hall #1800 (mailing address)
494 Wurster Hall (physical location)

Berkeley, California 94720-1800
telephone: 510.642-3439
fax: 510.642-8655 fax
E-mail: slides@socrates.berkeley.edu

About the Visual Resources Library

[AVRL Policy Pages](#)
[Collection](#)
[Access to the Collection](#)
[Library Services](#)
[Interlibrary Loan Policy](#)
[Hours](#)
[Staff](#)
[Architecture Faculty Handbook](#)

6.4. AVRL (Architecture Visual Resources Library) Giriş Sayfası

College of Environmental Design
University of California, Berkeley

Architecture Shop
Architecture Visual Resources Library
Building Science Laboratory
Computing
Environmental Design Archives
Environmental Design Library

News & Events
Alumni/Friends
People
Administration
Resources
Research
Programs
Admissions



*Harold Stump
World Architecture
Slide
Collection*

- The Architecture Visual Resources Library

- [About Harold Stump](#)
- [Collection Description](#) [[Countries](#) | [IMAGES™ \(flash\)](#)]
- [Use Guidelines](#)
- [Use Fees and Request Form](#)
- [Harold Stump Architectural Foundation](#)

Further Information:

6.5. AVRL (Architecture Visual Resources Library) İndeks Sayfası

Şekil 6.4’de ana sayfası gösterilen, “<http://www.lib.berkeley.edu/>” web adresindeki “AVRL” (Architecture Visual Resources Library) görsel mimari veritabanı sayfası California Üniversitesi Mimarlık Fakültesi tarafından hazırlanmış ve internet ortamında yayınlanmıştır. Arama sayfaları üyelik sistemi üzerinden çalıştırılan, sanal

kütüphanede bulunan mimari yapıların slaytları arasında, farklı kategorilerde arama yapılabilme olanağı sağlayan bir internet ağı kurulmuştur. Şekil 6.5'te indeks sayfası arayüzü gösterilen görsel veritabanı, oldukça geniş ve kapsamlı bir arşiv olmasıyla birlikte, 57.000 dijital resim, 225.000 bilgisayar ortamında slayt ve 20.000 dijital fotoğraf içermektedir.

6.3. “The Great Buildings Collection” Projesi



6.6. “The Great Buildings Collection” Projesi Giriş Sayfası

Şekil 6.6’te giriş sayfası gösterilen, “http://www.greatbuildings.com/” web adresindeki görsel mimari veritabanı sayfası, dünyanın önde gelen online mimarlık kütüphanelerinin başında gelmektedir. Binlerce yapı ve yüzlerce mimar ile ilgili tarihsel doküman ve bibliyografya, veritabanındaki yapıların 3D modelleri, dijital fotoğrafları, mimari çizimleri, tanınmış tasarımcılar ve tüm strüktür (yapım) şekilleri hakkında açıklamalar içermektedir. Mimarlık tarihinin ilk yapılarından, günümüzdeki en son yapıya kadar tüm mimari örnekleri, tasarımcıları, fikirleri ve akımları mimarlara ve ilgili kullanıcılara kullanıcı dostu arayüzüyle aktarmaya çalışan kapsamlı görsel mimari bir veritabanı çalışmasıdır. Şekil 6.7’te gösterilen basit arama sayfasında, sadece belirli kategorilerde arama yapılabilir. Sadece yapı, mimar veya bulunduğu yer ismi yazılırken arama yapılan bu sayfa dışında, Şekil 6.8’de gösterilen detaylı sorgulama (advanced search) sayfasında ise, yapı ismi,

mimarı ve yeri dışında tarihi, işlevi, strüktürü, mimari stili gibi birçok konu başlığıyla arama yapılabilir. Kullanıcının aradığı kriterlere uygun yapının yazılı bilgileri ve dijital ortamdaki tüm verileri, Şekil 6.8’de gösterilen sorgulama sonuç sayfasında detaylı olarak gösterilmektedir.

Welcome to architecture online, the leading architecture reference site on the web, **GreatBuildings.com**. This gateway to architecture around the world and across history documents a thousand buildings and hundreds of leading architects, with 3D models, photographic images and architectural drawings, commentaries, bibliographies, web links, and more, for famous designers and structures of all kinds. For up-to-the-moment coverage of the latest buildings, designers, ideas, and trends, GreatBuildings.com is richly cross-linked with [ArchitectureWeek](#), the leading architecture magazine online.

Interlinked with our friends at [RIBA](#) for one-click searching of the world's largest [architecture library catalog](#).

[ARCHITECTUREWEEK](#) — [PICTURE INDEX](#) — [POSTERS](#) — [GREATEST HITS](#) — [CD-ROM](#)

[Architecture Books](#) | [3D Models](#) | [Architectural Types](#) | [Advanced Search](#)

Name of Building Search for Building

Name of Architect Search for Architect

Name of Place Search for Places...

[ARCHWEEK](#) | [DESIGNCOMMUNITY](#) | [DISCUSSION](#) | [DESIGNWORKSHOP](#) | [FREE 3D](#) | [CAD OUTPOST](#) | [GALLERY](#) | [SEARCH](#)

Mixed Reactions to WTC Memorial Plans — [IOL 2004.1217](#)

Lord Foster: The World's Favourite Architect — [Independent Digital](#)

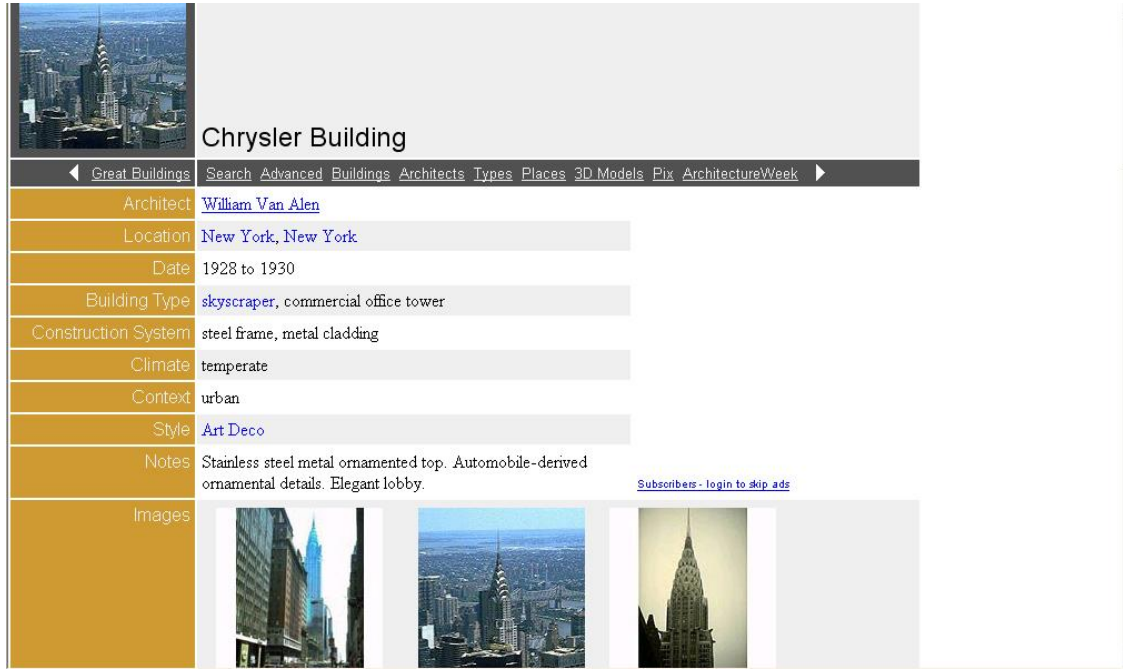
 [Send this to a friend](#) | [Linking](#) | [Contributing](#) | [Privacy](#) | [Media Kit](#) | [Photo Licensing](#) | [Comments](#) 

<http://www.GreatBuildings.com/gbc.html> Server Two © 1997-2004 [Artifice, Inc.](#) - All Rights Reserved.

6.7. “The Great Buildings Collection” Projesi Basit (Simple) Sorgulama Sayfası

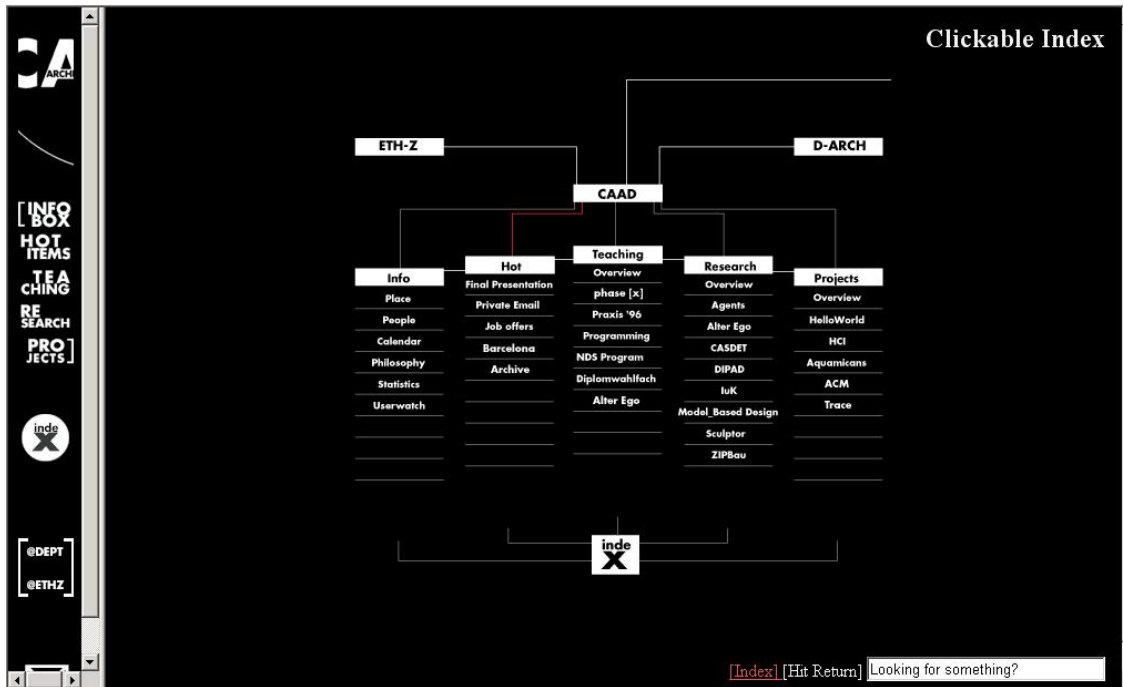
Building Name	☐	
Architect	☐	
Location	☐	
Date	☐	to
Building Type	☐	
Construction System	☐	
Climate	☐	
Context	☐	
Style	☐	
Notes	☐	
Images	☐	
Drawings	☐	
3D Model	☐	
Discussion	☐	
Resources	☐	

6.8. “The Great Buildings Collection” Projesi Detaylı (Advanced Search) Sorgulama Sayfası



6.9. “The Great Buildings Collection” Projesi Sonuç Sayfası

6.4. “Architecture & CAAD, ETHZ” Excite Search Projesi



Şekil 6.10. “Architecture & CAAD, ETHZ” Projesi İndeks Sayfası

Şekil 6.10’de indeks sayfası gösterilen, “http://caad.arch.ethz.ch/main.html” web adresinde bulunan Zürih Üniversitesi Mimarlık Fakültesi (Darch) CAAD sayfasında, bilgisayar ortamına aktarılmış sanal ve dijital veriler arasında, Şekil 6.11’de gösterilen arama sayfasındaki gibi belirli bir anahtar kelimeye göre yapılan aram sonucu Şekil 6.12’de ki resimler şeklinde gösterilmektedir.



Şekil 6.11. “Architecture & CAAD, ETHZ” Projesi Arama Sayfası



Şekil 6.12. “Architecture & CAAD, ETHZ” Projesi Sonuç Sayfası

6.5. “The Architectural Index” Projesi



Şekil 6.13. “The Architectural Index” Projesi Indeks Sayfası

Şekil 6.13’de index sayfasın gösterilen, “http://www.archindex.com/” web adresinde bulunan “The Architectural Index” Projesi’nde Şekil 6.14’te gösterilen sorgulama sayfasındaki pull-down menüler ve anahtar kelime girişiyle arama yapılabilir.

Şekil 6.14. “The Architectural Index” Projesi Sorgulama Sayfası

6.6. “Dalhousie Slide Library” Projesi

“http://architectureandplanning.dal.ca/architecture/current/slide_library.shtml” web adresinde bulunan ve Şekil 6.15’de giriş sayfası gösterilen, “Dalhousie Slide Library” Projesi’nde arama yapıldığında Şekil 6.16’da görülen sonuç sayfası arayüzü ile karşılaşılmaktadır.

Dalhousie University
Architecture
Faculty of Architecture and Planning

Prospective Students
Current Students

2004-05 dates
Registration
Continuing courses
School procedures
Academic regulations
Grades and appeals
Year-end reviews
Scholarships
Graduate studies
Course references
Work term information
ASA
Studio
Resource centre
Slide library
Sexton Library
Photocopying
Computer labs
Computer printing

Slide Library

Database for slides and digital images

The database may be accessed by clicking on the link above. The high resolution scans for records which include a thumbnail image are available on the Faculty network. They are organized by the database Record Number, so you must note the record number to find the scans. Open the corresponding folder in Chooser > AppleShare > Imhotep (Main Server) > High Res Scans. These images may be downloaded for academic use.

Access

The slide library is in the Architecture Resource Centre and the hours are posted on the door. If you find it closed during daytime hours, you may ask a staff member in the Architecture office to open it for you.

Collection

The collection includes approximately 24,000 catalogued slides. Instructions for locating slides are posted inside. If you take slides out of the drawers, please do not refile them; instead, leave them in the 'Return slides here' box on the light table.

Şekil 6.15. “Dalhousie Slide Library” Projesi Giriş Sayfası

Form View | imagesarchplan | Home | Help

Previous | Record 2 | Total Records: 25368 | Next

Table View
Search
Show All
Sort

Architect: Wright, Frank Lloyd, 1867-1959
Object / Project Name: Ennis House
Location: Los Angeles (California)
Date: 1923
View 1: Exterior, south side
Keywords: dwellings, houses; 20th Century; cast-in-place concrete, steel rods

20th USA W7 1923 E5 C.2

image

high

Record Number: 00007
oalo: <http://129.173.100.221/images/00007.jpg>

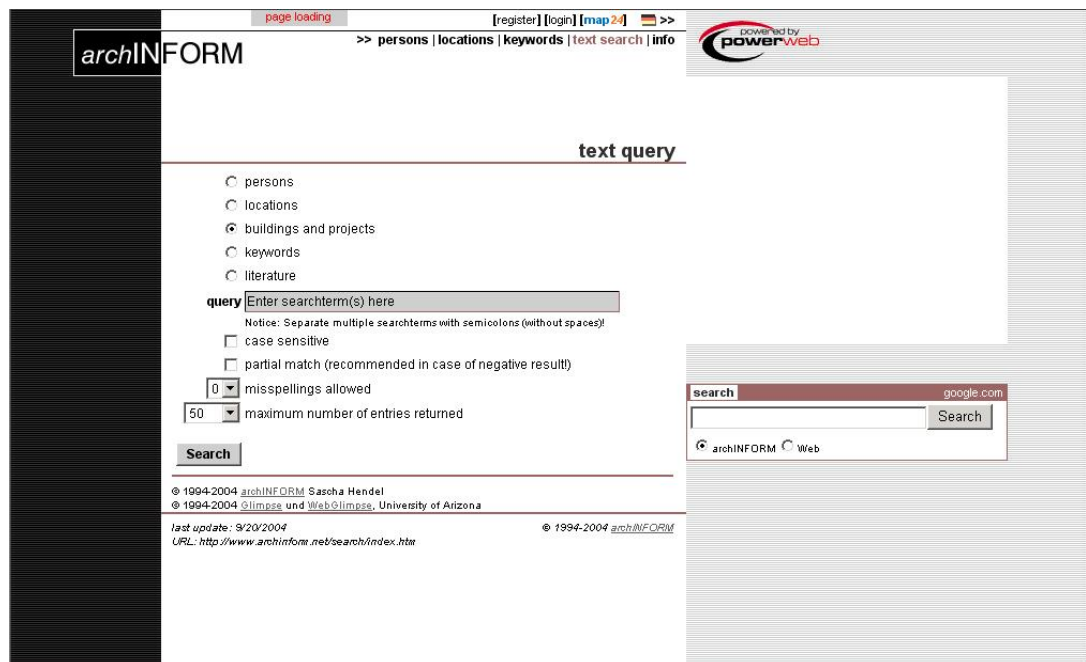
Şekil 6.16. “Dalhousie Slide Library” Projesi Sonuç Sayfası

6.7. “archINFORM” Projesi

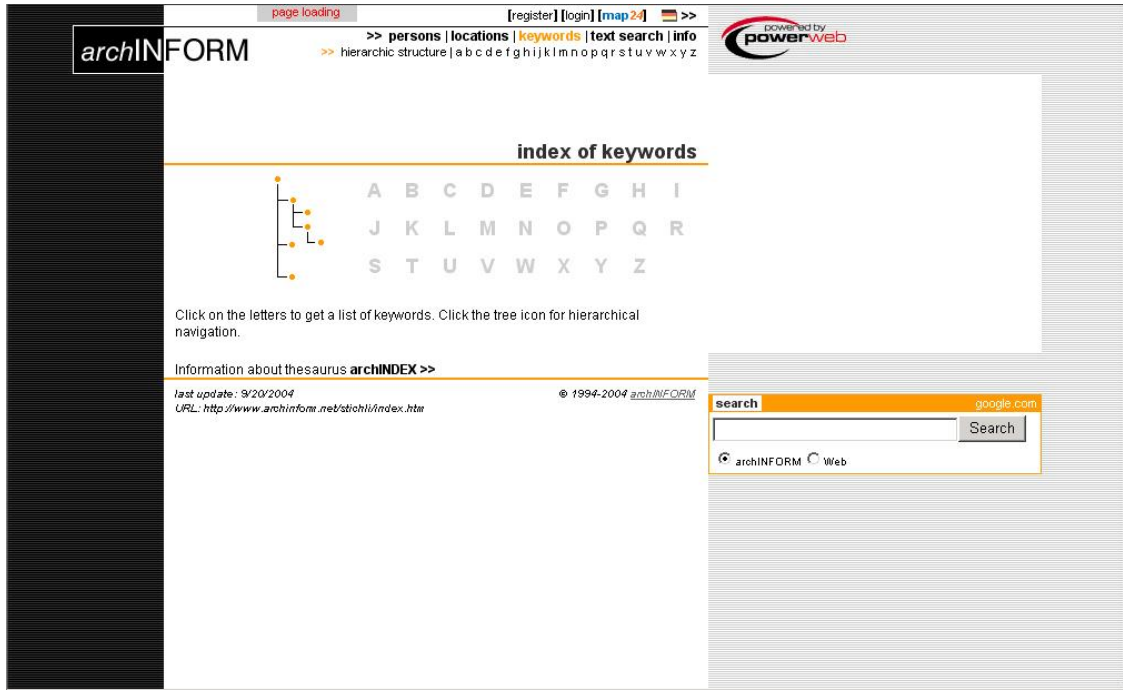


Şekil 6.17. “archINFORM ” Projesi Giriş Sayfası

“http://www.archinform.net/” web adresinde bulunan ve Şekil 6.17’de giriş sayfası gösterilen, “archINFORM” Projesi’nde Şekil 6.18’te gösterilen sorgulama sayfasında anahtar kelime girişiyle veya Şekil 6.19’daki anahtar kelimelerden seçerek arama yapılabilir.



Şekil 6.18. “archINFORM ” Projesi Sorgulama Sayfası



Şekil 6.19. “archINFORM ” Projesi Anahtar Kelime Sayfası

6.8. “SMART – Simple Modular Architecture Research Tool ” Projesi

“<http://smart.embl-heidelberg.de/>” web adresinde bulunan ve Şekil 6.18’te gösterilen sorgulama sayfasında “SMART – Simple Modular Architecture Research Tool ” Projesi’nde anahtar kelime veya dia ID numarası girişiyle arama yapılabilmektedir.

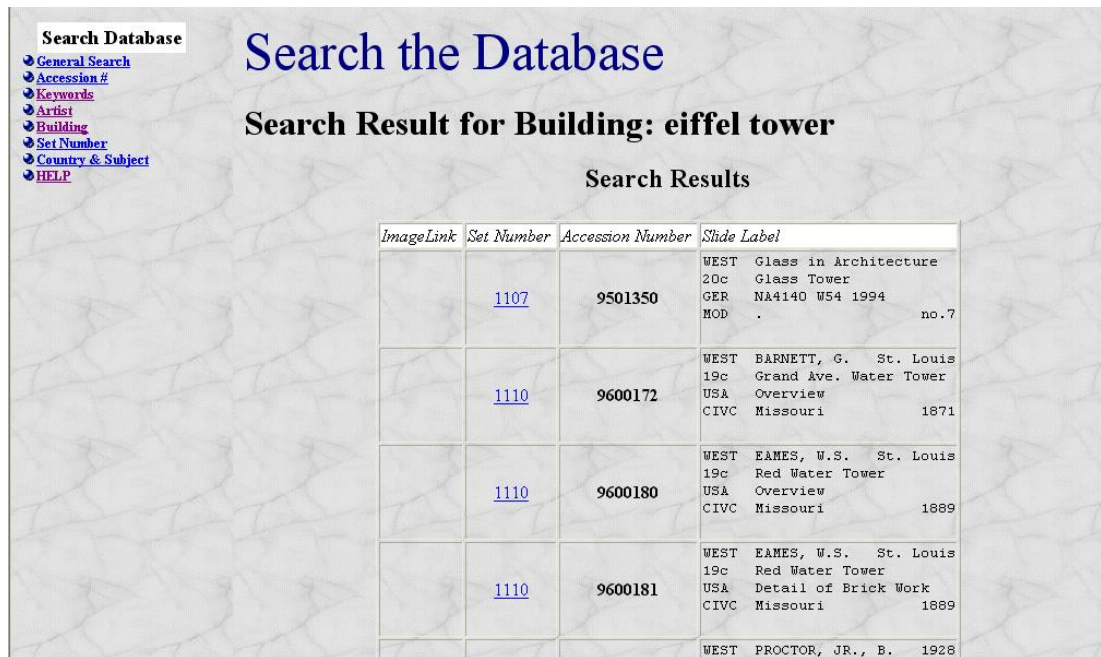
Şekil 6.20. “SMART ” Projesi Sorgulama Sayfası

6.9. “Virginia Tech Art & Architecture Library” Projesi

“http://db.lib.vt.edu/arch/” web adresinde bulunan ve Şekil 6.21’de giriş sayfası gösterilen, “Virginia Tech Art & Architecture Library” Projesi’nde arama yapıldığında Şekil 6.22’da görülen sonuç sayfası arayüzü ile karşılaşılmaktadır.



Şekil 6.21. “Virginia Tech Art & Architecture Library ” Projesi Giriş Sayfası



Şekil 6.22. “Virginia Tech Art & Architecture Library ” Projesi Sonuç Sayfası

6.10. “Walking Melbourne” Projesi

WALKING MELBOURNE

CITY **ARCHITECTS** **BUILDINGS**

SEARCH
FORUM
EVENTS
CONTACT
LINKS

DATABASE SEARCH You are here : [HOME](#) > [SEARCH](#)

SEARCH

Search by Style
Select an architectural **style** :

Search by Period
Select an architectural **period** :

Latest News
11/7/2003
St Kilda Cultural Landmark Destroyed



RIP St Kilda Kiosk - Razed by Fire 2003

The heritage registered second empire art-nouveau styled St Kilda icon, admired by many visitors, burned down overnight - but "will

Şekil 6.23. “Walking Melbourne ” Projesi Sorgulama Sayfası

“<http://www.walkingmelbourne.com/index.php>” web adresinde bulunan “Walking Melbourne” Projesi’nin arama sayfası Şekil 6.22’de gösterilen,

7. GÖRSEL MİMARİ VERİTABANI MODELİ – “SlidePack” SLAYT ARŞİVİ PROJESİ (VIRTUAL ARCHITECTURAL DATABASE – “SlidePack” SLIDE LIBRARY PROJECT)

7.1. Projenin İçeriğine Göre Tanımı

Veritabanı tanımı çok genel bir tanımdır. Karşımıza okulların, şirketlerin ve bireylerin her birinin bugüne kadar oluşturduğu sayısız veritabanı örnekleri çıkmaktadır. Hepsinin yapısı ve mantığı aynı olmasına rağmen veritabanları; içerdiği konularla, bilgilerle, işlenmeye hazır ham verileriyle birbirlerinden farklılık göstermektedir. Bu nedenledir ki veritabanları tanımlanırken, veritabanının konusu yani bilgi içeriği göz önünde bulundurulmaktadır.

Bu tezin bir parçası olan, “SlidePack” - Mimari veritabanı Uygulamasının konusu İstanbul Teknik üniversitesi Mimarlık Fakültesi slayt arşividir. Görsel Mimari Veritabanı, içerik olarak fakültemizin arşivindeki slaytların taranıp dijital resim formatına dönüşmüş halini ve bu slaytlar hakkındaki temel bilgileri taşımaktadır. İçeriğinin temelini fotoğraflar oluşturduğundan görsel bir veritabanı olarak tanımlanmaktadır. Amaçlanan temel unsur araştırmacıya istediği mimari fotoğrafı hızlı bir şekilde sunabilmektir.

7.2. Hazırlanan Model Uygulama İçin Amaç ve Değerlendirme

Hazırlanan bu veritabanı uygulamasındaki temel amaç; İstanbul Teknik Üniversitesi Mimarlık Fakültesi Mimari Slayt Arşivinin elektronik ortama aktarılması ve bu sayede hem öğretim üyelerinin, hem de öğrencilerin bu arşive kolay bir şekilde ulaşabilmesini sağlamaktır.

veritabanı oluşturulurken; yapının bulunduğu ülke ve şehir, yapının mimari tarzı, yaşı, işlevi ve teknolojik unsurları gibi bilgiler slaytlarla birleştirilerek veri paketleri oluşturulmuş oldu. Yani her bir dijitalleştirilmiş resim dosyasına, kendisine ait bu bilgiler eklenerek veri paketleri hazırlandı. Tüm veri paketleri birleştirildiğinde

ortaya çıkan Mimari Görsel Veritabanı sayesinde, erişilmesi ve kullanılması zor veri yığınları bir düzen içersine sokulmuş, daha da önemlisi hızlı bir şekilde kullanılabilir hale getirilmiştir.

veritabanına erişmek için kullanılan web kullanıcı arayüzü hazırlanırken, sorgulama yazılımları ile aranan bir slayta ulaşmak kolaylaştırılmıştır. Aranan bir slayta ait tek bir bilgi ile, o slaytı binlercesi arasından bulmak ve çekip almak saniyeler içinde yapılabilir olmuştur. Bu çok büyük bir zaman tasarrufu demektir. Zahmetli, sıkıcı ve hiçbir kazanımı olmayan “Bilgiyi binlercesi içinde aramak” işi bilgisayarlara bırakılmış; kullanıcıya ise sadece “bilgiyi kullanmak” kalmıştır. Slaytlara ulaşmak için artık Üniversite Arşivine gitmeye gerek yoktur, kullanıcılar bu slaytlara interneti kullanarak evden veya dünyanın herhangi bir yerinden ulaşabilmektedir. Bu hem zaman açısından hem de maddi açıdan kazanç demektir.

Veritabanı web arayüzünün görsel olarak çekici hale getirilmesine önem verilmiştir. Bu sayede konuyla ilgili bir araştırmaya gereksinimi olmayan kullanıcıların da slayt arşivini ziyaret ederek boş vakitlerini faydalı bir şekilde değerlendirmeleri amaçlanmıştır. Böylece öğrencilerin eğlenceli bir şekilde bilgi birikimlerinin arttırabilmeleri, diğer bir deyişle kendilerine artı değer kazandırmaları amaçlanmıştır.

Oluşturulan bu Görsel Mimari Veritabanının çok önemli bir avantajı daha vardır ki oda; uzun yıllar içerisinde çok büyük özveri, emek ve maddi harcamalarla biriktirilmiş bu slaytların güvenliğinin sağlanmasıdır. Binlerce slaytın içindeki her bir parçanın tek başına değeri çok büyüktür ve tek bir parçanın dahi zarar görmesi kabul edilemez bir durumdur. Bu slaytlar bir kereye mahsus olmak üzere, taranıp dijitalleştirilerek elektronik ortama aktarılmasıyla veritabanına eklenirler. Daha sonra tekrar kullanılmalarına gerek kalmadığından, Üniversite Arşivindeki uygun saklama ortamlarına kaldırılabilirler. Slaytların orijinallerine gerek kalmadan sergilenmesi mümkün kılınmıştır. Hatta ödünç verme yada kiralama gibi eski usul yöntemlerle tek bir slayttan o an için bir kişi yararlanabilmekteyken dijital ortamda durum artık böyle değildir. Düşük bir ihtimalde olsa aynı slayta, farklı fiziksel mekanlarda bulunan birden fazla kişi aynı anda ulaşabilmektedir. Bu yapı hem slaytların güvenliğini sağlamış hem de yine bilgiye erişimi kolaylaştırmıştır.

Görsel Mimari Veritabanı sayesinde, Üniversite Slayt arşivine ulaşmak bu kadar hızlı ve kolay hale getirilmesiyle, öğretim üyelerinin sınıflarda ders anlatımı esnasında eş zamanlı olarak bu slaytlara ulaşmaları mümkün kılınmış, böylece dersler sürükleyici, dikkat çekici ve sunulan bol örnekler sayesinde daha kolay anlaşılır hale getirilmeye çalışılmıştır.

7.3. Elektronik Ortamda Görsel Mimari Veritabanı Oluşumu

İlk aşamada yapılması gereken, “Plan”dır. Bir veritabanı oluşturulması esnasında en sık rastlanan hata öncelikle bilgi sistemlerinin gerçekleştirilmesidir. Oysaki başlangıçta yöntem sistemlerini, bilgi sistemlerine tercih etmek ve yöntem sistemleri arasında da somut sonuçlar verebilecek olanları seçmek genellikle daha yararlı olmaktadır. İlk aşamada veriler ikinci planda tutularak, verilerden bağımsız olarak çalışan, esnek ve kurallarla sınırlandırılmamış, geliştirilmeye açık bir saklama, sorgulama ve sunma yöntemi oluşturulmuştur. Bu aşama nispeten zor, karmaşık ve deneyim gerektiren bir aşama olmakla birlikte mutlaka yapılması gereken bir aşamadır. Modelimizde veritabanını oluşturmaya başlamadan önce şu şekilde bir plan yaptık;

Her bir veriyi diğerlerinden ayırabilmek için bir isim vermemizin şart olduğu açıktır. Bizim verilerimizi Slaytlar oluşturduğu için onları kolayca isimlendirmemiz mümkündü. Her bir slayta ilgili olduğu yapının ismini verdik. Ne var ki böyle bir isimlendirme ile amacımıza ulaşmamız mümkün değildi. Çünkü bu şekil bir adlandırma ile kullanıcıların istedikleri bir slaytı görmeleri için ya isimlerini tam olarak bilmeleri gerekirdi ya da binlercesini tek tek gözden geçirerek aradığına ulaşması gerekirdi. Bunu aşabilmek için her bir slayta diğer slaytlar ile ilişkilendirilebilmelerini sağlayacak, yapının bulunduğu ülke, yapının tarzı, mimarı gibi bilgiler eklendi. Böylece kullanıcı aradığı slaytın ismini bilmeseydi bile akılda kalıcı birkaç özelliğini kullanarak o slayta binlercesi arasından hızlı bir şekilde erişebilmesi sağlanmış oldu. Kullanıcı, ismini bilmediği yapının resmine ulaşmak için, yapım tarihi yada tarzı gibi bilgileri bilgisayara girmesiyle, bu özelliklere sahip olmayan birçok slaytın elenmesi mümkün olacaktır. Bu şekilde kullanıcının aradığını bulmak için gözden geçirmesi gereken slayt sayısı minimum sayıya indirilmiş olmaktadır.

Sonuç olarak veritabanımızın amaca uygun bir şekilde kullanılabilmesi için yöntem belirlenmiş oldu. Slaytlar veritabanında yapının ismiyle saklanırken ek olarak mimarı, yapım tarihi, tarzı gibi seçici unsurları da içermesine karar verildi. veritabanı yöntemi geliştirildikten sonra diğer aşama sistemin kurulması olmuştur. Bu işlem için ihtiyaçları karşılayacak, internet ve işletim sistemleriyle uyumlu, kolay öğrenilebilir ve geliştirilebilir, tanınan, bilinen yaygın olan yazılım dilleri ve programlar seçilmeye çalışılmıştır. Veritabanı programı olarak Microsoft Access, ihtiyaçları karşılamakta olduğundan ve de aranan niteliklere uyduğu için tercih edilmiştir. Bu ikinci aşama bilgiye dayalı uzun zaman alan bir süreç olmuştur. Üçüncü aşamada veritabanındaki bilgilere erişmeyi sağlayacak bir sorgulama yazılımı ve kullanıcı arayüz tasarımı yer almıştır. Bu işlemler içinde en son geliştirilen yaygın yazılım teknolojilerinden faydalanılmıştır. Sorgulama için ASP, SQL, Visual Basic yazılım dillerine başvurulmuş ve arayüz oluşturmak içinde HTML yazılım dilinin yanında Adobe Photoshop, Macromedia Dreamweaver ve Flash gibi paket programlardan faydalanılmıştır. En son adım ise verilerin sisteme dahil edilmesi olacaktır. Bu işlem slaytların tek tek bir elektronik tarayıcı cihaz yardımı ile dijital ortama aktarılması ve veritabanına dahil edilmesi işlemidir.

Tüm bu bahsettiğimiz veritabanının oluşum evrelerini daha ayrıntılı olarak, alt başlıklar halinde şu şekilde inceleyebiliriz;

7.3.1. Paradigma

Türk Dil Kurumunun güncel Türkçe sözlüğünden “Paradigma” sözcüğünün kelime anlamına baktığımızda, “Model” ve “Belirli bir alanda çalışan bilim adamlarının paylaştığı ortak değerler ve anlayışlar dizisi” olarak açıklandığı görülür. Yani hazırlayacağımız veritabanı projesi için öncelikle bir kurgu oluşturup, ihtiyaçlarımızı tam olarak belirlememiz, bizden istenenleri tam olarak netleştirmemiz gereklidir. Ancak bu adımdan sonra ihtiyaçları karşılayacak veritabanı için bir yöntem geliştirebiliriz.

Bu tez için hazırlanan veritabanı modelinde istenenler kısaca; Slaytların internet ortamında kullanıcıya sunulması, Sayıca binlerce olan tüm bu mimari slaytlar içinde, aranan tek bir slayta hızlı bir şekilde ulaşılabilmesi, öğretim üyelerinin bu arşivi ders esnasında eşzamanlı olarak kullanabilmeleri, zaman içerisinde veritabanına yeni

slaytları dahil edebilme, kullanıcıların kendi bilgi birikimlerini sınavabilmeleri için bir test bölümü ve kullanıcıların ilgisini çekecek, sıkıcı olmayan, kullanımı kolay bir görsel arayüz hazırlanması şeklinde sayılabilir. İhtiyaçlar belirlendikten sonra istenenleri karşılayabilecek bir veritabanı için “nasıl yapılandırılması gerektiği” hakkında bir yorumda bulunabiliriz yada bir fikir sahibi olabiliriz.

“SlidePack” Slayt arşivi veritabanı modelimizde slaytların listelenebilmesi için her bir slayta bir isim verilmesi gerektiğine karar verdik. Slaytların listelenmesinde hem görsel olarak resimlerin gösterimi yolunu hem de, yapı ismini aklında tutabilen, aradığı slaytın adını bilen kullanıcılar içinde isim listelemesi yolunu tercih ettik. Bu şekilde slaytlar kullanıcılara sunulmuş olacaktır. Kullanıcının slaytın ismini bilmediği durumda, aradığı tek bir slaytı bulmak için her birine teker teker bakmasını engellemek için, slaytlar arasında ilişki kurarak kendi aralarında gruplayabilmek için, bazı özelliklerini içeren bilgileri de veritabanına eklemeye karar verdik. Böylece veritabanı uygulamamız slaytların kullanıcıya sunulması işlevinin yanında, slaytlar hakkında bazı bilgilerin de sunulması özelliğini kendiliğinden kazanmış oldu. Artık kullanıcılara aradıklarını bulabilmek için slaytların hepsine tek tek bakmak yerine bir şehir ismi verip, yalnızca o şehirdeki mimari yapılara ait slaytları karşılarında bulabilmelerini sağlayabilir veya hem şehir ismini hem e mimar ismi girerek o mimarın belirtilen şehirdeki yapılarının slaytları gibi daha seçici ve detaylı bir şekilde arama yapmalarını sağlayabilir hale getirdik.

Slayt arşivi veritabanı uygulamamıza ait diğer beklentiler ile de, hangi tür yazılım teknolojilerini tercih edebileceğimiz konusunda genel bir fikre sahip olduk. Örneğin, slayt arşivinin ders esnasında kullanılabilir olması için, sistemin genel yapısı hızlı bir şekilde işlemeli, aynı anda birden çok kullanıcıya hitap edebilmelidir. Görsel arayüzü kullanışlı olmalı, kolayca tasarlanabilmeli ve ileride zaman içerisinde istenen değişiklikler hızlı ve sorunsuz bir şekilde yapılabilmelidir. Bu koşullara cevap verebilecek günümüz yazılım teknolojileri nelerdir? Biz bu teknolojilerden hangilerini seçmeliyiz? Sorularına yanıtlar aradık.

7.3.2. Veritabanı Uygulamasının Yapılanma Süreci

veritabanı uygulamamızın bu aşamasında kullanacağımız yazılım teknolojileri hakkında karara varmaya çalıştık. Bugüne kadar geliştirilmiş Veritabanı yazılım teknolojilerinden en yaygın olanlarını araştırarak işe başladık.

Bu güne kadar dünya üzerinde birçok veritabanı yazılımı geliştirilmiştir. Bunlardan bazıları; Access, MySQL, IBM DB2, Informix, Oracle, Interbase yazılımlarıdır. Adı geçen bu yazılımların hepsinin birbirine göre üstün ve zayıf yönleri bulunmaktadır. Her biri temelde aynı işi gerçekleştirmek için geliştirilmelerine karşın, farklı ihtiyaçlara cevap vermek için tasarlanmışlardır. Bu veritabanı yazılımlarına kısa bir şekilde değinmek gerekirse;

Microsoft firmasının Office paketi içinden çıkan Access, paralı veritabanları arasında nispeten ucuz olarak göze çarpar. Küçük ölçekli uygulamalardaki gereksinimleri karşılayabilir. Eğer bir web sitesinde veri miktarı ve aynı anda yapılan işlem sayıları az ise, Access kullanılabilir. Tek bir veri tablosunda 2 GB a kadar veri depolayabilir ve aynı anda 255 bağlantıya izin verilebilir. Access, Linux/MAC sistemlerinde kullanılamamaktadır, bu da yaygınlaşmasını engellemektedir.

MySQL Inc. Tarafından kodlanan MySQL, Access ile karşılaştırıldığında daha güvenlidir. Windows'un yanısıra Linux, OS/2, Solaris, AIX ve birçok işletim sistemini desteklemesi nedeniyle Access'e oranla daha yaygındır. Unix sürümü ücretsiz, ancak Windows sürümü ücretlidir. MySQL'in en büyük dezavantajlarından biri ücretsiz olmasından kaynaklanan destek eksikliğidir.

IBM firmasının ürünü olan DB2, Access ve MySQL'e göre daha performanslı, ancak küçük işletmelere göre daha yüksek maliyete sahiptir. Windows ve Unix sistemlerinde çalışabilir.

Informix, ücretli ve güçlü bir veritabanıdır. Orta ölçekli işletmelerin yükünü kaldırabilecek kapasitededir ancak yaygın olarak kullanılmaz.

Yine Microsoft firmasının bir ürünü olan Microsoft SQL Server (MSSQL), iyi bir performansa sahiptir. En büyük dezavantajı, sadece Windows üzerinde çalışabilmesidir. Kullanım kolaylığı, güvenilirliği ve işlem gücüyle dikkat

çekmektedir. Maliyeti diğer veritabanlarına göre yüksektir. Tablo başına 4 TB veri depolayabilmektedir.

Oracle, dünyanın en güçlü ve güvenilir veritabanı olarak gösterilmektedir, ancak çok yüksek maliyeti nedeniyle sadece büyük kurumların tercih edebilecekleri bir veritabanıdır. Windows ve Unix sistemlerinde kullanılabilir. Unix sürümü ücretsizdir, ancak özellikle Linux sistemleri üzerinde kurulumu gerçekten zahmetlidir. Oracle 8i, RedHat Optimized Linux 6.2 üzerine kurulabilir. Oracle, sınırsız sayıda tabloları desteklemektedir.

Interbase ise Linux platformu üzerinde çalışan, ücretsiz bir veritabanı sunucusudur. Yaygın değildir. Ücretsizdir. (<http://e-dergi.emo.org.tr>, 2004)

Edinilen bu bilgiler ışığında, yaygın kullanımı, yeterli performansı, diğer teknolojilerle uyumlu çalışması, kolay öğrenilebilmesi, olası bir sorunla karşılaşıldığında başvurulabilecek kaynak sıkıntısının bulunmaması nedeniyle Microsoft firmasının ürettiği Access Veritabanı yazılımının bizim için en uygun seçim olabileceğine karar verdik. Access veritabanı yazılımının izin verdiği sınırlar çerçevesince tablolarımızı oluşturmaya başladık. Aşağıdaki şekilde birbiriyle ilişkili olarak hazırlanan tabloların bir listesi görülmektedir.



Şekil 7.1. : Access ile hazırlanmış, birbirleri ile ilişkili tablolar listesi.

Veri tablolarımızı oluştururken mümkün olduğunca veri tekrarından kaçınmaya çalıştık. Veri tekrarı yazılımın işleyişini yavaşlatacak, gereksiz yere kapasiteyi dolduracak, yazılımın takibini güçleştirecek, tablolar arası ilişkiler kurmayı zorlaştıracaktır. En önemlisi de yanlış girilmiş bir kaydın, düzeltilmesi veya silinmesi istendiğinde, o kayıtla ilgili tüm tekrarlı verilerin tamamının silinmesinin gerekmesidir. Yani veri tekrarları, yalnızca tasarım aşamasında sorunlara yol açmaz, programın kullanıcıları için de, sonradan güçlükler ortaya çıkarır.

Kullandığımız Access veritabanı yazılımının, kullanım kuralları doğrultusunda hazırladığımız örnek bir tablo aşağıdaki şekilde görülmektedir;

Ogeler : Tablo								
	alanno	sirano	resim	alanad	ulke	mimar	sehir	tarih
▶	200	100	foto201.jpg	CASA BATLLO	ISpanya	ANTONIO GAUDI	BARSELONA	1907
	200	200	foto202.jpg	CASA BATLLO	ISpanya	ANTONIO GAUDI	BARSELONA	1907
	200	300	foto301.jpg	CASA BATLLO	ISpanya	ANTONIO GAUDI	BARSELONA	1907
	300	100	foto302.jpg	CASA MILA	ISpanya	ANTONIO GAUDI	BARSELONA	1910
	300	300	foto303.jpg	CASA MILA	ISpanya	ANTONIO GAUDI	BARSELONA	1910
	400	100	foto401.jpg	CENTRE POMPIDOU	FRANSA	ROGERS	PARIS	1976
	400	200	foto402.jpg	CENTRE POMPIDOU	FRANSA	ROGERS	PARIS	1976
	600	100	foto601.jpg	EIFFEL TOWER	FRANSA	GUSTAVE EIFFEL	PARIS	1889
	600	200	foto602.jpg	EIFFEL TOWER	FRANSA	GUSTAVE EIFFEL	PARIS	1889
	600	300	foto603.jpg	EIFFEL TOWER	FRANSA	GUSTAVE EIFFEL	PARIS	1889
	700	100	foto701.jpg	EMPIRE STATE	AMERIKA	SHREVE	NEW YORK	1931
	700	200	foto702.jpg	EMPIRE STATE	AMERIKA	SHREVE	NEW YORK	1931
	700	300	foto703.jpg	EMPIRE STATE	AMERIKA	SHREVE	NEW YORK	1931
	700	400	foto704.jpg	EMPIRE STATE	AMERIKA	SHREVE	NEW YORK	1931
	900	100	foto901.jpg	GUGGENHEIM	AMERIKA	FRANK LLOYD WRIGHT	NEW YORK	1959
	900	200	foto902.jpg	GUGGENHEIM	AMERIKA	FRANK LLOYD WRIGHT	NEW YORK	1959
*								0

Şekil 7.2. : Veritabanı uygulaması için hazırlanan “Ogeler” isimli tablo.

7.3.3. Hazırlanan Veritabanı İçin Arama Motorunun Oluşturulması

Arama motoru olarak bahsedilen şey aslında web sayfasının içine gömülmüş “Script” denilen program parçacıklarıdır. Bu program parçacıkları, sıradan web sayfalarını kullanıcı ile etkileşimli hale getirir. Herhangi bir internet sayfasına girildiğinde, bilgiler monitöre önceden planlanmış bir düzen içerisinde yerleştirilir. Bilgiler, web sayfasından kullanıcıya doğru tek yönlü bir şekilde aktarılır. Genellikle kullanıcıların web sayfasını açmak yada kapatmak veya başka bir sayfaya geçiş yapmak dışında bir rolü yoktur. Sayfayı ziyaret eden kullanıcılar, veri aktarımı esnasında son derece pasif bir rol oynarlar. Kendilerine aktarılabilecek veriler içinden seçim yapamazlar, yada kendilerine sunulan veriler dışında başka bilgilere erişmek gibi bir talepte bulunamazlar.

Gelişen web teknolojileri bu eksikliği ortadan kaldırmıştır. Günümüzde kullanıcıyla iletişim kurabilen dinamik web sayfaları oluşturabilmekteyiz. Bunu da ASP, JavaScript, VBScript gibi script dilleri ile hazırlayıp HTML belgelerinin içine yerleştirerek başarmaktayız.

Arama motorumuzun oluşturulmasında Microsoft firmasının geliştirdiği ASP (Active Server Pages) dilini kullanmaya karar verdik. Çünkü, ASP dili en son geliştirilmiş teknolojilerden biridir. Sunucu taraflı değil istemci taraflı çalışır. Yani siteyi ziyaret eden kullanıcı web sayfasıyla birlikte arka planda bu ASP kodlarını da web tarayıcısına yüklemiş olur. Arama motoru siteyi yayınlayan sunucu (Server) bilgisayarını arama motorunu ziyaretçinin bilgisayarına yani istemci (Client) bilgisayara göndermiştir. Kullanıcının verdiği komutları başka bir deyişle kullanıcı taleplerini sunucu bilgisayar değil istemci bilgisayar değerlendirir. Daha sonra bu talepler doğrultusunda sunucu bilgisayardaki kayıtlı veritabanı ile bağlantı kurar. Arama motorunun istemci tarafında çalışması, aynı anda birçok ziyaretçi için işlem yapan sunucu bilgisayarın iş yükünü azaltır. Son derece kullanışlı ve hızlı bir teknolojidir. ASP dili Visual Basic ve HTML dillerine de çok benzer olmasından dolayı kolay öğrenilebilen bir dildir. Çok yaygın olarak kullanıldığından bu script dili ile ilgili birçok kaynak bulmak mümkündür.

Bir internet sayfası yada arama motoru oluşturmak için ilk önce Web sunucusu bir bilgisayara ihtiyaç vardır. Microsoft firmasının üretmiş olduğu ve Windows işletim sistemi CD si ile birlikte ücretsiz olarak dağıtılan IIS (Internet Information Server) yazılımını yükleyip mevcut bilgisayarımızı kişisel bir web sunucusuna dönüştürdük. Böylece ASP ile programlar yazıp, bu programları test edebilmek için gerekli yazılım ve donanım koşullarını sağlamış olduk.

IIS; Windows işletim sisteminin, WWW (web tabanlı bilgiyi yayımlama ve uygulama teslim etme), FTP (dosya transferi), NNTP (haber grupları), ve SMTP (posta) kullanımını sağlar. IIS; Internet ve Intranet siteleri için bir sunucu çözümünüdür.

ASP script dili kullanarak hazırladığımız arama motorunun görevi; web ortamında, ziyaretçinin taleplerini değerlendirmek, bu talepler doğrultusunda Access ile hazırlamış olduğumuz veritabanı dosyasıyla irtibat kurarak, kullanıcı ekranına

istenen bilgileri ulařtırmaktır. Örneğın kullanıcı, “slayt arřivinde bulunan Barselona’daki tüm mimari yapıların slaytlarını görüntüle” gibi bir talepte bulunduğında arama motorumuz bu talebi deęerlendirerek veritabanındaki kayıtlı bilgilere ulařıp tamamını gözden geçirerek, Barselona řehrindeki binaların slaytlarını seçip kullanıcıya ulařtıracaktır.

Oluřturduğumuz arama motoru ile kullanıcıya web sayfası ierisinde istedięi slaytların görüntülenmesi iin üç yöntem önerilir; “Basit arama”, “Detaylı arama” ve “Tüm slaytlar”. Basit arama yöntemini seçen kullanıcı, hangi kritere göre arama yapacaksa o kriteri seçip anahtar kelimeyi girerek istedięi slaytlara ulařır. Basit arama kısmındaki arama kriterleri sırasıyla; Yapı adına göre arama, Mimar adına göre arama, řehir adına göre arama, Ülke adına göre arama, Mimari tarzına göre arama, Yapısına göre arama, İşlevine göre Arama olarak hazırlanmıřtır. Kullanıcı örneğın “İşlevine göre arama” seçeneğini seçip “Kütüphane” anahtar kelimesini girdiğinde, veritabanında kayıtlı tüm kütüphane binalarının slaytları listelenir.

Detaylı arama seçeneğinde ise kullanıcı Mimar adı, yapının adı, řehrin adı, Ülkenin adı, yapının işlevi, yapının tarzı kriterlerinden bir yada birkaçını veya hepsini birden girerek daha seçici bir arama yapabilir. Bu durumda yine oluřturduğumuz arama motoru bu talebi deęerlendirerek kriterlere uygun slaytları veritabanından arařtırıp, kullanıcıya sunacaktır.

Tüm slaytlar seçeneğinde ise, arama motoru kullanıcıdan belirli bir kriter beklemeksizin, tüm slaytların ismini ve bu slaytlara ait verileri bir liste halinde sunar. Kullanıcı buradan herhangi bir slaytın ismine tıklamak sureti ile istedięi slaytın resim dosyasına ulařır.

Birde yine yine ASP script dilini kullanarak arama motoruna eęlenceli bir test yerleřtirdik. Bu testte kullanıcıya, slayta ait resim dosyası görüntülenir ve kendisinden binanın ismi, yapısı, mimarı gibi bilgileri girmesi istenir. Kullanıcı bu bilgileri girdikten sonra arama motoru, veritabanı dosyasına ulařarak bu girilen bilgilerin doęruluęunu karřılařtırır ve yanlıřların yanına doęrularını yazarak görüntüler.

7.3.4. Kullanıcı Aarayüzünün Hazırlanması

veritabanına erişim, internet yoluyla olacağı için, kullanıcı arayüzünü web sayfası şeklinde tasarlamak uygundur. Nasıl Microsoft Word “.doc” ve “.txt” uzantılı dosyaları destekliyor ve bu dosyaları açabiliyorsa, Internet Explorer veya Netscape Navigator gibi web tarayıcı programları da “.html” ve “.htm” uzantılı dosyaları desteklemektedir. Web sayfaları aslında sadece “.htm” ya da “.html” uzantılı HTML kodlarını içeren veri dosyalarıdır (HTML; Hypertext Markup Language). Notepad veya Microsoft Word gibi herhangi bir kelime işlem programını çalıştırıp, HTML kodlarıyla bir web sayfası oluşturulursa ve sonra bu dosya “.htm” uzantılı olarak kaydedilirse, web tarayıcı programlar (Internet Explorer yada Netscape Navigator gibi) bu dosyaları açabilir, HTML kodları ile hazırlanan bu web sayfasını görüntüleyebilir.

Kelime işlem programlarından başka, HTML dili ile web sayfaları oluşturmak için özel olarak tasarlanmış “HTML Editörleri” de bulunmaktadır. Bu HTML editörleri, web tasarım işlemini kolaylaştır ve hızlandırırlar. Tasarım esnasında, HTML kodları üzerinde yapılan değişikliklerin veya eklentilerin sonuçlarını anında görüntüleyerek programcının hata yapmasını engeller. ASP kodlarının HTML kodlarına eklenmesine yardımcı olur ve senkronize bir şekilde çalışmalarını sağlarlar. Biz de projemizde, HTML yazılım dili ile Veritabanına erişim arayüzünü hazırlarken, Macromedia Firmasının ürettiği Dreamweaver isimli HTML editörünü kullanmayı uygun bulduk.

Sayfayı çekici hale getirmek önemlidir. Görsel uyarı okuyucuyu sayfaya daha çok çeker. İyi bir tasarım okuyucuyu yormamalıdır bu nedenle sayfayı gereksiz bilgilerle doldurursak okuyucunun kafası karışır. İlk anda okuyucuya çok fazla bilgi verdiğimizde okuyucu bundan sıkılacaktır. Başarılı tasarımlar, içeriği tam ve basit olarak aktarılanlardır. Bu aynı zamanda şıklık ve stil getirecektir.

Tasarımda amaçlanan görsel bütünlüğü ve düzeni sağlamak olmalıdır. Gerekli görsel elemanlar minimumda tutulup gruplanmalıdır. O zaman tasarımda bir bütünlükten bahsedebiliriz. Tasarımın en önemli kuralı bütünlüktür. Tasarımda elemanların bir uyum içinde bir araya getirilip gruplanması dağınıklığı önleyeceği gibi, tasarıma ritim de katacaktır. İyi bir sayfa tasarımı benzerliği ya da içeriği odaksal bir noktada

toplayıp onu görünür hale getirmelidir. Bütünlüğü, elemanları bir araya getirerek ve aralarındaki ilişkiyi kullanarak başarabiliriz.

Diğer yandan kontrastlık yaratılmadan yapılan tasarım konusuz ve iletişimsiz hale gelir. Benzerlikle kontrastlık arasındaki dengeyi iyi sağlarsak, tasarım etkileyici ve dinamik olur. Bu da, tasarımdaki elemanların iyi gruplandığı anlamına gelir. Ekrandaki boş alanlarının (negatif alan) ve harflerin kullanımı bütünlüğü yaratmanın iki önemli faktörüdür. Sayfa tasarımında olması gereken koşulları şöyle sıralayabiliriz; Bütünlük, Boşluk (negatif alan), Hiyerarşi, Denge ve Renk.

Bütünlük, görsel benzerliği olan elemanlarda vardır. Bütünlüğü kuramazsak okunamayan ve kafa karıştırıcı bir tasarım elde ederiz. Fakat çeşitliliği olmayan bir tasarım da hareketsiz, cansız ve ilgisiz hale gelir.

Boşluk (negatif alan) ve harfler bütünlüğü yaratmanın en önemli elemanlarından. Boşluklarla harfler birbirleriyle kontrastlık yaratırlar. Kontrastlığı iyi ayarlırsak ortaya dağınık olmayan uyumlu bir durum çıkacaktır. Tasarımın amacı bütün boşluğu doldurmak olmamalıdır.

Hiyerarşi, verilmek istenen mesajın elemanlar yardımı ile işaret edilmesine dayanır. İyi tasarım, okuyucuyu yazının ve görsel elemanların anlamına göre yönlendirir. En önemli daha az önemli sıralaması gibi. Tasarım içindeki elemanları vurgulamak istenen mesaja göre ölçülendirirsek, ki bunu renkle, uzaklık yakınlık veya beyaz boşlukla da yapabiliriz, o zaman tasarımda hiyerarşi denen koşul gerçekleşmiş olur.

Bütünlüğü başarmanın en iyi yolu dengedir. Eğer kullandığımız elemanlar dengede ise, tasarımda bütünlükten bahsedebiliriz. Denge tek başına bir etki yaratır. Eğer denge yoksa, elemanlar tek tek ortada kalırlar. Dengeyi sağlamanın en iyi yolu elemanları optik bir noktada gruplamaktır. Daha küçük ve okunmaz elemanlarla daha büyük ve parlak olanlar arasında denge sağlanmalıdır. Optik nokta geometrik olarak sayfa ortasının biraz üstüdür. Çağdaş tasarım daha dinamik olmak zorundadır, dolayısıyla odak noktasını biz kendi dinamiklerimize göre yaratabiliriz. Grupladığımız tasarım elemanlarını diğer koşulları da göz önüne alarak sayfa içinde odaklayabiliriz.

Renk bilimsel olmakla beraber kullanımı da, tamamen sağduyuya bağlıdır. Renk, tasarımın önemli bir elemanıdır. Renk ve ton değerlerini ayarlayarak elemanları öne veya arkaya alabiliriz. Renk, elemanların önem sırasını belirler ve yönlendirme sağlar. Renk yanlış kullanılırsa tasarımda problemler oluşmaya başlar. Sıcak renkler (kırmızı, turuncu, sarı gibi) dikkatimizi çeker ve bize yakın dururlar, soğuk renkler (mavi, yeşil, mor gibi) ise, daha geride bizden uzak duranlardır. Siyah ve beyaz, renk olmasalar da etkin ve vurucu özelliklere sahiptirler. Kısaca renk hangi elemanın daha önemli olduğunu belirtir. O elemanın küçük olması önemli değildir.

Eğer bütün bunları özetlersek, şöyle sıralayabiliriz: Okuyucu tasarımın sadeliğine dikkat eder. Sayfalar birbirleriyle ilintili olmalıdır. Uzaklık bizden ayrılır, ama yakınlık bizle beraberdir. Büyük ve koyu önemlidir, küçük ve açık daha az önemlidir. Bütünlük boşluk ile dengelenmelidir. Tasarım elemanları gruplanmalıdır. Herşeyin bir şekli vardır, boşluk da bunlardan biridir. **(İstek, 2004)**

Bu önemli hususları göz önünde bulundurarak yaptığımız web sayfası tasarımımızı daha ilgili çekici hale getirmek için yararlandığımız çok önemli bir program da, Adobe firmasının geliştirdiği Photoshop programıdır. Fotoğraf ve resim işleme yazılımı olan Photoshop ile web sayfamızda kullandığımız tüm resim dosyalarını düzenledik. Veritabanına erişim için kullanılacak web sayfası formatındaki “veritabanı kullanıcı arayüzü” nün sayfa düzeni (Şekil 7.6 da görülmektedir) ve grafik tasarımı tamamlandıktan sonra, HTML kodlarının arasına daha önceden ASP ile hazırladığımız arama motoru scriptleri (program kodları) yerleştirildi. Böylece Uygulamamızın tüm yazılım kısmı tamamlanmış oldu.

7.3.5. Tamamlanan Projenin İnternet Ortamına Aktarılması ve Test Edilmesi

Proje, henüz tasarım aşamasında iken defalarca kişisel bilgisayarlarda test edilmesine karşılık, internet üzerinde de test edilmesi gereklidir. İnternet ortamı, başı ve sonu olmayan, uçsuz bir ağ ortamıdır. Sayısız bilgisayar bu ağ ortamına çok farklı teknolojilerle bağlıdır. Ülkemizde en yaygın olarak ADSL ve Dial-Up kullanılmaktadır ancak, uydu bağlantısının optik bağlantılara kadar birçok teknolojiyle her türlü bilgisayar sistemi internet vasıtasıyla birbirine bağlantı kurmuştur. Dolayısıyla çalışma ortamımızda, basit ethernet kartlarıyla ve Cat5

standart network kablolarıyla oluşturduğumuz ağ ortamı ile hazırladığımız web veritabanı projemizi test etmemiz yeterli olamayacağı açıktır.

İnternet ortamına projemizi taşıyabilmek için bir İnternet Server bilgisayarına ihtiyaç vardır. İnternet Server bilgisayarları sürekli olarak açık ve internete bağlı durumdadırlar. İnternetteki diğer tüm bilgisayarlar bu Serverlara erişebilir ve yayınladıkları web sayfalarına yada dosyalara (önceden tanımlanmış izinler dahilinde) ulaşabilir. Bu Serverlar üniversitelerin bilgi-işlem laboratuvarlarında olabileceği gibi, birçok özel firma da server bilgisayarlara sahiptir ve ücretli olarak bu hizmeti vermektedir.

Projemiz ASP kodları içerdiğinden, uygulamamızı yayınlayacak olan Serverların da bu desteği sağlaması gerekmektedir. İnternetteki bazı Serverlar hem ASP desteği vermekte hem de ücretsiz olarak bu hizmeti sağlamaktadır. Örneğin <http://www.domaindx.com/> sayfasında 100 mb'lık alan ile ASP desteği ücretsiz olarak kullanıma sunulmuştur. Yine <http://www.7host.com/> adresinden erişilebilen Serverlarda da 50 mb'lık alan ASP desteği ile ücretsiz kullanıma açıktır. Yerli ve yabancı birçok server gerek deneme amaçlı olarak herkese, gerekse akademik çalışmalarda yardımcı olmak amacıyla tüm öğrencilere ücretsiz olarak bu hizmetleri vermektedir. Bu örnekleri <http://www.phidji.com/> , <http://www.brinkster.com/> , <http://www.turkpark.net/>, sitelerinden ulaşılabilen Serverlar ile çoğaltmak mümkündür.

Server ile bağlantı kurmak için bir de FTP programına ihtiyaç vardır. FTP (File Transfer Protocol) programları, Server bilgisayarlarına bağlantı kurmak için en kolay yollardan birisidir. Bu programlar ile uzaktaki server bilgisayarın web adresi sayesinde bağlantı kurulur, gerekli kullanıcı parolaları girilerek kişisel bilgisayarımızdaki hazırladığımız proje dosyaları bu Serverlara gönderilir. (UpLoad İşlemi)

Uygulamamızı, Serverlara yükledikten sonra, sıradan bir kullanıcı gibi web sayfasına bağlanılarak programın doğru çalışıp çalışmadığı, yeterli hızda verilere ulaşıp ulaşılamadığı test edilmiştir. Uygulamamızın testi için, geçici adresimiz <http://www.saglik-info.com/aysegul/myweb/index.html>'dır.

7.4. SlidePack Slayt Arşivi Projesinde Kullanılan Bilgisayar Teknolojileri

7.4.1. Veritabanı Oluşturulmasında Microsoft Access 2002

Microsoft Office Program paketi içerisinde yer alan Access veritabanı yönetim programı ile veritabanı oluşturulabilir veya özel uygulamalar geliştirilebilir. Access, Oldukça basit yapısı ve sunduğu geniş fonksiyonlar ile basit veritabanlarından gelişmiş bilgi sistemlerine çeşitli uygulamalar geliştirmek için uygun bir altyapı sunmaktadır.

Farklı uygulamalar kullanılarak geliştirilen veritabanları ile Microsoft Access altında çalışmaya devam edilebilir veya burada oluşturulan veritabanları, farklı dosya biçimlerinde saklanabilir. Kullanım kolaylığı ile amatör kullanıcılar için bir başlangıç noktası teşkil eden program, sunduğu SQL gibi standart çalışma araçları ile de profesyonel kullanıcıların da ihtiyaçlarına cevap verebilmektedir. Microsoft Access ile entegre olarak sunulan Microsoft Visual Basic for Application (VBA) programlama dili sayesinde daha etkili veritabanı çalışma ortamları geliştirilebilir ve çeşitli uygulamalar programlanabilir. Ayrıca tüm Office uygulamaları içerisinde bulunan VBA ile Access'in diğer programlar ile de etkileşimli olarak çalışması sağlanabilmektedir. Microsoft Access'in en önemli özelliklerinden birisi de Web bağlantısıdır. Bu sayede veri giriş çıkışı için kullanılan form ve raporlar, grafik tasarım fonksiyonları sayesinde herhangi bir kod yazmaya gerek kalmadan direkt olarak internet veya intranet ortamında yayınlanabilmektedirler.

7.4.2. Kişisel Bilgisayarlarda İnternet Server Ortamının Yaratılması İçin IIS 5.1

Internet Information Server (IIS), Microsoft Windows XP Professional işletim sistemi içinde hazır olarak yer alan bir Web server yazılımıdır. Belgeleri paylaşmak ya da belgelere Intranet ya da Internet yoluyla ulaşmak için kullanılan bir üründür.

Bu program bilgisayarımızda, veritabanı uygulamamıza erişmek amacıyla hazırlayacağımız web sayfası ve ASP scriptleri için bir alt yapı oluşturmaktadır. Internet Information Server programı sayesinde bilgisayarımız geliştirdiğimiz

program kodlarını işletebilmekte ve ağıdaki diğer bilgisayarlarımıza bu sayfaları yayınlayabilmektedir.

IIS programı, WWW (web tabanlı bilgiyi yayımlama ve uygulama teslim etme), FTP (dosya transferi), NNTP (haber grupları), ve SMTP (posta) hizmetlerinin kullanımını sağlar. IIS, Microsoft Visual Basic programlama dilini, VBScript, Microsoft Jscript geliştirme araçlarını, ve Java bileşenlerini tamamen destekler. IIS ayrıca Web tabanlı programlar için ASP uygulamalarına ve .asp uzantılarına tamamen destek verir.

IIS'in sunduğu WWW Servisi HTTP yi destekleyerek kullanıcıların Internet üzerinde web sayfalarının yayını yapmalarına izin verir. Web, Internet üzerindeki grafiksel servistir ve en karışık link yeteneklerine sahiptir. Oluşturulan dosyalar Web sitesi üzerindeki klasörlere yerleştirilir. Böylece kullanıcılar dosyaları bir Web tarayıcı kullanarak görebilirler. Hypertext Markup Language (HTML) kullanarak metin, grafik, animasyon veya video içeren dokümanlar yayınlanabilir.

Yine IIS'in servislerinden olan FTP hizmeti (Transmission Control Protocol / Internet Protocol – TCP / IP), ağ üzerinde bulunan bilgisayarların dosya transferi için kullandıkları standart bir protokoldür. FTP, TCP / IP ağları ve Internet üzerinde kullanılan en eski protokollerden birisidir.

WWW ve FTP siteleri bir bilgisayar üzerinde birden çok alan adı (domain name) bulundurulmasına olanak verir. Bir bilgi sunucusu (Information Server) kurulumuyla bir bilgisayar üzerinde birden fazla farklı WWW ve FTP siteleri kurulabilir. Örneğin, bir Internet Service Provider (Internet hizmet sağlayıcı – ISP) bir bilgisayar üzerinde farklı kuruluşlara ait ve her birinin kendine ait bir domain name ve IP adresi bulunan Web siteleri bulundurabilir. Bir ISP, farklı müşterilere ait Web ve FTP sitelerini oluşturmak için Internet Service Manager (Internet Sunucu Yöneticisi) kullanır ve bu sitelerin her birine tekrarlanmayan (Unique) bir tanımlama bilgisi ve özellik ayarları atar. IIS, sayesinde sınırsız sayıda web sitesi oluşturulabilir.

IIS'in sunduğu SMTP Servisi, standart ve güvenli olan bir e-mail (elektronik posta) sunucusudur. SMTP servisi, SMTP ve Post Office Protocol version 3 (POP3) için dağıtılmış bir e-posta hizmeti sağlar. SMTP Servisi gelenlerin (Inbox) birden çok

sunucuya bölünmesini sağlar. Ayrıca gelenlerinin yedeklenmesini, tekrar yüklenmesini ve sunucular arasında taşınmasına olanak verir.

NNTP Servisi ise elektronik tartışma grupları oluşturur. NNTP, Internet siteleri ve hizmetlerine aşağıdakileri sağlar:

- Genel, sadece okunabilir, değiştirilebilir veya özel (yetkilendirilmeli) tartışma grupları oluşturur.
- Kullanıcılara izin vererek uzaktaki bir tartışma grubuna erişimi sağlar. Bu, Internet üzerinde en çok kullanılan elektronik tartışma grupları olan Usenet haber gruplarını da içerir.
- Standart Windows NT Server mekanizması içerisinde güvenli yetkilendirmeyi sağlar.
- SSL kullanarak haber gruplarına şifreli erişim sağlar. (**www.bilgisite.com, 2004**)

7.4.3. Veritabanı Erişimi İçin Web Kullanıcı Arayüzü Oluşturulmasında HTML

HTML (Hyper Text Markup Language) internet ortamında doküman yaratmak için geliştirilmiş bir programlama dilidir. HTML dokümanı bir metin dosyadır ve tek başına bir yazı olmaktan öteye gidememekle birlikte her hangi bir internet tarayıcısı (Internet Explorer, Netscape Navigator v.b) yardımıyla çalıştırıldığında içerdiği komutlara göre ve de tarayıcı programın desteklediği özelliklere göre, hazırlanan dokümanlar bir anlam kazanmaktadır. (**www.pcakademi.com, 2004**)

HTML dili programlama mantığından biraz uzak, görsel yönü oldukça önemli olan bir dildir. HTML'nin öğrenilmesi, diğer programlama dillerine göre daha kolay olup bazı temel kural ve komutların bilinmesi yeterli olmaktadır. HTML'in temel mantığı "Ne görürseniz onu alırsınız" (What you see what you get) şeklinde açıklanmaktadır.

HTML' den "dil" diye söz edilmesine karşılık, bazı kaynaklar onu "programlama dilleri" kapsamına almamaktadırlar ve "dil" olarak kabul etmemektedirler. Çünkü HTML bir programlama dilinin karmaşıklığından oldukça uzak basit bir dildir ve derlenmeye gereksinim duymamaktadır. Yani HTML kullanarak çalıştırılabilir bir dosya elde edilmemektedir. Sonuç olarak ortaya çıkan, derlenmiş .EXE ya da .COM

uzantılı bir dosya olmamaktadır. Hazırlana HTML dosyaları bilgisayarın anlayacağı dile çevrilip anlamsız karakterler dizisine dönüşmemektedir. Bir HTML belgesi düz metinlerden oluşan bir dosyadır. Yani bu dosyalar, Windows ortamında NotePad (Not Defteri) programı ile, Mac ortamında Simple Text programı ile oluşturulabilmektedir. (www.geocities.com/htmlrehberi/html.htm, 2004)

Bir HTML belgesi temel olarak iki bölümden oluşmaktadır. Web tarayıcılarının dosyayı olması gerektiği gibi görüntülemesini sağlayan, dosyayla ilgili bazı bilgileri bilgisayara aktaran ve ziyaretçinin normalde göremediği HEAD bölümü ve tarayıcı penceresinde görüntülenen tüm içeriğin (kodlarının) yer aldığı BODY bölümü. Bir HTML dosyası her zaman <html> etiketi ile başlar. Web' de bir alan ziyaret edilmek istendiğinde, web server, tarayıcıya gerekli dosya ya da dosyaları gönderir ve tarayıcı eline geçen bu malzemeyi "yorumlamaya" başlar. İşte bir HTML dosyası, tarayıcıya HTML dosyası olduğunu ilk satırındaki <html> etiketi ile söylemektedir. Daha sonra bu etiketi <head> etiketi takip etmektedir. <head> etiketinin görevi bitince, </head> şeklinde sonlandırılır ve ardından sayfa içeriğini taşıyan <body> etiketinin sırası gelmektedir. Sayfa içeriğini oluşturan etiketler bitince, </body> şeklinde BODY bölümü sonlandırılmaktadır ve HTML belgesi </html> yazımı ile bitirilmektedir.

7.4.4. Web Sayfalarının Etkileşimli Hale Getirilmesinde ASP

Microsoft'un sunucu taraflı Web hizmetleri aslında Internet Veritabanı Bağlayıcısı (Internet Database Connector-IDC) ile başlamaktadır. IDC teknolojisi, veritabanları için ODBC+SQL tanımları içeren .idc uzantılı dosyaların işletilmesi sonucu elde edilen verilerin html şablon dosyalarında (htx) tanımlanan biçimde dinamik sayfa haline getirilmesine sağlayan bir öncü teknoloji olarak Web tabanlı veri sunumları-sorgulamaları uygulamalarında Web geliştiricilere büyük kolaylıklar sağlamıştır. O zamana kadar dinamik sayfalar yaratmak için kullanılan yöntem sadece, Common Gateway Interface (CGI) tekniğine dayalı sunucu taraflı yürütülebilir programlar (exe) yazmaktı.

Geleneksel program dillerindeki yetersizlikleri ve birtakım güçlükleri (derleme gibi) aşmak için, büyük ölçüde Internet uygulamalarına özel ve CGI standartları destekleyen bir yorumlama dili olarak ortaya çıkan Perl, uzun yıllardan beri Web

programcılarının en vazgeçilmez araçlarından biri olmuştur. Perl ile yazılan ve tamamıyla metin türünde olan script dosyaları Web sunucusu üzerindeki Perl yorumlayıcısı tarafından işletilerek sonuçlar HTML biçiminde istemciye gönderilmektedir. Böylece dinamik sayfaların oluşturulması sağlanmaktadır. Ancak, Perl dili ile çalışmak bazı açılardan güçtür. Her şeyden önce bir çok programcı için Perl, öğrenilmesi gereken, deneyim kazanılması gereken yeni bir dildir. Diğer yandan programcılarının tamamına yakın bir kısmı Basic dilinde kodlama yapabilecek durumdadır.

Microsoft firması bu sıkıntının aşılması amacıyla, Visual Basic diye tanınan ürünü parçalara ayırarak özelleştirmeye karar vermiştir. Visual Basic for Application (VBA), Visual Basic Scripting (VBS), Visual Basic for Access vs. gibi yeni ve özelleştirilmiş ürünler ortaya çıkarmıştır. Microsoft, Internet Explorer 3 sürümüyle birlikte VBS'yi alternatif bir script aracı olarak piyasaya sürmüştür. Ancak, VBS o tarihlerde yalnızca Internet Explorer tarafından desteklenmekteydi ve buna karşılık tüm tarayıcılarca desteklenen Javascript kullanımı yaygındı. Bu durumda ürünün yaygınlaşması için Microsoft firması şu iki konuya çözüm bulma yoluna gitmiştir;

- Web sayfasının dinamik unsurları tarayıcıdan bağımsız olabilmelidir. Yani tarayıcı ne olursa olsun dinamik sayfalar üretilebilmeli ve istemciye saf HTML biçiminde gönderilebilmelidir.
- Web sunucu ve/veya veritabanı sunucuları üzerindeki veritabanları veya diğer kaynaklara erişim sağlanabilmelidir.

Microsoft firması, sorunun çözümünü, scripting teknolojisini Web sunucu platformuna taşımakta olduğuna karar vermiştir. ASP yani Active Server Pages bu şekilde ortaya çıkmıştır.

1996 Yılı başlarında “Denali” projesi ile ilk adımlar atılmıştır. Denali, ASP projesinin kod adıdır ve 0.9 Beta sürümü 1996 yılı başlarında yayınlanır yayınlanmaz büyük bir ilgi görmüştür. Kullanımı ve öğrenilmesi çok basittir. Tüm programcılarının bildiği Visual Basic dilinin uyarlanmış hali olduğundan, çok kısa zamanda öğrenilebilmekte ve tüm web tarayıcılar tarafından desteklenmektedir. Çünkü program kodları ASP olsa da sonuçlar web tarayıcılarına daima standart HTML dili

ile iletilmektedir. (<http://activez.cu.edu.tr/content/cyberspace/aspplus/introaspx1.aspx>, 2004)

ASP dilinin çıkış öyküsü, bize bu dilin ne işe yaradığını, neler yapabildiğini göstermektedir. ASP bir script dilidir. ASP yi daha iyi anlamak için “scripting” terimini anlamak gerekmektedir.

Klasik bir programlama işinde önce program yazılır. Programın bu haline kaynak kod (source code) denir. Bu kod dosyası bir derleyici (o dil için yazılan derleyici-compiler) tarafından derlenir ve derleme işleminin ardından işletilebilir program (exe dosya) elde edilir. Ardından programın bu son şekli işletim sistemi tarafından çalıştırılır. Scripting ise daha basit bir süreçtir. Script yazmak, tam bir program yazmak anlamına gelmez. Script yazmak, genellikle belli bir işi yerine getirmek için komut satırı yazmak gibi daha kısa ve doğrudan çalıştırılan program parçaları yazmak anlamına gelir. Script'ler bir program gibi derlenmezler, onlar yalnız yorumlanırlar (interpreted). Sonuç olarak, programların yazılmasında iki kavram karşımıza çıkmaktadır. "Derlenen" ve "yorumlanan" programlar. Derlenen programlar önce derlenirler, sonra işletilirler. Yorumlanan programlar ise doğrudan işletilirler.

Yazılan bir script, uygulama ya da işletim sistemi tarafından yorumlanarak işletilmektedir. Bu aynı MS-DOS ortamında bir komutun ya da .BAT dosyasının yazılması ve işletilmesi gibidir. Örneğin, DOS işletim sisteminde DIR komutu yazılıp, Enter tuşuna basıldığında sistem bu komutu anında algılamakta ve kullanıcıya sonucu sunmaktadır.

ASP programlamasında iki tür script işi karşımıza çıkar. Birincisi istemci tarafında etkileşimi sağlayacak program parçaları yazmak, ikincisi sunucu üzerindeki işlemleri yönetmek için, ASP kodu olarak belirlediğimiz alanda script yazmaktır.

İstemci tarafında yazılan script'ler tarayıcı (browser) olarak adlandırılan Microsoft Internet Explorer ve Netscape Navigator gibi programlar tarafından işletilmektedir. Diğer bir deyişle web tarayıcıları, script komutlarını anlamakta ve yerine getirmektedir. Özellikle ziyaretçi ile etkileşimde (bir düğmeye tıklanması gibi) bu tür scripting işlemi kullanılmaktadır. Burada şu iki unsuru iyi anlamak gerekmektedir;

- Client Side Scripting (İstemci taraflı script yazmak). Kod satırları satır satır işletilmektedir. Özellikle kullanıcı olaylarına karşı tepki vermek üzere yazılan bu scriptler, kullanıcının bir Submit (gönder) düğmesine tıklaması gibi belli bir olayı yorumlamaktadır. Internet Explorer üzerinde işletilen scriptler bu alana girmektedir. VBScript ve JavaScript istemci taraflı script yazmak için kullanılan dillerdir. İstemci taraflı kodların çalışması, tarayıcının çalıştığı bilgisayarın işlemcisini kullanmaktadır.
- Server Side Scripting (Sunucu taraflı script yazmak). Kod satırları sunucu (server) üzerinde çalıştırılmaktadır. ASP sayfaları bu alana girmektedir. ASP kodu sunucu üzerinde çalışmaktadır. Yazılan kod, son sayfayı HTML olarak üretir ve tarayıcıya göndermektedir. Sunucu taraflı kodların çalışması, sunucu bilgisayarın işlemcisini kullanmaktadır.

ASP tek başına bir programlama dili olmamaktadır. Özellikle Web sunucusu ve istemcisiyle birlikte çalışan bir sistemdir. İnternet ya da Intranet üzerinden verileri kullanıcılara ulaştıran bir tekniktir. Programlama olarak da Web sayfalarının HTML (HyperText Markup Language) kaynak kodları içinde yazılan (gömülü) bir scripting tekniğidir. ASP sayesinde özellikle veritabanı uygulamaları, Intranet ve elektronik ticaret siteleri yaratılmaktadır. Çünkü bu sitelerde kullanıcıların etkin biçimde veri girmesi ya da istekte bulunması gerekmektedir. Bu isteklerin veritabanlarına ulaştırılması, işlenip yanıtlanması, Web sunucusu üzerinde duran ASP sayfalarıyla yerine getirilmektedir.

İnternetin işleyişinde, web sunucusu üzerinde yer alan web sayfaları, istemcinin isteği üzerine ekranına gönderilmektedir. Bu süreç http protokolü ile çalışan tarayıcı (Browser) programı aracılığıyla gerçekleştirilmektedir. Ziyaretçi bu sayfalara bir gazete okuyormuş gibi bakmaktadır. Bu şekilde düzenlenmiş sayfalara statik sayfalar denilmektedir. Örneğin ziyaretçi www.itu.edu.tr ya da www.milliyet.com.tr adresini yazarak internet bağlantısı aracılığıyla web sunucusundan istekte bulunur. Ardından bu adresin ana sayfası kullanıcının tarayıcısına gönderilir ve ekranında görüntülenir. Dikkat edilirse bu işlem sunucudan ziyaretçiye olmak üzere tek yönlü olarak gerçekleşmektedir.

Ziyaretçinin tarayıcısı aracılığıyla ekranına getirdiği web sayfası, HTML bileşenlerinden ve grafik bileşenlerden oluşur. Ziyaretçi, tarayıcı programı aracılığıyla bu HTML etiketlerini görebilse de, sayfada değişiklik yapması bir şey ifade etmemektedir. Çünkü sayfayı yeniden istediğinde, sayfa web sunucusundan tekrar aynı şekilde gelecektir. Bu sistem bir şirketin ürünlerini tanıtacağı bir site için mantıklı olsa da, siparişlerin girileceği bir elektronik ticaret sitesi için mantıklı olmamaktadır. Bu gereksinimle dinamik sayfalar ve ASP teknolojisi geliştirilmiştir.

ASP sayfaları sunucu tarafında çalışacak kodlar içermektedir. Sunucu tarafı bu dosyalar dinamik içerikli web sayfalarının yaratılmasını sağlamaktadır. Örneğin Web sitesi üzerinde sipariş vermek ya da herhangi bir veritabanı uygulamasına web tarayıcısından erişmek gibi. Ayrıca ASP sayfalarında çok sayıda yerleşik nesne bulunabilmektedir. Bu nesneler ASP dosyalarının script olarak programlanmasını sağlamaktadır. Örneğin “Request” nesnesi kullanılarak kullanıcıdan bilgi alınabilmektedir, “Response” nesnesiyle de kullanıcıya bilgi gönderilebilmektedir.

ASP sayfaları ayrıca birtakım bileşenlerle zenginleştirilebilmektedir. Sunucu tarafı ActiveX bileşenleri veritabanlarıyla çalışmayı, elektronik posta göndermeyi ya da dosya sistemine erişmeyi sağlamaktadır. Bu anlamda ASP sayfalarıyla yapılabileceklerin sınırı yoktur.

ASP teknolojisi ile oluşturulan sayfalar, yani içinde ASP teknolojisinin gerektirdiği kodlar bulunan HTML dosyaları, .asp uzatmasıyla kaydedilmektedir. Bu dosyalar talep edildiğinde, Web Server programı, eğer ASP teknolojisini tanırıyorsa, talep edilenin ASP sayfası olduğunu, dosya adının uzatmasından anlamaktadır ve bu sayfayı ziyaretçiye yollamadan önce kendisi işleme tabi tutmaktadır. ASP sayfasındaki kodlar bir takım dinamik işler yapacaktır. (Hesaplamalar, araştırmalar, veritabanından veri çekme, dosya sisteminden dosya isteme, vs.) Daha sonra ortaya çıkacak olan HTML dosyasını ziyaretçiye gönderecektir. **(Demirkol, 2003)**

SlidePack Slayt Arşivi projemizin dinamik web sayfalarını hazırlarken, ASP scriptlerini kullanarak, tüm internet tarayıcı programlar ile veritabanına erişimini gerçekleştirmiş olduk. Aynı zamanda yine ASP sayesinde veritabanının yüklü olduğu web Serverlar, aynı anda birçok kullanıcıya hizmet verebilecektir.

7.4.5. Web Ortamında Veritabanı Sorgulama İşlemleri İçin SQL

SQL, Structured Query Language in kısaltmasıdır. Türkçemizdeki tam karşılığı Yapılandırılmış Sorgulama Dili olarak çevrilebilir. Bağlantılı veritabanlarının (Relational Database) tanımlanması, değiştirilmesi, sorgulanması amacıyla geliştirilmiş bir script dilidir. (www.rehberim.gen.tr, 2004)

SQL bazı kaynaklarda programa dili değil bir alt dil (Sub Language) olarak tanımlanmaktadır. SQL ile yalnızca veritabanı üzerinde işlem yapılmaktadır. SQL scriptleri kullanılarak veritabanına yeni kayıtlar eklenebilir, var olan kayıtlar değiştirilebilir, silinebilir ve ya da bu kayıtlardan listeler oluşturulabilir. SQL scriptleri genellikle aynı olmakla birlikte farklı veritabanı ortamlarında değişebilmektedir. Ayrıca veri tabanlarının kendilerine özgü SQL komutları da vardır. (www.pcakademi.com, 2004)

Web sayfalarının en önemli özelliklerinden birisi de, sayfanın açılma hızıdır. Eğer web sayfasında, bir veritabanından bilgi alış-verişi yapılıyorsa, hedef noktaya, yani veritabanı içindeki, sadece istenilen kayıtlara ulaşmak, sayfanın açılma hızını arttırmaktadır. İstenilen kayıtlara hızlı bir şekilde ulaşmada SQL sorgulama dili son derece başarılıdır.

7.4.6. HTML Editörü olarak Macromedia Dreamweaver MX

Macromedia firmasının ürünü olan DreamweaverMX 2004 görsel olarak web siteleri ve Internet uygulamaları geliştirmek için profesyonel bir çözüm olmaktadır. Dinamik HTML (DHTML) öğeleri ve sayfaları yaratabilme olanağı sağlamaktadır. Tablolar yaratabilmekte, katmanlar (Layers) oluşturabilmekte, hatta web sayfasına küçük animasyonlar ekleyebilmektedir. JavaScript, ASP ve CSS (Cascading Style Sheets) için tam destek vermektedir. (Towers, 2001)

Dreamweaver MX 2004 diğer web geliştirme araçları olan Macromedia Fireworks ve Macromedia Flash, Microsoft Office, e-ticaret geliştirme ve sunucularıyla uyumlu çalışmaktadır. Dreamweaver, gelişmiş siteler hazırlanabilmesi için özelleştirilmiş HTML, Javascript ve XML kullanılmasını sağlamaktadır.

Html bilerek Dreamweaver kullanmak daha etkili web sayfaları oluşturulmasını sağlamaktadır, fakat birçok kullanıcı Dreamweaver'i Html öğrenmek için kullanmaktadır. Kod ve Tasarım sekmesini kullanarak bir yandan siteler yapılabilir diğer yandan da Html kodlarında meydana gelen farklılıklara göz atılabilir. Eğer herhangi bir kod ile ilgili sorun varsa, “Reference” panelinden rahatlıkla o kodun anlamına ve kullanımına ulaşılmaktadır. Ayrıca Dreamweaver MX 2004 kullanarak rahatlıkla Html ve Xhtml sayfaları oluşturulabilir, Html sayfaları Xhtml e dönüştürülebilmektedir.

Dreamweaver html kodlarına, kod görünümü, tasarım ve kod görünümü veya code inspector ile direk erişim sağlamaktadır. Bir kere kod yazıldıktan sonra bunun yansımaları normal görünümde görülebilmektedir. Ayrıca “quick” etiket editörü sayesinde görsel çalışma bırakılmadan kodlar yazılabilmektedir. Tag Inspector ile kodların içeriği görülebilmektedir ve bunlar üzerinde değişiklik yapılabilir. Kod görünümü, kodları yazarken ipuçları göstermektedir.

Dreamweaver MX 2004 Macintosh ortamında da çalışabilmektedir. Dreamweaver klasik MX OS X yüklü işletim sistemlerinde çalışmamakla beraber halen işletim sistemi 9.1 ve üstü olan işletim sistemlerinde çalışabilmektedir. **(www.macromediatürk.com, 2004)**

SlidePack Slayt arşivi projesinde HTML editörü olarak DreamWeaverMX programı kullanılmıştır. Bu program sayesinde ASP kodları son derece kolay bir şekilde HTML kodları içine yerleştirilmiştir. Aynı zamanda kullanıcı arayüzünün tasarım aşamasında, layer ve frame özellikleri sayesinde bize zaman kazandırmıştır.

Macromedia DreamWeaverMX in çok önemli bir avantajı da içerisinde, kullanışlı bir FTP programı bulunmasıdır. Bu program sayesinde tasarlanan web sayfaları uzaktaki server bilgisayarlarına gönderilebilir (UpLoading işlemi).

7.4.7. Resim ve Fotoğrafların Düzenlenmesinde Adobe Photoshop 8.0 CS

Adobe, dünyaca tanınan, dijital imaj, resim işleme, dizayn, video ve dokümantasyon düzenleme araçlarını üreten yazılım firmasıdır. Adobe firması 22 yıllık bir geçmişe

sahiptir ve en alt düzeyden en profesyoneline kadar tüm bilgisayar kullanıcılarınca bilinmektedir.

Adobe firmasının bir ürünü olan Photoshop programı, dünyanın en yaygın resim ve dijital fotoğraf işleme programıdır. Projemizdeki, gerek fotoğraflar gerekse grafikler tamamen Photoshop programı ile düzenlenmiştir.

Bu program ile resimlerin boyutları, çözünürlükleri, renkleri hatta dosya formatları tamamiyle değiştirilebilmektedir. Gelişmiş dosya gezgini, renk uyum ve eşleştirme komutları, hazır çizim araçları, sayısız boyama fırçaları, ışık-gölge düzenlemeleri, onlarca hazır resim efektleri ile kullanıcılara büyük kolaylıklar sağlamaktadır.

7.4.8. Projenin Hazırlanmasında Kullanılan Bilgisayar Donanımları

“SlidePack” - Görsel Mimari Veritabanı Slayt Arşivi projesinin geliştirilmesi esnasında, standart bir masaüstü bilgisayar ile dizüstü bir bilgisayar beraber kullanılmıştır. Bu iki bilgisayar arasında 100.0 Mbps (Mega Byte Per Second) hızında, ethernet tabanlı, yerel bir ağ kurulmuştur. Her iki bilgisayar da Windows Xp Professional işletim sistemi ile yüklüdür.

Projenin tüm grafiksel tasarımı masaüstü bilgisayarda gerçekleştirilmiştir. Dizüstü bilgisayarların monitör teknolojileri henüz tüplü monitörler kadar gelişmemiştir. Gerek gölgelendirmeler, gerek renk tutarlılıkları, gerekse kontrast ve çözünürlük özellikleri yetersiz kalmakta ve çoğu zaman hatalı sonuçlar verebilmektedir. Dizüstü bilgisayarların TFT ekran teknolojileri, günümüz koşullarında, görsel tasarım için uygun değildir. Bu nedenle tüm imajlar ve fotoğraflar masaüstü bilgisayar ve ona bağlı olarak CRT tüplü klasik bir monitörle gerçekleştirilmiştir.

Projenin asıl can alıcı bölümü olan slayt arşivi veritabanının oluşturulması ve ASP sorgulama sayfaları dizüstü bilgisayarımızda gerçekleştirilmiştir. Taşınabilir özellikli dizüstü bilgisayar, çalışma ortamı dışında da projemizi sürdürebilmemize imkan vermiş ve bize kayda değer bir zaman tasarrufu sağlamıştır.

Ağ ortamındaki, IIS yüklü olan Laptop (dizüstü) bilgisayar “Server” (Sunucu) rolünde olup, web sayfasını ağdaki diğer bilgisayara yayınlamaktadır. İçerisinde Windows ile beraber gelen web tarayıcı programı (Internet Explorer) bulunan

masaüstü bilgisayarımız ise “Client” (İstemci) rolündedir ve hazırladığımız sayfaları izlememizi sağlamaktadır. Aslında IIS yüklü olan server bilgisayarımızın kendisi de bir istemcidir. İçerisinde web tarayıcı programı olması nedeniyle hem sunucu hem de istemci olarak çalışabilir. Kendi içindeki web tarayıcı programı ile yine kendi içindeki IIS sunucuna bağlanabilmektedir.

Aşağıda, Tablo 7.1 de, her iki bilgisayarın, en temel donanımsal konfigürasyonları verilmektedir.

Tablo 7.1. : Kullanılan bilgisayar sistemlerinin konfigürasyonları

	IIS yüklü Dizüstü Bilgisayar (Sunucu ve İstemci)	Internet Explorer Yüklü Masaüstü Bilgisayar (İstemci)
Mikroişlemci	Intel Pentium 4A, 2656 MHz (5 x 531)	AMD Athlon-PECM, 1000 MHz (5 x 200)
Sistem Belleği	448 MB (DDR SDRAM)	256 MB (SDRAM)
Anakart Çipseti	SiS 650	VIA VT8363A Apollo KT133A
Grafik Bağdaştırıcı	SiS M650 (64 MB)	ATI Radeon 9200 SE (128 MB)
Depolama	40 GB, 4200 RPM, Ultra-ATA/100	40 GB, 7200 RPM, Ultra-ATA/133

7.5. Görsel Mimari Veritabanı Modelinin İşleyişi

7.5.1. Modelin Yapısal İşleyişi (Three-Tier Web Mimarisi)

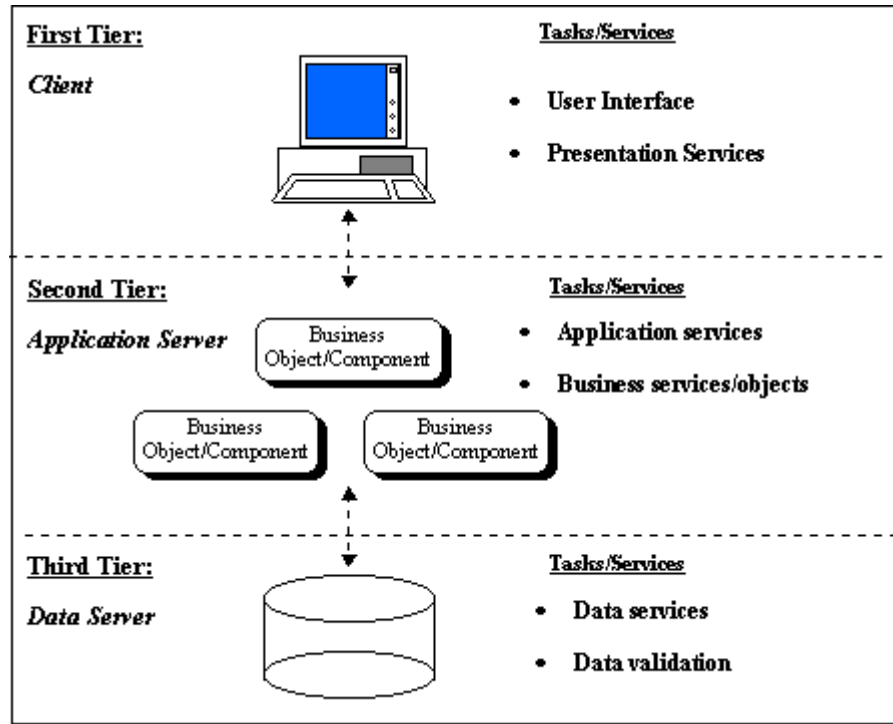
Oluşturduğumuz model, veritabanı sistemine internet ortamından erişimi sağlamaktadır. Bu tarz veritabanı çözümlerine “Üç Katmanlı Web Mimarisi” de denmektedir. Şekil 7.3’ de de görüldüğü gibi, bir three-tier uygulama, her biri bir ağın farklı bir yerine dağılmış, üç ana parçadan oluşmaktadır. Bu parçalar:

- İş istasyonu veya sunum arayüzü
- İş mantığı
- Veritabanı ve veritabanı yönetimi ile ilgili program

Tipik bir three-tier uygulamada, uygulama kullanıcısının (Client) bulunduğu katman ilk katmandır. Kullanıcıya arayüz sağlayan GUI programı bu katmanda yer almaktadır. Bizim sistemimizde GUI (Graphical User Interface) programı, web sayfası olarak tasarladığımız veritabanına erişim kullanıcı arayüzüdür.

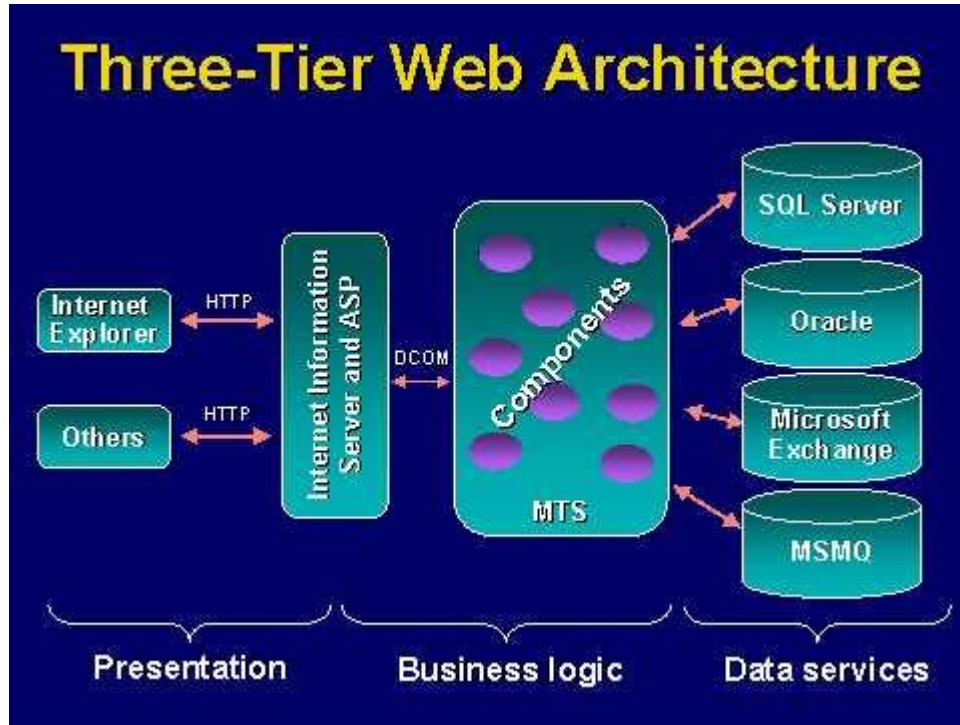
İkinci Katmanda “İş Mantığı” yer almaktadır İş mantığı, istemcilerden (kullanıcılardan) gelen istemlere sunucu olarak karşılık verir. Hangi verilerin gerekli olduğuna (ve nerede bulunduğu) karar verir ve üçüncü katman programı ile (veritabanı programı) ilişki kurar. Bu katmanda server bilgisayarlardaki IIS yazılımı işletilmektedir.

Üçüncü katman, veritabanını ve bu veritabanına okuma ve yazma erişimini yöneten programı içerir. Access ile oluşturduğumuz veritabanı yapısı bu katmanda bulunmaktadır.



Şekil 7.3. : Three-Tier yapısındaki katmanlar (<http://www.godoro.com>, 2004)

Three-tier bir uygulama istemci-sunucu modelini kullanır. Her üç katman, farklı programlama dilleri ile çalışan farklı takımlar tarafından paralel olarak geliştirilmektedir. Bir katmanın programı, diğer katmanlar etkilenmeden değiştirilebilir veya taşınabilmektedir. Böylece, yeni ihtiyaçlar doğduğunda değişiklik yapmak kolay olmaktadır. Üç katmanlı web mimari yapısındaki her bir katmanın içerdiği yazılımlar ve bu katmanların verdiği hizmetler Şekil 7.4’ de şematize edilmiştir.



Şekil 7.4. : Üç katmanlı mimari yapının bölümleri (<http://www.godoro.com>, 2004)

Bu hizmetler şu şekilde özetlenebilir;

Sunum Hizmetleri; Uygulamanın kullanıcıya veri sağlayan kısmına tekabül eder. Ek olarak, kullanıcının veri ile etkileşimini sağlayan yapıyı da sağlar. Kısaca, sunum hizmeti kullanıcı arayüzünü tanımlar ve onunla etkileşir. Kendi uygulamamızda HTML ile oluşturduğumuz web sayfaları bu hizmeti sunmaktadır.

İş Hizmetleri/Nesneleri; Uygulama hizmetlerinin bir kategorisidir. Bu hizmet, bir organizasyonlar işi işlevlerini ve gereklerini içermektedir. Bu kurallar gelen bilginin geçerli bir tip ve formatta olduğundan emin olmak için kullanılan geçerlilik kuralları veya bir işlemi tamamlamak için takip edilen uygun iş adımlarından emin olmak için kullanılan işlev kuralları olabilmektedir. (HTML sayfalarını hazırlarken kullandığımız ASP kodlarımız bu katmanda çalışmaktadır.)

7.5.2. Veritabanına Erişimde Kullanıcı Arayüzünün İşleyişi

Kullanıcılar Internet Explorer ya da herhangi bir web tarayıcı programlarını açıp, web sitesiyle bağlantı kurduklarında Şekil 7.5 da görülen ana sayfa ile karşılanmaktadır. Bu anasayfa içerisinde, orta alanda küçük bir açıklayıcı bilgi,

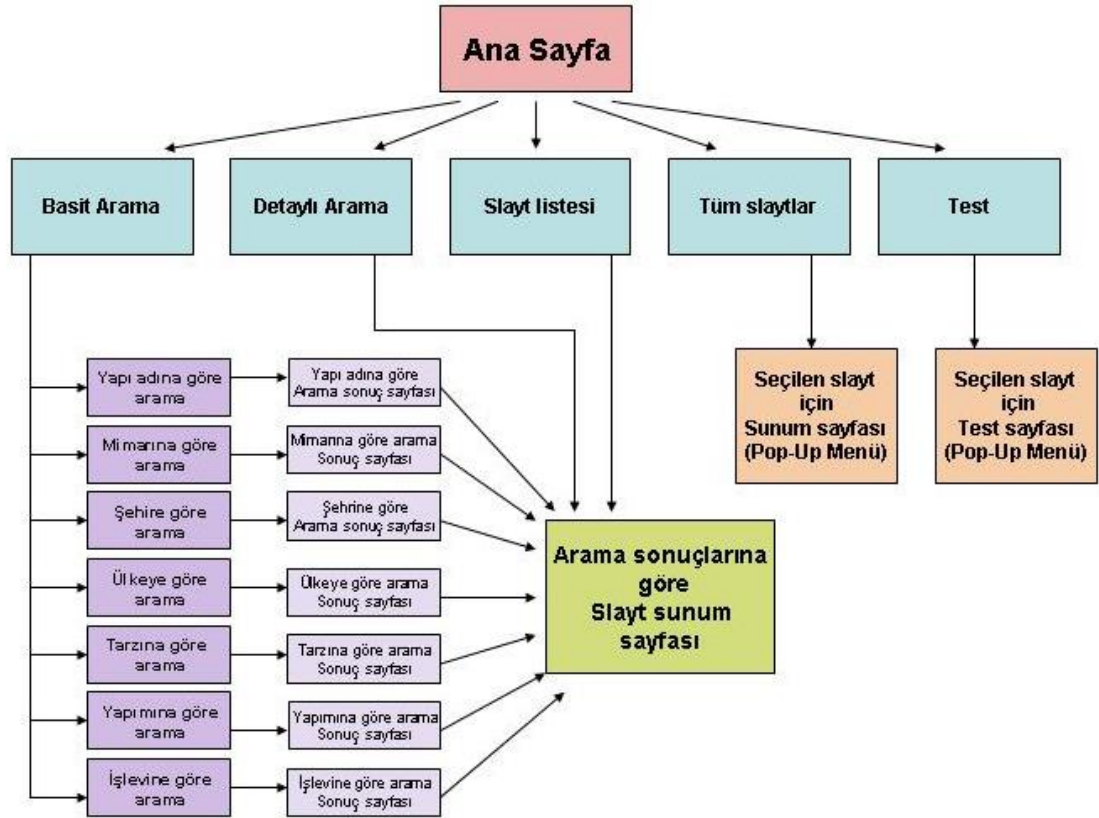
sayfanın üst kısımlarında yine kullanıcıyı bilgilendirmeye yönelik linkler (bağlantılar) ve bunların en sağında da İ.T.Ü anasayfasına bir link (bağlantı) yer almaktadır.

Sayfanın en sol tarafında ise kullanıcıları, hızlı bir şekilde arama yapmalarını sağlayan sorgulama sayfalarına bağlantılar yer almaktadır. Kullanıcılar veritabanına erişmek için, bu sorgulama sayfalarından kendilerine uygun olanını tercih etmeleri gerekmektedir.

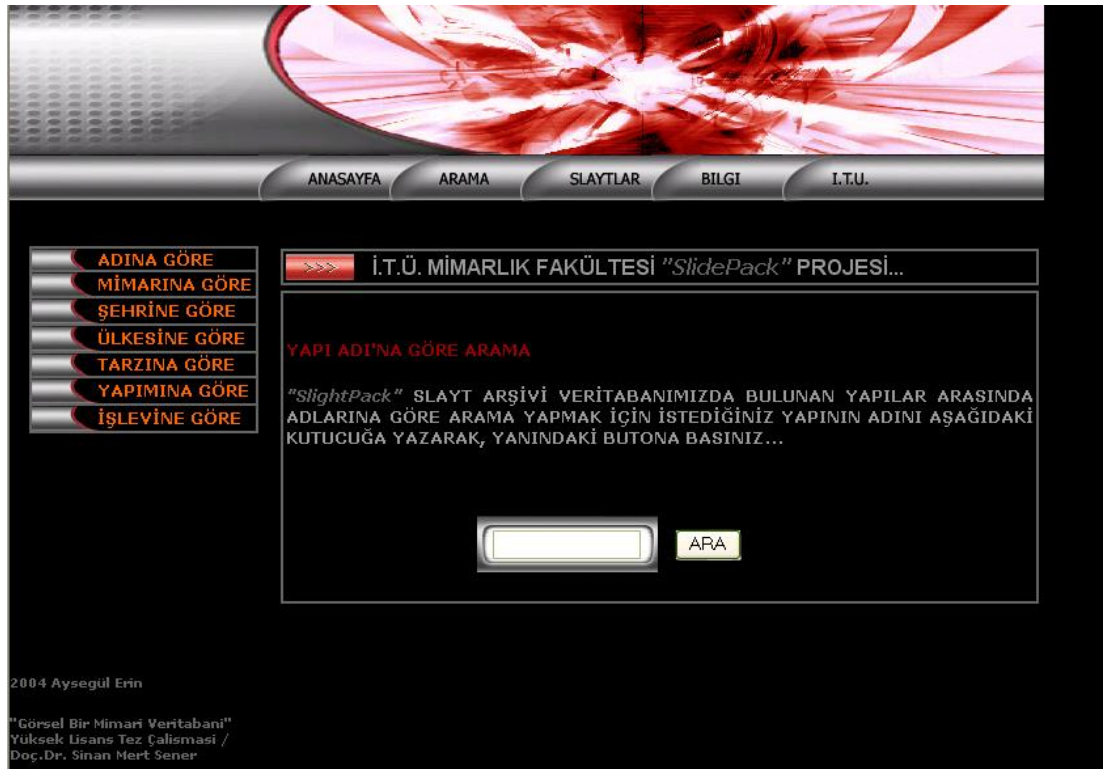


Şekil 7.5. : SlidePack Slayt arşivi ana sayfası

Basit arama, Detaylı arama, Slayt listesi, Tüm slaytlar ve Test sayfaları olarak sayabileceğimiz bu arama/sorgulama sayfalarının işleyişi ve sonuç sayfaları ile olan bağlantıları Şekil 7.6’ da şematik olarak gösterilmektedir.



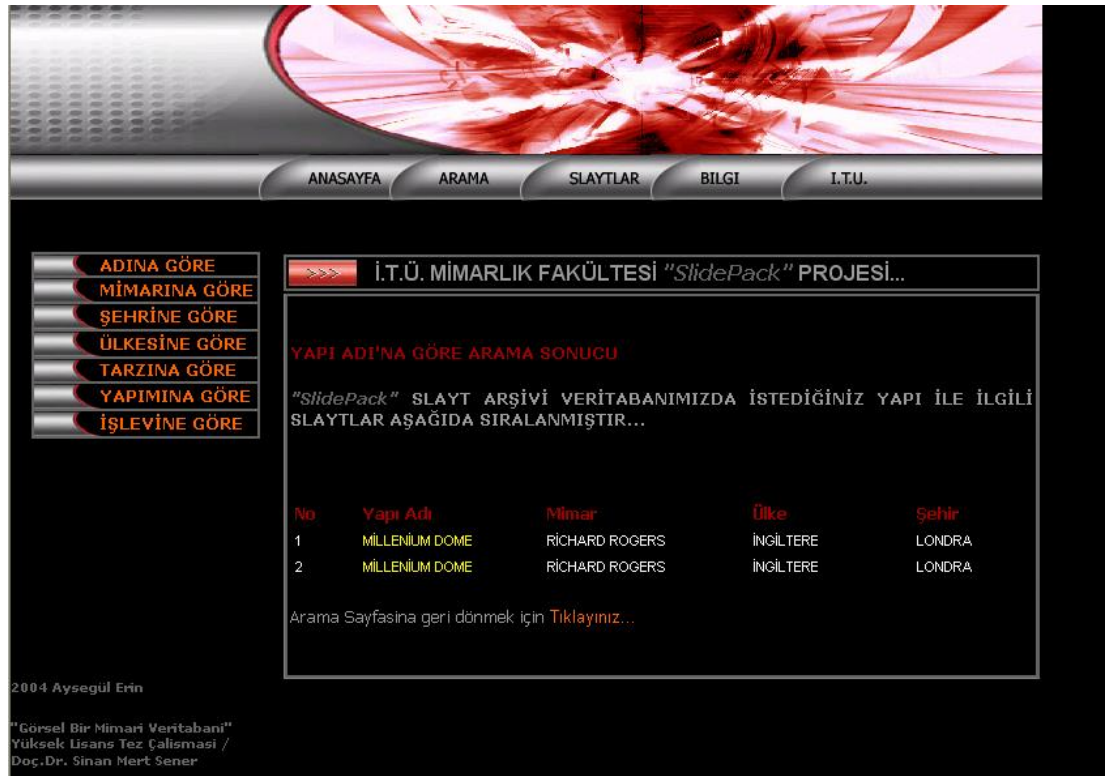
Şekil 7.6. : SlidePack Slayt arşivinin arama sayfaları arasındaki bağlantılar.



Şekil 7.7. : SlidePack Slayt arşivi yapı adına göre arama sayfası

Basit arama sayfalarında yalnızca tek bir kritere bakılarak arama yapılmaktadır. Örneğin kullanıcı aradığı slaytın sadece hangi yapıya ait olduğunu biliyorsa yani sadece yapı adını biliyorsa, basit arama sayfasındaki “yapı adına göre arama” linkine tıklamalıdır.

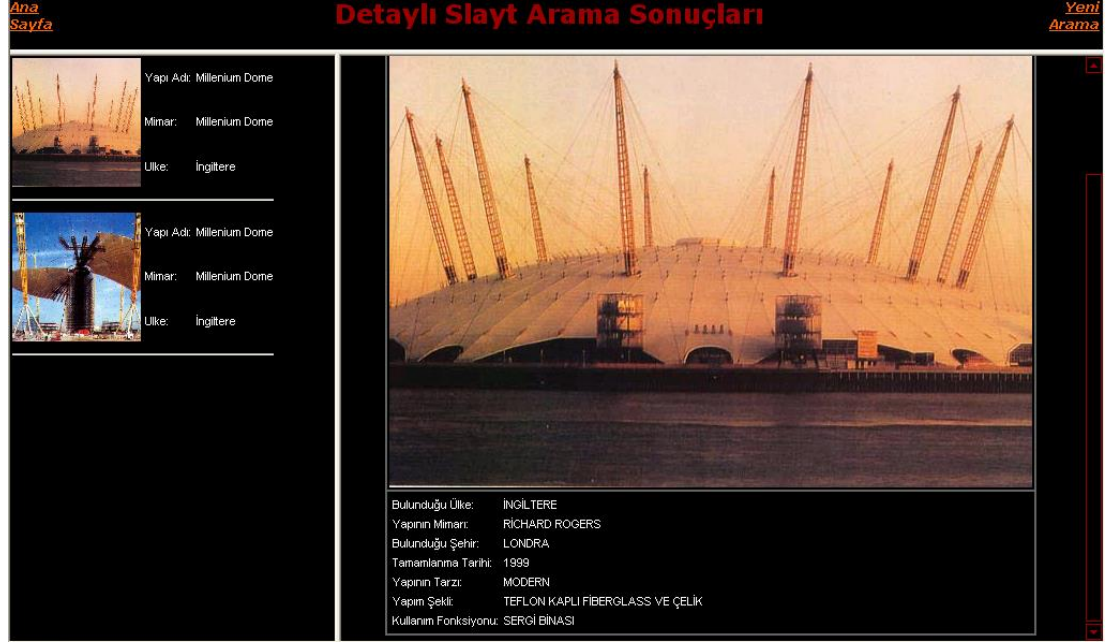
Bu şekilde yapı adına göre arama sayfasına bağlanan kullanıcının ekranına Şekil 7.7 de gösterilen menü gelecektir. Bu menü içerisindeki kısa açıklamadan da anlaşıldığı gibi kullanıcı, kendisi için ayrılmış olan veri giriş kutusuna slayta ait yapı adını yazıp “ARA” butonuna tıkladığında, sonuçlar Şekil 7.8 deki “Yapı adına göre arama – Sonuç sayfası” içerisinde sunulacaktır.



Şekil 7.8. : SlidePack Slayt arşivi yapı adına göre arama - sonuç sayfası

Arama sonuç sayfasında, girilen yapı adına ait olan tüm slaytlar listelenmektedir. Örneğin Yapı adına göre arama sayfasında, kullanıcı “Millenium Dome” ismin girip “ARA” butonuna tıkladığında Şekil 7.6’da da görüldüğü gibi veritabanında bu yapıya ait tüm slaytlar listelenecektir. veritabanında Millenium Dome için iki tane slayt bulunmaktadır.

Bu listeden Sarı renkli olan yapı isimlerine tıklanarak istenen slaytın resmi kullanıcıya “slayt sunum sayfası” içerisinde ulaştırılmaktadır. Bu sunum sayfası Şekil 7.9 de gösterilmektedir.



Şekil 7.9. : SlidePack Slayt arşivi yapı adına göre arama – sunum sayfası

Bu slayt gösterim sayfasının sol tarafında arama kriteri olarak girilen “Millenium Dome” yapısına ait veritabanındaki tüm slaytlar listelenmekte olup, sağ tarafta ise arama sonuçları sayfasında seçilen slayt gösterilmektedir.

Bu kısımda Millenium Dome yapısına ait diğer slaytın büyük resmini görmek için tekrar “arama sonuçları sayfası”na dönmeye gerek yoktur. Sol taraftaki diğer küçük slayta tıklandığında, sağ tarafta büyütülmüş resim dosyası ve bu yapı hakkındaki kısa bilgilendirme karşımıza çıkmaktadır.

Basit arama sayfasındaki diğer, mimarına göre arama, şehrine göre arama, ülkesine göre arama, tarzına göre arama, yapımına göre arama ve işlevine göre arama bölümleri yapının “adına göre arama” işlemiyle aynı şekilde yürütülmektedir.

Slidepack slayt arşivi veritabanı erişim arayüzünün en güçlü kısmı detaylı arama bölümüdür. Bu bölümde arama işlemi, yine basit aramadaki gibi tekbir kriter göz önüne alınarak yapılabildiği gibi, aynı anda kullanıcının belirleyeceği iki, üç ve hatta dört kriter birden göz önünde bulundurularak yapılmaktadır. Bu arama işlemi

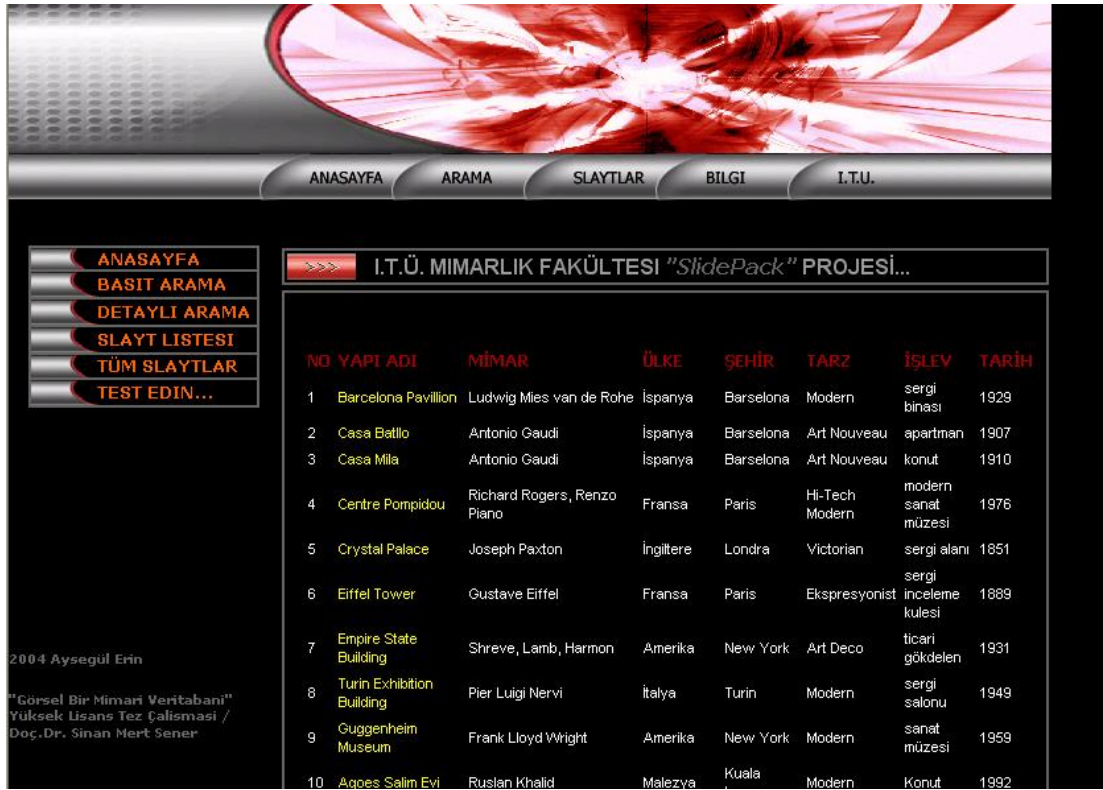
son derece seçicidir. Şayet, arama kriterlerinin kullanıcı tarafından doğru şekilde girilmesi halinde, tek bir slayt, binlercesi arasından bir seferde bulunabilmektedir. Bu detaylı arama sayfası Şekil 7.10’ de gösterilmektedir.

Şekil 7.10. : SlidePack Slayt arşivi detaylı arama sayfası

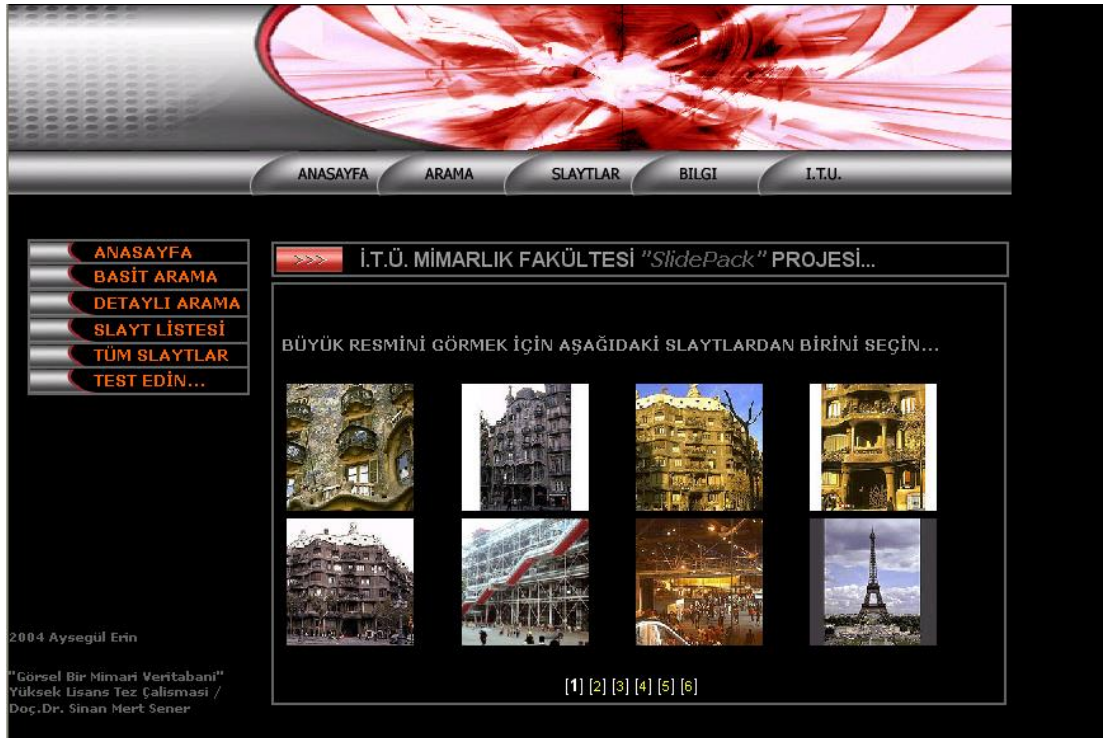
Detaylı arama sayfasında kullanıcının işine yarayacak çok önemli iki seçenek mevcuttur. Bu seçenekler ile istenirse girilen tüm kriterleri aynı anda sağlayan slaytlar listelenebileceği gibi, bu kriterlerden herhangi birini sağlayan tüm slaytlar da kullanıcıya liste halinde sunulabilir.

Arama yapmadan önce kullanıcı “Hepsi” ve “Herhangi biri” seçeneklerinden birini işaretlemiş olmalıdır. Detaylı arama sonucunda, kriterleri sağlayan slaytlar yine Şekil 7.8 de ki slayt sunum sayfasında kullanıcıya aktarılmaktadır.

Tasarladığımız bu veritabanına erişim web arayüzü, kullanıcıya hiçbir arama kriteri girilmeden de arama yapma imkanını sunmaktadır. Bunu iki şekilde sağlamaktadır. Bu yöntemlerden ilki, kullanıcı ana sayfadan “Slayt Listesi” linkine tıklayabilir ve kullanıcıya Şekil 7.11 deki menü sunulur.



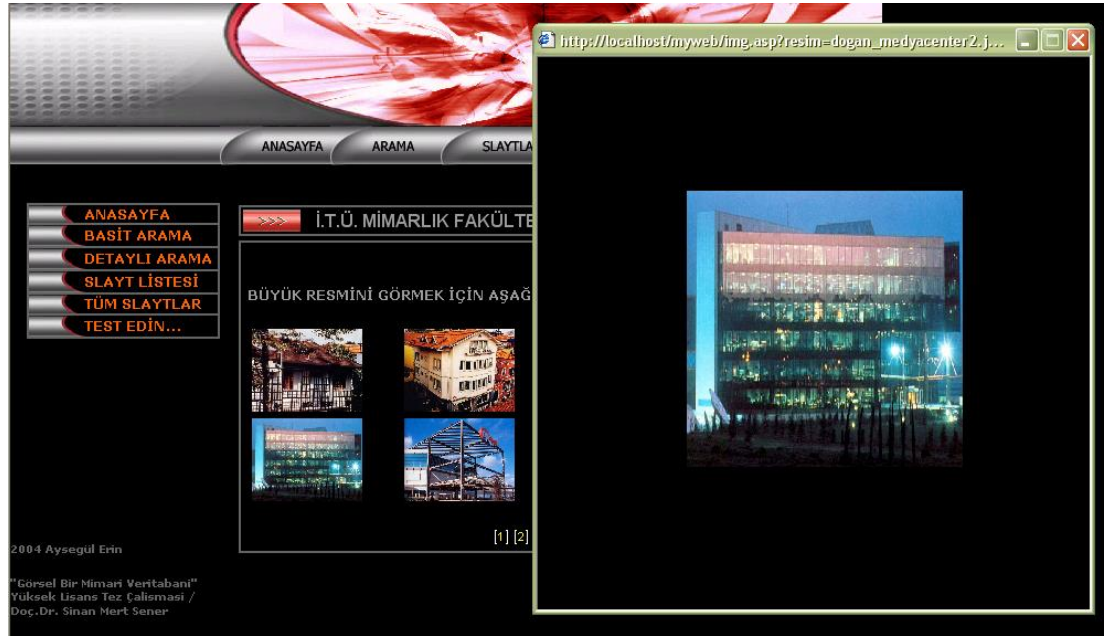
Şekil 7.11. : SlidePack Slayt arşivi “Slayt Listesi” sayfası



Şekil 7.12. : SlidePack Slayt arşivi “Tüm Slaytlar” sayfası

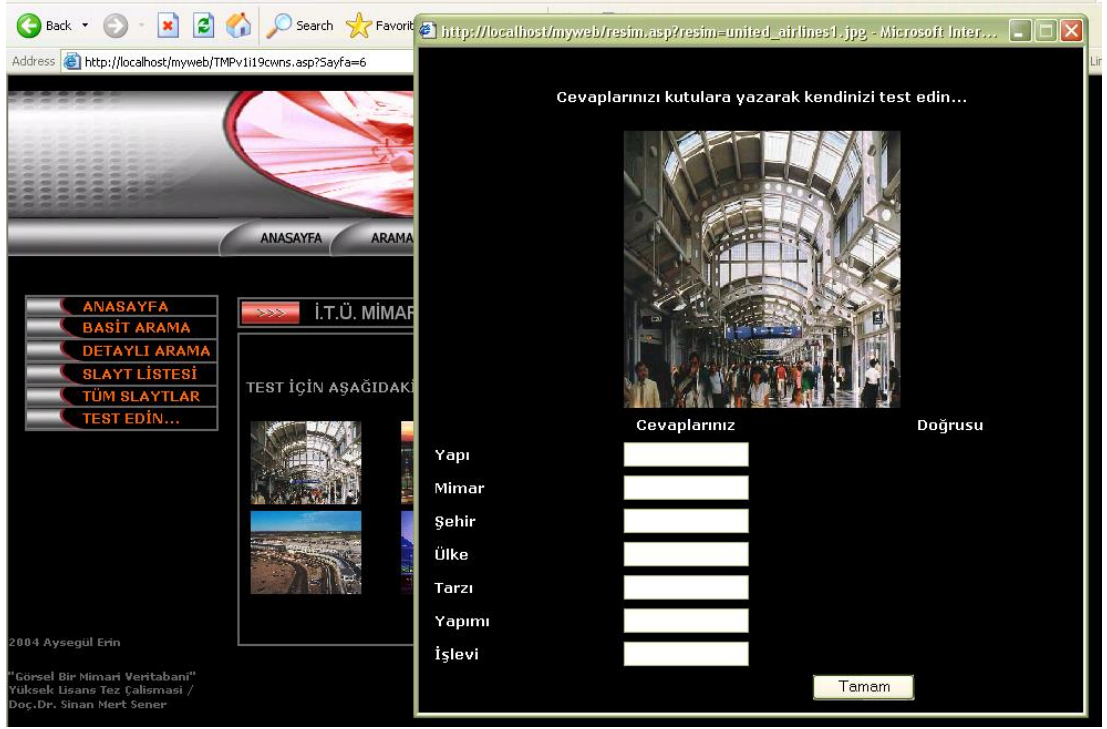
Kullanıcı bu menüde veritabanındaki tüm slaytların listesine bakarak bir tanesini seçebilir. Burada görsel bir tanımlama olmayıp, slayta ait bilgiler bulunmaktadır. İkinci yöntem ise yine ana sayfadan erişilebilen “Tüm Slaytlar” sayfasıdır. Bu seçenek, ilk seçeneğin tam tersi olarak sadece slaytların küçük imajlarını ekrana getirir ve kullanıcı bu imajlardan birini seçerek aradığı verilere ulaşabilir. Şekil 7.12 da Tüm Slaytlar sayfası gösterilmiştir.

Şekil 7.13 de görüldüğü gibi Tüm slaytlar sayfasından kullanıcı herhangi bir slayta ait resme tıkladığında, o resmin orijinal hali yeni bir web tarayıcı penceresi açılarak (Pop-Up menü) kullanıcıya sunulmaktadır.



Şekil 7.13. : SlidePack Slayt arşivi “Tüm Slaytlar” sayfasında açılan Pop-Up menü.

Slidepack slayt arşivi veritabanı erişim arayüzü içerisinde kullanıcıların ilgisini çekecek “Test” sayfası yer almaktadır. Şekil 7.14 de görülen bu web sayfasında kullanıcıya ilk aşamada kendi seçeceği slaytın resmi görüntülenir. Daha sonra kendisinden bu slayta ait yapı adı, mimar adı, şehir gibi verileri girmesi beklenir. Kullanıcı bu verileri girip “Tamam” butonuna tıkladığında, kendisine girmiş olduğu verilerden hangilerinin doğru ve hangilerinin yanlış olduğu gösterilmektedir.



Şekil 7.14. : SlidePack Slayt arşivi “Test” sayfası

8. SONUÇ

Yaptığımız bu araştırmanın sonucu olarak diyebiliriz ki, bilgisayar bilimlerinde yaşanan tüm yeni gelişmeler hızlı bir şekilde uygulamaya sokulmakta ve bu teknolojileri kullanan diğer bilim dallarının hizmetine sunulmaktadır. Ancak mimari çalışmaların görsel nitelikli olması, sanatsal değerler taşıması, mimarlığı diğer bilim dallarından farklı kılar. Bilgi-İşlem teknolojilerindeki yenilikler, diğer bilim dallarının hizmetine hızlı bir şekilde sunulurken, mimari çalışmalara destek verebilmesi için fazladan bir adım daha atması gerekir. Oluşturulan bu yeni teknolojilere görsel unsurların eklenmesi gerekliliği açısından, mimarlık ve görsel sanatlar, diğer bilimlerden farklı olarak, bilgisayar teknolojilerinin sınırlarını zorlamaktadır.

Çalışmamızın amacı, görsel nitelikli bir mimari veritabanı oluşturmaktır. Binlerce slaytı internet ortamında öğrencilere ve öğretim üyelerine sunacak, aranan bir slaytı binlercesinin içinden kısa sürede bulacak ve kullanıcıların ekranına gönderebilecek bir sistem geliştirmek için günümüzün en yeni yazılım teknolojilerini kullandık. Eğer amacımız hiçbir görsel unsur içermeyen, sıradan bir veritabanı sistemi kurmak olsaydı, kullanmış olduğumuz yazılımların bir bölümüne hiç ihtiyacımız olmayacaktı. Yine bu çalışmamızda da mimari yaklaşımdaki görsel unsurlar, bizi bir adım daha öteye gitmeye zorlamıştır.

Yaptığımız araştırmalar ile yazılım sistemlerinin, daha da önemlisi veritabanı sistemlerinin mimari çalışmalarda nasıl kullanıldığını inceledik. En eski mimari veritabanı örnekleriyle birlikte, günümüzde kullanılmakta olan görsel mimari veritabanı sistemlerini araştırdık. Daha sonra da kendi mimari veritabanı sistemimizi oluşturduk.

Oluşturduğumuz görsel mimari veritabanı modeli ile, uzun zaman zarfında, yoğun emekler harcanarak hazırlanmış olan slayt arşivini dijital ortama aktararak, öğrenci ve öğretim üyelerinin kullanımına sunmuş olduk. Böylece, artık slaytların orijinallerine ihtiyaç kalmadığından uygun saklama ortamlarına kaldırılmaları sağlanmış oldu.

Veritabanı sistemine erişimi internet ortamına aktararak, dünyanın herhangi bir yerinden bu slayt arşivine erişimi mümkün hale getirildi.

Kurulan bu veritabanı sistemi sayesinde aranan belirli bir slayt çok hızlı bir şekilde kullanıcı ekranına gelebilmektedir. Bu sayede slayt arşivi üniversitemizde, ders esnasında da eş zamanlı olarak kullanılmak üzere öğretim üyelerinin ve öğrencilerin hizmetine sunulmuştur. Kurulan veritabanı sisteminde, yalnızca slaytlar dijitalleştirilmiş resim formatı halinde kullanıcıya sunulmamıştır. Slaytlar kendilerine ait olan mimari bilgileri de içermektedir ve böylece veritabanı sistemimiz öğrencilerin çalışmalarında faydalanabilecekleri gerçek bir kaynak haline getirilmiştir.

KAYNAKLAR

- Arnheim, R.**, 1998. Visual Thinking, (Eriřim Tarihi: 21.04.2003),
<http://www.arts.uwaterloo.ca/~iperderse/Arnheim.htm>
- Adm, E.D.**, 1990. Veritabanı-Database, Evrim Basım, İstanbul.
- Bashioum, D.L.**, 1980. Interactive Systems: Calibrating the Model,
<http://portal.acm.org/citation.cfm?id=1009375.806148>, (Eriřim Tarihi: 15.04.2004).
- Bock, R.K.**, 1998. Benchmarking (Eriřim Tarihi: 15.04.2004),
<http://rkb.home.cern.ch/rkb/AN16pp/node14.html>,
- Bowers, D.S.**, 1993. From Data to Database, Chapman & Hall, Inc., London.
- Brown, A.W.**, 1991. Object Oriented Databases: Applications in Software Engineering, Mc Graw Hill.
- Buyens, J.**, 2000. Web Database Development Step by Step, Microsoft Press, Washington.
- Curtis, S.**, 2000. Response Time Measurement valuable in Timing Network Activity,
<http://www.serverworldmagazine.com/compaqent/2000/05/response.shtml>
- Çölkesen, R.**, 2003. Veri Yapıları Ve Algoritmaları, Papatya Yayıncılık., İstanbul.
- Altan, A.U.**, 2002. Görsel Bir Veritabanı Uygulaması: Boğaziçi Yalıları, *Yüksek Lisans Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.
- David, M.M.**, 1998. Universal Data Access: Fulfilling the Promise, *Intelligent Enterprise's Database On Line Programming and Design*,
<http://www.dbpd.com/vault/9812/feature.shtml>, (Eriřim Tarihi: 21.04.2003).
- David, M.M.**, 1998. Multimedia Databases Through Looking Glass , *Intelligent Enterprise's Database On Line Programming and Design*,

- <http://www.dbpd.com/vault/9705david.html>, (Eriřim Tarihi: 21.04.2003).
- Demirdađ, İ.**, 2001. Mimari Uygulama Ve Sunumlar İin Grsel Bir Veritabanı Modeli, *Yksek Lisans Tezi*, İ.T.. Fen Bilimleri Enstits, İstanbul.
- Demirkol, Z.** , 2003. ASP ile Web Programcılığı ve Elektronik Ticaret, Pusula Yayıncılık ve İletişim, İstanbul.
- Elmasri, R. and Navathe, S.B.**, 1989. Fundamentals of Database Systems, The Benjamin/Cummings Publishing Company Inc., California.
- Goldschmidt, G.**, 1997. Visual Displays for Design: Imagery Analogy And Databases of Visual Images, Visual Databases in Architecture.
- Godoro.,** Three-Tier Mimarisi nedir? (Eriřim Tarihi: 15.03.2004), http://www.godoro.com/Divisions/Ehil/Mahzen/Programming/ThreeTierArchitectural/txt/html/document_Examples.html.
- Gross, M.D.**, 1994. The Fat Pencil, The Cocktail Napkin and The Slide Library, <http://depts.washington.edu/dmachine/PAPER/ACADIA94/acadia94.html>, (Eriřim Tarihi: 02.12.2002).
- Gross, M.D.**, 1994. Recognizing And Interpreting Diagrams In Design, Proceedings of Advanced Visual Interfaces, ACM Press, New York.
- Gross, M.D.**, 1994. Drawing Analogies: Finding Visual References by Sketching, <http://depts.washington.edu/dmachine/PAPER/ACADIA94/dogross.html>, (Eriřim Tarihi: 02.12.2002).
- Gross, M.D.**, 1996. Demonstrating the Electronic Cocktail Napkin: a paper-like interface for early design, (Eriřim Tarihi: 21.04.2003), http://www.acm.org/sigchi/chi96/proceedings/demos/Groos/mdg_txt.htm
- IBM.**, 2003. Kurumsal Sistemlerdeki Veri ve Bilgileri Ynetmek, <http://www-5.ibm.com/tr/products/software/db2.html>

- Gross, M.D.,** 1997. Visual Displays for Design: Imagery Analogy And Databases of Visual Images, Visual Databases in Architecture.
- Jamil H.M. nd Zhou R.,** 2002. Towards an Interface Description Language for Relational Databases, AVI2002 (Advanced Visual Interfaces), <http://www.diel.univaq.it/avi2002/showabs.php?subid=105>, (Eriřim Tarihi: 03.12.2002).
- Harvard,.** 1993. Harvard Office For Information Technology, Doors, GSD Pilot Imaging Project in Technology Window, Harvard University.
- Koutamanis, A.,** 1995. Multimedia and Architectural Disciplines, <http://dpce.ing.unipa.it/Wwwroot/ecaade95/Indice.htm>, (Eriřim Tarihi: 25.02.2004).
- Koutamanis, A.,** 1997. Visual Databases in Architecture, <http://caad.bk.tudelft.nl/koutamanis/visual/koutamanis.htm>, Eriřim Tarihi: 21.04.2003.
- Kalıpsız, O.,** 2001. Bilgisayar Veri Tabanı Sistemleri, Der Yayınları., İstanbul.
- Karahoca, A.,** 1999. ASP İle Web Programcılığına Giriř, Beta Basım, İstanbul.
- Laufe, G.,** 1978. Design Database and Database Design, CAD 78, Third International Conference of Computer Aided Design, IPC Science And Technology Press.
- Loudon, K.C.,** 1998. Management Information Systems, Prentice Hall, Inc., New Jersey.
- Luen, E.,** 1996. The Right Tool At The Right Time, Drawing as ab Interface to Knowledge based design aids, (Eriřim Tarihi: 02.12.2002), <http://depts.washington.edu/dmachine/PAPER/ACADIA96/rt2.html>,
- Millhollon, M. and Castrina, J.,** 2003. Faster Smarter Web Page Creation, Microsoft Press, Washington.
- Muller,W. and Pasman, G.,** 1996. Typology and the Organization of Design Knowledge, *Design Studies*, sayı2,111,130.

- Muller, W. and Pasman, G.,** 1998. Management Information Systems, Prentice Hall Inc., New Jersey.
- Oktay, D.,** 1997. Access, Pusula Yayıncılık, İstanbul.
- Park, T. and Miranda, V.,** 1997. Representation of Architectural Concepts In The Study of Precedenys: A Concept Learning System, *The 16th Annual Conference of the Association of Computer Aided Design in Architecture*. Cincinnati, Ohio.
- Patel, P.,** 1998. DHTML for Web Database Applications, E-Database, Improving Your Site's Forms With Dynamic HTML, *Intelligent Enterprise's Database On Line Programming and Design*, <http://www.dbpd.com/vault/9808edb.html>, (Erişim Tarihi: 21.04.2003).
- Riccardi, G.,** 2001. Principles of Database Systems with Internet and Java Applications, Addison Wesley.
- Rosenman, M.A., Gero, J.S., Oxman, R.E.,** 1992. What's In a Case: The Use of Case Bases, Knowledge Bases And Databases In Design, *The Annual Conference of The Association For Computer Aided Design In Architecture*, Braunschweig, Wiesbaden.
- Shin, D.,** 2000. Rensselaer at Hartford Computer Science Seminar, <http://www.rh.edu/news7news00/cissem20000317.html>, (Erişim Tarihi: 03.12.2002).
- Thuraisingham-Bhavani.,** 1997. Data Management Systems: Evolution and Interoperation, Auerbach Publications.
- Tombre, K. and Paul, J.C.,** 1997. Document Analysis: A Way to Integrate Existing Paper Information in Architectural Databases, Visual Databases in Architecture.
- Towers, J.T.,** 2001. Visual QuickStart Guide Dreamweaver, Peachpit Press.
- Villareal, B.,** 2002. Access 2002 Programming By Example, QUE Corporation.
- Weldon, J.L.,** 1983. Database Administration, Plenum Press, İstanbul.

ASP'nin Çıkışı : <http://activez.cu.edu.tr/content/cyberspace/aspplus/introaspxl.aspx>

Benchmark : <http://en.wikipedia.org/wiki/Benchmark>

Dreamweaver : <http://www.macromediaturk.com>

IIS : www.bilgisayarogren.com/iidersler.htm

Html'e Giriş: http://www.geocities.com/html_rehberi/html.htm

Html Diline Giriş: <http://www.pcakademi.com/html/html-html-dilne-giris.asp>

I.I.S. Kurulum ve Yönetimi : <http://www.bilgisite.com/btez-II/btez2.iis1.htm>

SQL'e Giriş : <http://www.rehberim.gen.tr/webokulu.asp>

Three-Tier Mimari Nedir? : <http://www.godoro.com>

Veritabanlarına Giriş : www.gunduz.org/download.php?dlid=18

Veritabanı Nedir? : <http://e-dergi.emo.org.tr/index.php>

ÖZGEÇMİŞ

Ayşegül GÖNENÇ ERİN, 12 Nisan 1979'de İstanbul'da doğdu. İlk öğrenimini Erenköy İlkokul'unda tamamladıktan sonra, orta öğrenimine 1997 yılında Kartal Burak Bora Anadolu Lisesi'nde devam etti. Yıldız Teknik Üniversitesi Mimarlık Fakültesi'nden 2001 Yılında "Mimar" olarak mezun oldu. 2001 yılından itibaren İstanbul Teknik Üniversitesi "Bilgisayar Ortamında Mimari Tasarım" Yüksek Lisans programına devam etmektedir. 1997 yılı ile 1999 yılları arasında Kemer Country Dekorasyon Departman'ında çalıştı. 2000 yılı ile 2003 yılı sonuna kadar Ericsson, Turkcell, Digiturk gibi iletişim sektöründeki büyük firmaların genel müdürlüklerini yapan Erk Mimarlık İnş. Ltd. Şti.'de yönetici ve şantiye şefi mimar olarak görev yaptı. Şu anda Alice Village İnşaat ve Taahhüt Yatırım Ltd. Şti. firmasında mimari tasarım, uygulama, şantiye, satış ve satış sonrası müşteri hizmetlerinde yönetici mimar olarak görev yapmaktadır. Görsel yeniliklere ve teknoloji kullanımına çok önem vermekte ve reklama yönelik mimari çizimler, perspektif ve animasyon çalışmalarına devam etmektedir. İngilizce ve Fransızca olmak üzere iki yabancı dil bilmektedir.