

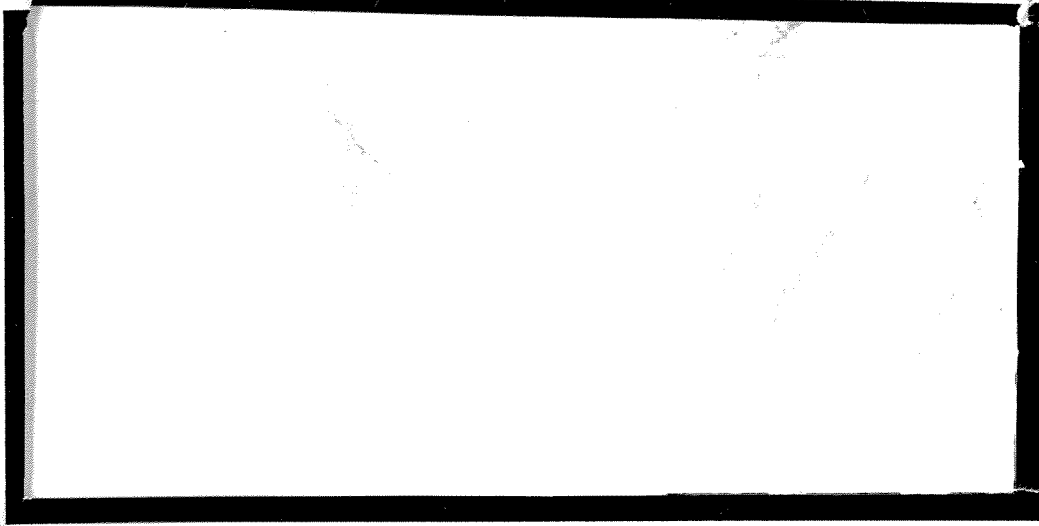


TÜRKİYE BİLİMSEL VE  
TEKNİK ARAŞTIRMA KURUMU

THE SCIENTIFIC AND TECHNICAL  
RESEARCH COUNCIL OF TURKEY

04P

2



2004/171

Elektrik, Elektronik ve Enformatik Araştırma Grubu

Electric, Electronics and Informatics Research  
Grant Committee

**GÖRÜNTÜ VE VIDEO VERİTABANLARINA  
ETKİLİ ERİŞİM SİSTEMİ**

**PROJE NO : 100E016**

**DOÇ.DR. GÖZDE BOZDAĞI AKAR**

**ARALIK 2003  
ANKARA**

## ÖNSÖZ

Bu çalışmada imge içeriğine göre imge sorguları oluşturan bir araç önerilmektedir. Her imgeye ait bir öznitelik vektörü veritabanında kaydedilir. Öznitelik vektörü renk, pozisyon ve metadata bilgilerinden oluşur. Veritabanı erişimi ve kullanıcı arayüzü ise JAVA Servlet/ VBScript ve ASP/ Applet kullanılarak oluşturulmuştur.

Bu proje ODTÜ Elektrik Elektronik Mühendisliği Bölümünde gerçekleştirilmiş olup TÜBİTAK EEEAG tarafından desteklenmiştir.

## İÇİNDEKİLER

ÖZ .....	6
ABSTRACT .....	7
GİRİŞ .....	8
<b>Video veritabanı erişim sistemleri</b> .....	8
ASP .....	9
JAVA SERVLET/JSP – ASP Karşılaştırması .....	11
<b>Öznitelik vektörlerinin çıkarılması</b> .....	16
Renk uzayı .....	17
Renk histogramı .....	17
Ana renkler .....	17
<b>Öznitelik vektörlerinin saklanma yöntemi</b> .....	17
<b>Veritabanı yapıları</b> .....	18
RENGE DAYALI ÖZNİTELİK ÇIKARMA .....	19
GELİŞTİRİLEN VERİTABANI YAPISI .....	22
SONUÇLAR VE DEĞERLENDİRME .....	29
<b>Performans</b> .....	35
PROJENİN GERÇEKLENMESİNDE KARŞILAŞILAN ZORLUKLAR .....	38
REFERANSLAR .....	39
EKLER .....	42
<b>Ek 1</b> .....	42

## TABLO LİSTESİ

**Tablo 1:** Öznitelik vektörleri.

**Tablo 2:** PATH ve NODE tabloları.

**Tablo 3:** Örnek PATH ve NODE tablosu

## ŞEKİL LİSTESİ

**Şekil 1:** QBIC imge arama sistemi

**Şekil 2:** ASP arama kütüphanesi

**Şekil 3:** Örnek XML dosyası.

**Şekil 4:** Baskın renk tespitinde kullanılan komşuluk alanı

**Şekil 5:** Baskın renk ayrıştırma

**Şekil 6:** Mpeg 7 dosyasının veritabanında saklanması

**Şekil 7:** Giriş sayfası.

**Şekil 8:** Veritabanı tiplerinin gösterildiği sayfa.

**Şekil 9:** Üzerinde arama yapılacak olan imge.

**Şekil 10:** Arama sonuçları.

**Şekil 11:** Yeni bir imge üzerinde arama sonuçları.

**Şekil 12.** Baskın renk özelliğine göre karşılaştırma.

## ÖZ

Bilgisayar yazılım ve donanımındaki son gelişmeler elektronik bilginin daha kolay bir şekilde üretilmesi, işlenmesi ve saklanması sağlamıştır. Elektronik bilgi başta sadece yazılı metinden ibaretken, giderek artan bir oranda grafik, imge, animasyon, video, ses ve diğer çoğul ortam verileri de bu kapsama dahil olmaktadır. Bu verilerin çoğuna , hızlı ağlar, arama makineleri ve gözetme araçları vasıtasıyla WEB üzerinden erişilebilmektedir. Fakat verilere ulaşma amaçlı yapılan sorguların çoğu istenilen verilerden daha çok ilgisiz verileri listelemektedir. Çoğul ortam verilerinde durum daha kötüdür. Geleneksel çoğul ortam erişim yöntemleri arama yapan kişinin verdiği anahtar sözcüklere dayanması nedeniyle, verimlilikten uzaktır. Bu sebepten dolayı sayısal görüntülerin içerik tabanlı erişimi veritabanı yönetiminde aktif bir araştırma konusudur. Büyük çoğulortam veritabanlarında veri arama konusundaki diğer önemli bir nokta da kullanıcı arayüzünün kolay kullanılabilir ve yüksek etkileşimli olmasıdır. Bunun yanısıra sistemin geniş ulaşılabilirlik amacıyla çok platformlu olması da gerekir. Literatürde bilinen içerik tabanlı indeksleme ve erişim araçlarının çoğu JAVA APPLET ve CGI-BIN temellidir. Fakat bu tekniklerin, yavaş çalışması ve/veya sınırlı sayıda kullanıcıya ulaşabilmesi gibi dezavantajları vardır. Bu sorunları çözmek için son yıllarda önerilen iki teknoloji ASP (Active Server Pages) ve JAVA Servlet'dir. Bu tekniklerin kullanıcı makineye sadece basit HTML kodlanmış bir sayfa göndermesi ve erişim sürecinin sonuçlarının herhangi bir gözetme aracı kullanılarak izlenebilmesini olanaklı kılması düşünülmektedir. Bu nedenlerden dolayı bu projede JAVA Servlet ve ASP'ye dayanan içerik tabanlı bir veritabanı erişim sistemi geliştirilmiş, performans değerlendirmeleri yapılmıştır.

### **Anahtar Sözcükler:**

Çoğulortam, veritabanı, ASP, Servlet

## **ABSTRACT**

Due to the recent developments in hardware and software, it becomes much more easier to generate, process and store electronic information. Being only text at the beginning, this electronic information now includes graphics, images, animations, video, audio and other multimedia data at an increasing rate every year. Most of this information can be accessed through WEB by the help of high speed networks, search engines and browsing tools. However, most queries result in many retrieved documents, only some of which are relevant. In case of multimedia data the problem is even worse. Since traditional methods of retrieving multimedia information depend on a number of descriptive keywords given to the data by the person indexing it, these methods are inefficient. Due to this fact, content based retrieval of digital imagery became an active area of research in the field of database management .

Another issue in search of information from large multimedia databases is that, the interface should be user friendly and highly interactive. Also the system should be able to work on different platforms for wide accessibility. The tools that are found in the literature for content based indexing and retrieval are mostly based on JAVA and CGI-BIN applications. However these technologies have drawbacks which make the tools either slow or reach a limited group of users. The disadvantage of CGI-BIN applications is that the executable codes must be put in a specified place where every user does not have the write permission whereas JAVA has the disadvantage of slow execution time because of applet's execution in the client side.

In this project we developed systems based on Java Servlet and Asp and make comparisons.

### **Keywords:**

Multimedia, database, ASP, Servlet

## GİRİŞ

Bilgisayar yazılım ve donanımındaki son gelişmeler elektronik bilginin daha kolay bir şekilde üretilmesi, işlenmesi ve saklanmasını sağlamıştır. Elektronik bilgi başta sadece yazılı metinden ibaretken, giderek artan bir oranda grafik, imge, animasyon, video, ses ve diğer çoğul ortam verileri de bu kapsama dahil olmaktadır. Bu verilerin çoğuna , hızlı ağlar, arama makineleri ve gözetme araçları vasıtasıyla WEB üzerinden erişilebilmektedir. Fakat verilere ulaşma amaçlı yapılan sorguların çoğu istenilen verilerden daha çok ilgisiz verileri listelemektedir. Çoğul ortam verilerinde durum daha kötüdür. Geleneksel çoğul ortam erişim yöntemleri arama yapan kişinin verdiği anahtar sözcüklere dayanması nedeniyle, verimlilikten uzaktır. Bu sebepten dolayı sayısal görüntülerin içerik tabanlı erişimi veritabanı yönetiminde aktif bir araştırma konusudur [6-13].

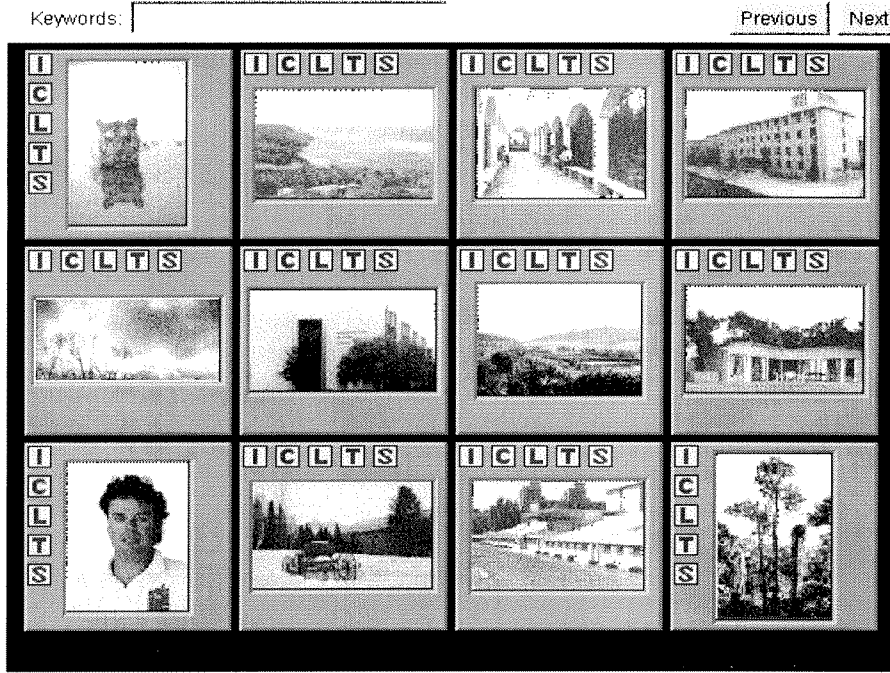
Büyük çoğulortam veritabanlarında veri arama konusundaki diğer önemli bir nokta da kullanıcı arayüzünün kolay kullanılabilir ve yüksek etkileşimli olmasıdır. Bunun yanısıra sistemin geniş ulaşılabilirlik amacıyla çok platformlu olması da gerekir.

İlerleyen bölümlerde literatürde bulunan örnek video veritabanı yapıları, erişim sistemleri, arayüz tasarımları hakkında bilgi verilecektir.

### Video veritabanı erişim sistemleri

Literatürde içerik tabanlı veritabanları kullanılan erişim tekniği, veritabanı yapısı ve kullanılan nitelik vektörleri açısından çeşitli sınıflara ayrılabilir [1,6,10]. Var olan bu sistemlere örnek olarak en popüler ve de veritabanı literatüründe baz teşkil etmeleri açısından QBIC ve de WEBSEEK verilebilir (Şekil 1) [9,10]. Bu sistemler de dahil olmak üzere önerilen veritabanı erişim sistemlerinin büyük bir kısmı JAVA ve CGI-BIN temellidir Fakat bu tekniklerin, yavaş çalışması ve/veya sınırlı sayıda kullanıcıya ulaşabilmesi gibi dezavantajları vardır. CGI-BIN uygulamalarının dezavantajı, çalıştırılabilir program kodlarının, her kullanıcının yazma yetkisi bulunmayan belirli yerlere yazılması zorunludur. JAVA uygulamalarındaki sorun ise applet'lerin kullanıcı tarafında çalışmasından kaynaklanan yavaşlıktır. Bu teknolojilerde bulunan problemler göz önünde bulundurularak ortaya çıkan yeni bir yöntem ise ASP (Active Server Pages) dir [26]. ASP Microsoft tarafından geliştirilen, WEB programlamasına devrimsel bir vizyon getiren ve daha önce değinilmiş olan sorunları çözme yetisine sahip yeni bir teknolojidir. Bu yaklaşımdaki ana fikir program kodların sunucu bilgisayarda çalışması ve üretilen HTML kodlanmış sayfanın kullanıcı makineye gönderilmesidir. Bunların yanısıra, ASP Sunucu bilgisayar üzerindeki herhangi bir veritabanı için ODBC bağlantısı sağlamaktadır. Kullanıcı makineye sadece basit HTML kodlanmış bir sayfa gönderilmesi, erişim sürecinin sonuçlarının herhangi bir

gözetme aracı kullanılarak izlenebilmesini olanaklı kılmaktadır. Benzer yapıdaki diğer bir teknoloji ise JAVA/SERVLET'lerdir [29,30]. İleride daha detaylı anlatılacağı gibi SERVLET'ler sık erişilen sayfalarda ASP'ye oranla daha iyi performans gösterdiklerinden çoğulortam dünyasında daha ilgiyle takip edilmekte ve daha sıklıkla kullanılmaktadır.



Query was:

Example: =tiger.jpg

Query Type: Color Histogram

Şekil 1: QBIC imge arama sistemi.

## ASP

ASP, Microsoft tarafından geliştirilen ve WEB programlamasına yeni bir vizyon getiren bir teknolojidir [26]. Bu teknolojiye temel farklılık, özel kodların sunucu tarafında çalışması ve neticede oluşturulan HTML kodlu sayfaların istemci tarafına yollanmasıdır. ASP tabanlı kodlar VBScript kullanarak doğrudan HTML kodlu bir sayfanın içine yazılabildiği gibi, VB 5+ kullanarak bir DLL haline getirilip daha sonra bunlara sayfa içinden nesne veya bileşen oluşturularak ulaşılabilir. ASP nesneleri ve bileşenleri ActiveX bileşenlerinden çok farklı değildir. ASP nesneleri VBScript'in her zaman kullanımına açık olan ActiveX elemanlarıdır (OLE nesneleri). ASP, Application, Session, Request, Response, Server nesnelerini destekler. ASP bileşenleri ise, ASP yapısı dışındaki DLL'lerdir. Bu bileşenler herhangi bir programlama dili tarafından oluşturulabilirler. ASP "Database access", "Browser Capabilities", "Ad Rotator" ve "Content Linking" bileşenlerini destekler.

Bütün ASP bileşenleri arasında en yararlı olanı "Databases Access" bileşeni, diğer bir adıyla ActiveX Data Objects (ADO)'dur. WEB üzerindeki veritabanı ulaşımı bu bileşen ve bunun içindeki nesneler

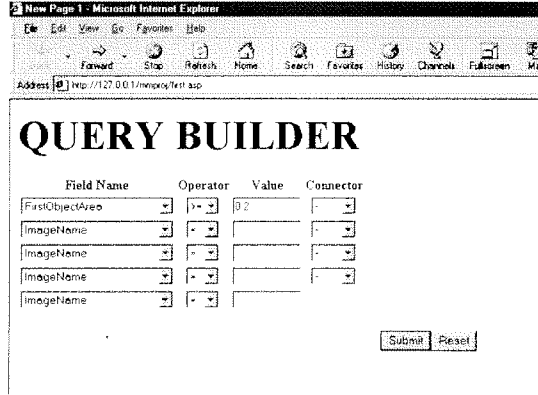
kullanılarak yapılır. Bunlar sayesinde Open Database Connectivity (ODBC) veri kaynaklarına OLE DB nin ODBC veri sağlayıcısı kullanarak ulaşılabilir. “Connection” nesnesi kullanılarak veritabanı ve program arasında bir bağlantı kurulur. Veri kaynağına bağlantı açıldığı zaman da, “Recordset” nesnesi kullanılarak herhangi bir sorgunun sonucu edilebilir.

Geliştirilen sistemde bir WEB sunucusuna veri yollamak için, istemci tarafında bir form doldurulması gerekmektedir. Kullanıcı kriterleri seçince, sorguya esas imgenin karşılık düşen değerleri otomatik olarak bir ‘Request’ nesnesine yüklenir (Tablo 1). Bu veri daha sonra Şekil 2’te gösterilen ASP algoritma kütüphaneleri tarafından işlenir. Daha sonra elde edilen resimler ve bunların öznitelik vektörleri gözetme aracına yollanır. Eğer bir kullanıcı başlangıçta hiç bir imge seçmeden kendi vereceği değerlerle bir tarama yapmak istiyorsa, o zaman Sorgu Oluşturma Aracını kullanarak, bu isteğini gerçekleştirebilir.

Burada dikkat edilmesi gereken nokta tüm çalışma boyunca dominant renklerin, alanlarının ve /veya histogram değerlerinin öznitelik vektörü oluşturulmada kullanılmasıdır.

Order	Field Name	Data Type
1	ImageName	String[255]
2	FirstObjectArea	Float
3	SecondObjectArea	Float
4	ThirdObjectArea	Float
5	FirstObjectRAverageColor	Byte
6	FirstObjectGAverageColor	Byte
7	FirstObjectBAverageColor	Byte
8	SecondObjectRAverageColor	Byte
9	SecondObjectGAverageColor	Byte
10	SecondObjectBAverageColor	Byte
11	ThirdObjectRAverageColor	Byte
12	ThirdObjectGAverageColor	Byte
13	ThirdObjectBAverageColor	Byte
14	FirstObjectCenterofMass	Float
15	SecondObjectCenterofMass	Float
16	ThirdObjectCenterofMass	Float

**Tablo 1:** Öznitelik vektörleri.



**Şekil 2:** ASP arama kütüphanesi

## **JAVA SERVLET/JSP – ASP Karşılaştırması**

Bu proje kapsamında ağırlıklı olarak JAVA Servlet ve JSP tabanlı sistemler üzerinde durulmuştur. Bunun nedeni de ASP ile karşılaştırması verildiğinde açıkça görülebilir. Karşılaştırma iki ana başlık altında yapılabilir: Programlama dili: Java'ya karşı Visual Basic; Sistem yapısı: Servlet/JSP'ye karşı ASP.

### **Java - Visual Basic Karşılaştırması**

Visual Basic dünyanın en çok kullanılan programlama dillerinden biridir. Öğrenmesi ve Windows ortamında program geliştirmesi oldukça kolaydır. Buna karşı Java dili tarafından sunulan bazı özellikler (Inheritance, encapsulation, abstraction, exception handling) Visual Basic'te bulunmamaktadır.

Inheritance kodun varolan fonksiyonlarının üst sınıftan alınıp alt sınıfta geliştirilerek tekrar kullanılmasını sağlar. Java 'abstraction'ı arayüzler ile sağlamaktadır. Arayüz karmaşık mantığın basit bir görünümle ifade edilmesini sağlar. Örneğin arayüz yol atamaya arabağ sağlayan bir ağ yazılımı olup bunu değişik algoritmalar kullanarak gerçekleştirebilir. Yazılım kullanıcıya tek bir arayüz sağlamasına rağmen protokole bağlı olarak hangi algoritmanın kullanacağını seçebilir.

'Encapsulation' bilginin basitlik için değil de başka nedenlerden dolayı saklanmasıdır. Örneğin aynı ağ örneğini düşünürsek, kullanıcı kimsenin kendi gerçekleştirmesini görmesini istemiyorsa o zaman 'Encapsulation' kullanabilir. Son olarak 'exception handling' uygulamalara beklenmeyen durumlarda koşmaya devam edip sorunları çözme imkanı tanır.

## Sistem karşılaştırması

1995’de SUN Sun Microsystems tarafından geliştirilen JAVA ortamı iki parçadan oluşmaktadır: Java programlama dili ve JAVA çalıştırma ortamı. Java programlama dili bir önceki bölümde belirtilen tüm özellikleri taşır. Java çalıştırma ortamı ise yaklaşık tüm donanım ve işletim sistemlerinde yaratılan uygulamaların çok az veya hiç değişiklikle çalışmasını sağlar.

### Servlet

Sun Microsystems’e göre Java Servlet’ler JAVA’da yazılmış ve JAVA yetkili sunucuların kapsamını arttıran protokol ve ortamdan bağımsız sunucu öğeleridir. Java Servlet’ler HTTP ve iyi bilinen GET ve POST metodları etrafında bir API (Application Program Interface) seti sağlamasına karşın bu herhangi başka bir protokol için de değiştirilebilir. Java servlet API’ları oturum yönetme, ‘cookie’ takip, ‘thread’ modelleme, diğer API’lara kolay erişim gibi çeşitli işlevsellikler sağlamaktadır.

### ASP

Visual basic sözdizimi ile ASP programlaması oldukça kolaydır. Buna rağmen Visual Basic’in belirtilen problemleri vardır. Ayrıca ASP tam bir programlama dili değildir. ASP uygulamaları çalıştığı zaman ASP sayfaları her kullanıcı erişimde ayrı ayrı ayrıştırılır. Bu da JSP’ye göre ASP’nin çoklu erişim durumunda performansını düşürmektedir. Ayrıştırma işlemine aşağıdaki örneği verebiliriz.

#### Ör : Basit bir ASP Kodu

```
<%  
Dim s = ‘Merhaba’  
  
response.write s  
%>
```

#### Denk JSP Kodu

```
<%  
String s = “Merhaba”;  
  
response.println(s);  
%>
```

Bu kodu içeren bir JSP sayfasına erişildiği zaman ilk olarak sayfa ayrıştırılarak bir java kaynak kodu yaratılır. Daha sonra Servlet mekanizması java derleyicisini çalıştırarak kaynak kodunu derler. Problemsiz derlem gerçekleşirse yaratılan sınıf dosyası önbelleğe konur. JSP sayfasına bundan sonraki

erişimlerde önbellekteki bu dosya kullanılır ve ayrıştırma işlemi tekrar yapılmaz. Apache Tomcat 4.0 kullanılarak örnekteki kod için yaratılan önbellek kodu aşağıdaki gibidir [30].

```
package jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Vector;
import org.apache.jasper.runtime.*;
import java.beans.*;
import org.apache.jasper.JasperException;

// Tomcat JSP derleyicisi (Jasper) tarafından yaratılan servletler HttpJspBase sınıfının mirasçısıdır.
// Tomcat Jasper hakkında detaylı bilgi
// http://jakarta.apache.org/tomcat/tomcat-4.1-doc/jasper-howto.html adresinde bulunabilir.

public class _0002fjsp_0002faaa_0002ejspaaa_jsp_0 extends HttpJspBase {

    static {
    }

    public _0002fjsp_0002faaa_0002ejspaaa_jsp_0( ) {

    }

    private static boolean _jspx_inited = false;
    public final void _jspx_init() throws JasperException {

// Jsp sayfamızın basitliğinden dolayı _jspx_init() metodu işlevsizdir. Bu metod Tomcat
// JSP derleyicisi Jasper tarafından standard olarak yaratılmaktadır.

    }
```

```

public void _jspService(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {

    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    String _value = null;
    try {

        if (_jspx_inited == false) {
// jsp_inited bayragini kullanarak Tomcat JSP derleyicisi (Jasper), jspx_init()
// methodunun sadece bir kere çağrılmasını sağlar.
            _jspx_init();
            _jspx_inited = true;
        }
        _jspxFactory = JspFactory.getDefaultFactory();
        response.setContentType("text/html;charset=ISO-8859-1");
        pageContext = _jspxFactory.getPageContext(this, request, response,
            "", true, 8192, true);

        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();

//Apache Jasper derleyicisi JSP kodunu java koduna dönüştürmekte ve hangi
// java kodunun jsp sayfasındaki hangi satirlara denk geldigini de yorum olarak
// java koduna eklemektedir.
        // begin [file="C:\\tools\\tomcat\\webapps\\examples\\jsp\\aaa.jsp";from=(0,2);to=(4,0)]

        // JSP sayfasında yazdığımız Java kodu servlet içinde uygun yere yerleştirilir.
        String s = "Merhaba";

```

```

        out.println(s);
    // end
    // HTML
// begin [file="C:\\tools\\tomcat\\webapps\\examples\\jsp\\aaa.jsp";from=(4,2);to=(5,0)]
        out.write("\r\n");
    // end

    } catch (Exception ex) {
// hata yakalandığı takdirde çıktı tamponu temizlenir, ve hata pageContext nesnesine
// delege edilir
        if (out != null && out.getBufferSize() != 0)
            out.clearBuffer();
        if (pageContext != null) pageContext.handlePageException(ex);
    } finally {
// hata olsada olmasada tampondaki verileri çıktıya yaz
if (out != null) out.flush();
// kullanılan kaynakları serbest bırak
        if (_jspxFactory != null) _jspxFactory.releasePageContext(pageContext);
    }
}
}

```

Örnekden görüldüğü gibi Jsp tüm özelliklere sahip bir Servlet gibi çalışmaktadır. Bu da Java programlama dilinin bütün özelliklerine ulaşmayı sağlamaktadır. Aynı zamanda tek bir kullanıcı erişimi sonrası kodun hazırlanması çoklu erişimlerde büyük avantaj getirmektedir. Buna karşın, ASP dosyalarının her erişimde işleniyor olması performans açısından dezavantajdır. ASP eksperleri ve Microsoft bu dezavantajı yok etmek için koddaki boş satırların mümkün olduğunca az kullanılmasını tavsiye etmektedirler. Örneğin:

JSP için:

```
int a=3;
```

```
String s="a";
```

ve

```
int a=3;  
String s="a";
```

kodunda boşlukların olmasıyla olmaması arasında bir fark yokken, ASP'de

```
dim a=3
```

```
dim s='a'
```

ve

```
dim a=3
```

```
dim s='a'
```

aynı değildir.

Microsoft ASP'deki bu problemleri gidermek için son zamanlarda ASP.NET adlı yeni bir sistem üzerine çalışmaktadır. JSP ile benzer yapıya sahip olması beklenen bu ortamda aynı zamanda Java'nın rakibi C# dili ile de programlar yazılabilecektir.

Bu dezavantajlar giderilse dahi JSP ve servlet'lerin ASP'ye karşı iki avantajı daha vardır: Güvenlik ve taşınabilirlik. Microsoft'un WEB sunucusu (IIS – Internet Information Server) saldırılara karşı çok açıktır. Neredeyse her ay yazılıma ilişkin bir açık nokta bulunmaktadır. Buna karşın Java ortamı son derece güvenlidir. JSP ve servlet'lerin diğer bir avantajı da istenilen herhangi bir ortam ve işletim sisteminde çalışabiliyor olmasıdır.

Bu nedenlerden dolayı bu çalışmada JSP ve servlet tabanlı sistemler üzerinde durulmuş ve geliştirilen veritabanı bu sistemlerde kullanılmıştır.

### **Öznitelik vektörlerinin çıkarılması**

Çoğulortam veritabanları için bir diğer önemli nokta da saklanan veriler için kullanılan öznitelik vektörleridir. Bu vektörler durağan imgeler için renk, şekil, doku olabildiği gibi kullanıcının girdiği yazısal bilgileri içeren metadata verileri de olabilir. Video veritabanları düşünüldüğünde ise durağan imgeler için kullanılan öznitelik vektörleri aynen geçerli olmakla beraber bunlara ek olarak hareket karakterisitği, yoğunluğu vb bilgileri içeren değerler de öznitelik vektörlerine eklenebilir. Kullanılan

bu öznitelik vektörleri varolan her sistem için farklı olabileceği gibi ortak bir standart çerçevesinde de geliştirilebilir. Bu standart oluşumu için yapılan çalışmalar 2001 yılı içerisinde kabul edilen MPEG-7 standardının oluşmasına neden olmuştur [13-18]. Bu çalışma kapsamında da baz alınan MPEG-7 standardı sadece söz edilen öznitelik vektörlerinin ne olacağı yönünde bir standartlaşma getirmektedir. Bu vektörlerin çıkarılma ve de saklanma yöntemleri ise standart dışı kalmaktadır.

MPEG-7 standardında bulunan ve de bu çalışma kapsamında kullanılan renk öznitelik vektör tanımları aşağıda verilmiştir. Daha sonraki aşamalarda bu vektörlerin nasıl çıkarıldığı da anlatılacaktır. MPEG-7 standardı kapsamında satın alınabilecek ya da geliştirilebilecek diğer öznitelik vektörleri de kullanılarak bu proje kapsamında önerilen sistem daha da geliştirilebilir.

### **Renk uzayı**

Bu vektör resimlerde kullanılan renk uzayını (RGB, YUV, HSV, HMMD) gösterir. Buna ek olarak tanımlanabilecek bir linear değişim ile daha farklı uzaylar da yaratılabilir.

### **Renk histogramı**

Görsel verinin renk karakteristiğini göstermek için kullanılmaktadır. Resim içerisinde bulunan renklerin yüzdesini verir.

### **Ana renkler**

Görsel veride ağırlıklı olarak bulunan renkleri gösterir ve bu renklerin resimde kapladığı alanı gösterir. Özellikle birkaç renk ile tanımlanabilecek resimlere erişim için çok etkin bir veridir.

### **Öznitelik vektörlerinin saklanma yöntemi**

MPEG-7 standardıyla uyumlu olabilmek amacıyla bu proje kapsamında öznitelik vektörleri XML (EXTENSIBLE MARKUP LANGUAGE) yapıda saklanmaktadır. XML bilgieri hierarşik yapıda ve içeriğe göre saklama özelliğinden dolayı HTML'e göre daha üstün olan bir dildir [20-27]. Örnek bir XML dosyası Şekil 3'de verilmiştir. Bu örnekte resimde bulunan 3 ana renk değeri %35 oranında R=100, G=100 ve B=100, %25 oranında R=0, G=10 ve B=0, %15 oranında ise R=225, G=100 ve B=185'dir.

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<DominantColors>
  <Color Ratio="35">
    <R> 100 </R>
    <G> 100 </G>
    <B> 100 </B>
  </Color>
  <Color Ratio="25">
    <R> 0 </R>
    <G> 10 </G>
    <B> 0 </B>
  </Color>
  <Color Ratio="15">
    <R> 225 </R>
    <G> 100 </G>
    <B> 185 </B>
  </Color>
</DominantColors>
```

### Şekil 3: Örnek XML dosyası.

XML’de kullanılan diğer bir önemli yapı da DTD (Document Type Declaration) ‘dir. DTD XML’de yazılmış ve eleman tipleri için hangi isimlerin nasıl kullanılacağını gösteren aşağıda belirtildiği gibi bir tanım dosyasıdır.

```
<!ELEMENT BaskınRenkler (Renk)+>
<!ELEMENT Renk (R,G,B)>
<!ELEMENT R (#PCDATA)>
<!ELEMENT G (#PCDATA)>
<!ELEMENT B (#PCDATA)>
<!ATTLIST Renk Oranı PCDATA #REQUIRED>
```

Bu örnekte DTD “Baskın renkler”i birden fazla “Renk”e sahip bir öge olarak tanımlamış ve bu renkler de R, G ve B (red,green,blue) öğelerinden oluşmuştur.

### Veritabanı yapıları

Bu çalışmada üzerinde durulan diğer bir önemli nokta XML yapısında oluşturulan öznitelik bilgisinin en etkin nasıl saklanacağına araştırılması ve gerçekleşmesidir. Proje kapsamında yapılan araştırmalarda bulunan yöntemleri üç gruba ayırmak mümkündür [1,8]:

- XML dökümanlarını basit metin olarak saklamak: Gerçeklenmesi en kolay sistemdir fakat metin sayısı arttıkça bilgiye erişmek güçleşir. Aynı zamanda metinlerin güvenilirliğini sağlama zorluğu da vardır..
- XML dökümanlarını yerel XML veritabanlarında saklamak: Bu sistemler XML yapıda verinin onaylı ya da onaysız biçimde saklayabilir. Bu sistemlerin en popülerleri Software AG Tamino XML Server ve Data Engine Excelon B2B server'dır. Yeni gelişen sistemler olduğu için özellikle MPEG 7'de kullanılması gereken bazı özellikleri desteklememektedirler.
- XML dökümanlarını ilişkisel veritabanlarında (RDBMS) saklamak: RDBMS uzun zamandır kullanılan, denemiş ve optimize edilmiş sistemlerdir. Bilgi tanımı ve de işlenmesi için SQL (Structured Query Language) destekleri vardır. Bu nedenlerden dolayı bu çalışmada saklama ortamı olarak RDBMS kullanılmıştır.

## RENGE DAYALI ÖZNİTELİK ÇIKARMA

Renk öznitelik vektörünün çıkarılması için ilk aşama resimlerin 3-boyutlu histogramlarının hesaplanmasını gerektirmektedir. Bu amaçla aşağıdaki yapı kullanılmıştır.

```
int[][][] c_map = new int[256/q_factor][ 256/q_factor][ 256/q_factor] ;

// q_factor: quantization factor: histogram uzunlugunu belirliyor

for (int j=0; j<h; j++) // görüntü yüksekliği
{
    for (int i=0; i<w; i++) // görüntü genişliği
    {
        int pixel = pixels[i+w*j] ;

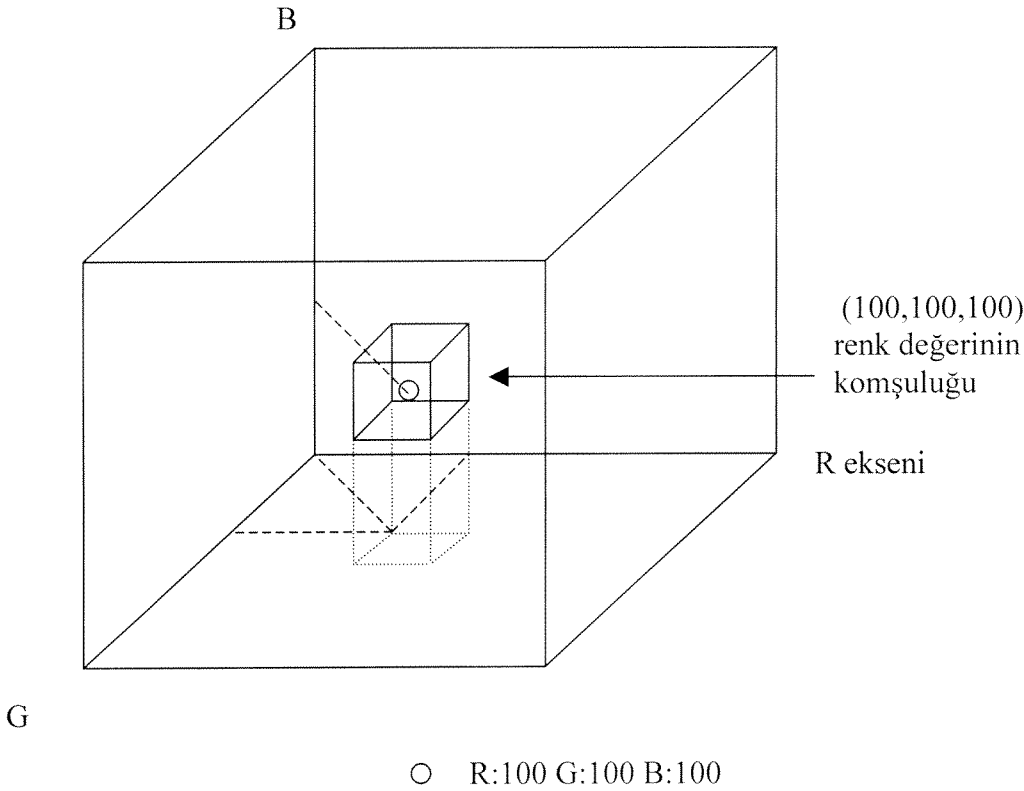
        int red    = (pixel >> 16) & 0xff;           // görüntülerin r,g,b, olarak
        int green  = (pixel >>  8) & 0xff;           // okunması
        int blue   = (pixel      ) & 0xff;

        c_map[red/q_factor][green/q_factor][blue/q_factor]++;

        //r,g,b histogramlarının olusturulması
        r_hist[red]++;
        g_hist[green]++;
        b_hist[blue]++;
    }
}
```

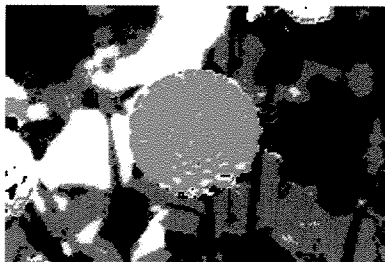
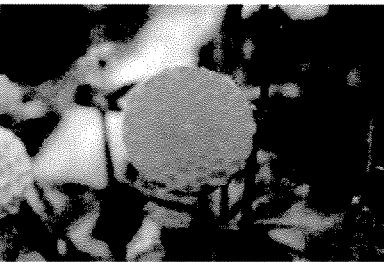
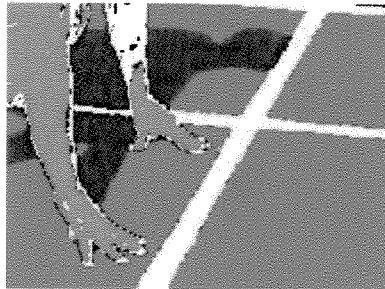
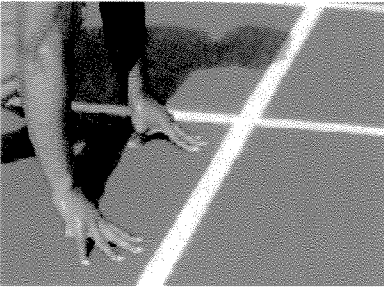
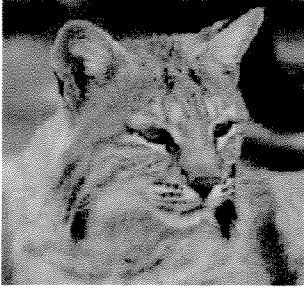
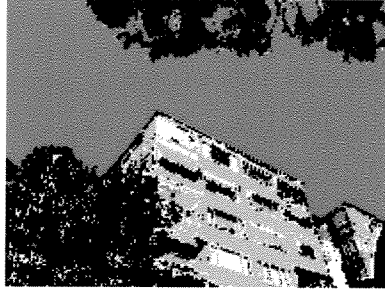
Algoritmadaki 2. basamak ise elde edilen 3-Boyutlu histogramda baskın olan renklerin saptanmasıdır. Bu çalışmada en baskın 3 ana rengin bulunması hedeflenmektedir. Yapılan çalışmalar resimleri modellemek için 3 ana rengin yeterli olacağını göstermiştir. Bu aşamada histogram sırasındaki en büyük değerlere sahip 3 rengi seçmek uygulanabilecek en basit yoldur. Örneğin, (100,100,100) R.G.B değerlerine sahip C1 rengi görüntünün %25'ini (101,101,101) renklerine sahip C2 rengi ise % 20'sini oluşturuyor olsun. Sözü edilen yöntemle birbirlerine çok yakın değerlere sahip olmalarına karşın C1 ve C2 renkleri 2 baskın renk gibi gözükecektir.

Bunu önlemek için 3-boyutlu histogram üzerinde bir komşuluk kavramı geliştirilmiştir (Şekil 4). Eğer komşuluk içerisinde büyük bir değer bulunursa kullanılan kübün merkezi yeni değere getirilir. Bu algoritma sonucunda 3'den fazla baskın renk bulunmakta fakat en büyük üçü öznitelik vektöründe kullanılmak üzere seçilmektedir.



**Şekil 4:** Baskın renk tespitinde kullanılan komşuluk alanı

Şekil 5’de önerilen metod kullanılarak bulunan baskın renkler gözükmemektedir. Beyaz baskın renge sahip olmayan bölgeleri göstermektedir.



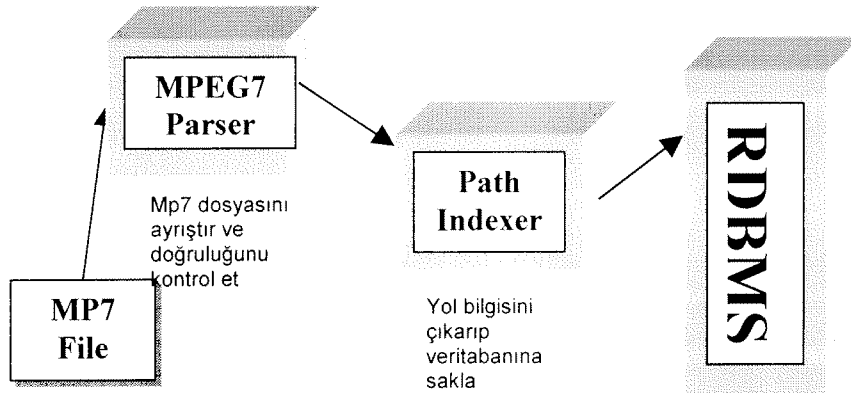
Şekil 5. Baskın renk ayrıştırma.

Sonuç olarak her resim için histogram ve de baskın renklerden oluşan öznitelik vektörü çıkarılmıştır.

## GELİŞTİRİLEN VERİTABANI YAPISI

Bu proje kapsamında saklama ortamı olarak RDBMS kullanılmıştır. XML datalarını en uygun şekilde saklamak ve erişimini kolaylaştırmak için ise başarılı bir eşleme gerçekleştirilmiştir. Geliştirilen veritabanı yapısı çoklu şemaları içerisinde saklayabilmektedir. Bu çalışmada XML bilgisini RDBMS’de saklamak için kullandığımız temel fikir, yol bilgisine bağlı bir saklama kriteri geliştirmektir. Schema’dan elde edilen yol bilgisi, “PATH” adı verilen bir tabloda (Tablo 2) saklanır. Bu tablo schema’daki olabilecek tüm yol bilgisini içerir. Tablodaki yolların kendilerine özgü bir “ID” bilgileri vardır. Aynı zamanda bu yolun ait olduğu schem’yi gösteren SCHEMA\_ID bilgileri de saklanır. SCHEMA\_ID bilgisi MPEG-7 haricindeki schema’ların da desteklenmesi istenildiği takdirde kullanılacak ek bir özelliktir.

Bir XML dosyası (göresl verinin öznitelik vektörlerini taşıyan dosya) sunucuya yüklendiği zaman Şekil 6’de gösterildiği gibi ilk olarak ayrıştırılır ve doğruluğu kontrol edilir (uygun bir MPEG-7 dosyası olup olmadığı). Daha sonra her düğümdeki bilgi çıkarılarak veri tabanında saklanır.



Şekil 6: Mpeg 7 dosyasının veritabanında saklanması

Örnek bir yol tablosu ve bir dökümanın bu tabloya göre incelenmesi aşağıda verilmiştir. Burada kullanılan döküman metadata bilgilerini içeren ve de kullanıcı tarafından girilen bir dökümandır. İçerisindeki veriler TextAnnotation/StructuredAnnotation/WhatAction, TextAnnotation/StructuredAnnotation/Where etc. MPEG7 standardı ile uyumlu olarak gösterilmiştir.

Buradaki dökümanda where = Spain, whataction = soccer game, who = suecia ve whatobject = succer field olarak verilmiştir.

<<RelationalTable>>	
PATH	
◆ID : NUMBER	
◆VALUE : VARCHAR2	
◆SCHEMA_ID : NUMBER	

ID	VALUE	SCHEMA_ID
0	/TextAnnotation	0
1	/TextAnnotation/StructuredAnnotation	0
2	/TextAnnotation/StructuredAnnotation/Who	0
3	/TextAnnotation/StructuredAnnotation/WhatObject	0
4	/TextAnnotation/StructuredAnnotation/WhatAction	0
5	/TextAnnotation/StructuredAnnotation/Where	0

<<RelationalTable>>	
NODE	
◆ID : NUMBER	
◆DOC_ID : NUMBER	
◆PATH_ID : NUMBER	
◆VALUE : VARCHAR2	

ID	DOC_ID	PATH_ID	VALUE
0	0	2	Suecia
1	0	3	Soccer field
2	0	4	Soccer game
3	0	5	Spain

**Tablo 2:** PATH ve NODE tabloları.

Geliştirilen veritabanı yapısında PATH ve NODE tablolarına ek olarak aşağıdaki tablolar da bulunmaktadır. Her tablonun ne amaçla kullanıldığı yanında verilmiştir. Böyle bir yapı kullanmadaki amaç daha önce belirtildiği gibi karmaşık aramalara olanak tanınmasıdır. Örneğin bir video dizisi içerisinde bulunan bir karakterin diğer bir resimde ya da yazıda tekrar geçmesi halinde bu verilerin otomatik olarak kullanıcıya sunulmasına olanak tanınmaktadır.

1. Attribute
2. AVObject\_Constants
3. AVObject
4. Cache
5. Document
6. DS
7. Node
8. Path
9. PathAttribute
10. Path\_Dummy
11. Path\_Patterns
12. PathToPath
13. PathTree
14. Prm\_UserTypes
15. QueryCart
16. Users

#### 1. Attribute

ATTRIBUTE		Bu tablo MPEG7 dokümanlarındaki öznitelikleri saklar. Öznitelikler ID anahtarı tarafından tanımlanır. PATH_ID, DOC_ID, NODE_ID anahtarları özniteliklerin tanımlandığı PATH, DOCUMENT ve NODE tablolarını referans gösterir. VALUE alanı özniteliğin değerini saklar.
PK	ID	
	PATH_ID DOC_ID NODE_ID	
	VALUE	

## 2. AVOBJECT\_Constants

AVOBJECT_CONSTANTS		Değişik AVOBJECT tiplerini tanımlamakda kullanılır. Şu aşamada 3 tip AVOBJECT vardır: Ses, video, imge. Bu tiplerin çalışmanın ileriki aşamalarında artırılması planlanmaktadır. Bu tablo AVOBJECT'leri saklayan AVOBJECT tablosunu referans gösterir.
	ID DESCRIPTION	

## 3. AVOBJECT

AVOBJECT		Audio-Visual (AV) öğeleri gösterir. Her AV öğe tek bir numara, bir değer, tanım dosyasına referans, tip belirtgeci, dosya adı ve de MIME tipinden oluşur.
PK	ID	
	VALUE DOC_ID TYPE NAME CONTENTTYPE	

## 4. Cache

CACHE		Önbellek tablosu sistem performansını arttırmak için tasarlanmıştır. Önceki aramaları saklar
PK	ID	
	QUERY STRING AVOBJECTRESULTS DOCUMENT RESULTS CALLS LASTCALLED LASTEXECUTIONDATE	

## 5. Document

DOCUMENT		Document tablosu MPEG-7 dosyalarını veri tabanında saklamak için kullanılır. ID ana anahtar, AV_ID AVOject tablosunu gösteren anahtar, VALUE, dosyanın saklandığı alan, NAME dosya adı, ISPARSED dosyanın ayrıştırılıp ayrıştırılmadığını gösteren alan ve CONTENTTYPE MIME tipini gösteren (çoğunlukla “text/xml”) alandır.
U1	ID	
	AV_ID VALUE NAME ISPARSED CONTENTTYPE	

#### 6. DS

DS		DS, Descriptor Schemes içingerekli bilgileri saklar.
	ID NAME	

#### 7. Node

NODE		Node tablosu satır sayısı bakımından veritabanındaki en uzun tablodur. Ayrıştırılan MPEG7 dosyalarındaki tüm bilgi burada saklanır. MPEG7 dosyası ayrıştırıldığında yol bilgisi ve de eşleşen bilgi çıkarılarak buraya konur.
U1	ID	
	DOC_ID PATH_ID VALUE	

#### 8. Path

PATH		Path tablosu MPEG-7 schema tanımlarını tutar. Schema’da bulunan tüm olası yollar bu tabloda bulunmaktadır.
PK	ID	
	VALUE SCHEMA_ID DS_ID	

#### 9. PathAttribute

PATHATTRIBUTE		PathAttribute tablosu Path tablosuna benzer şekilde yolları tutar. Fakat Path tablosu düğümler için tasarlanmasına karşın bu tablo öznelikler için tasarlanmıştır.
PK	ID	
U1	PATH_ID VALUE SCHEMA_ID	

#### 10. Path\_Dummy

PATH_DUMMY		Path_Dummy yollar için geçici bir saklama alanıdır.
PK	ID	
U1	VALUE SCHEMA_ID DS_ID	

#### 11. Path\_Patterns

PATH_PATTERNS		Path_Patterns tablosu yolların tekrar eden kısımları için kullanılır.
PK	ID	
U1	VALUE SCHEMA_ID	

#### 12. PathToPath

PATHTOPATH		PathToPath tablosu yollar arasındaki ilişkileri tutar..
PK	ID	
	RELATED_ID	

#### 13. PathTree

PATHTREE	PathTree ana ve alt yollar arasındaki bağıntıları tutar.
ID PATH_ID CHILDREN	

14. Prm\_UserTypes

PRM_USERTYPES	PRM_Usertypes, kullanıcı tiplerini tanımlayan parametre dosyası.
DEFINITION VALUE	

15. QueryCart

QUERCART	QueryCart tablosu kullanıcı aramalarını saklar
U1 USER_ID	
CART CART_NAME	

16. Users

USERS	Kullanıcıların sisteme ulaşım derecelerini tutan tablo.
PK ID	
USERNAME NAME SURNAME PASSWORD TYPE ISVALID	

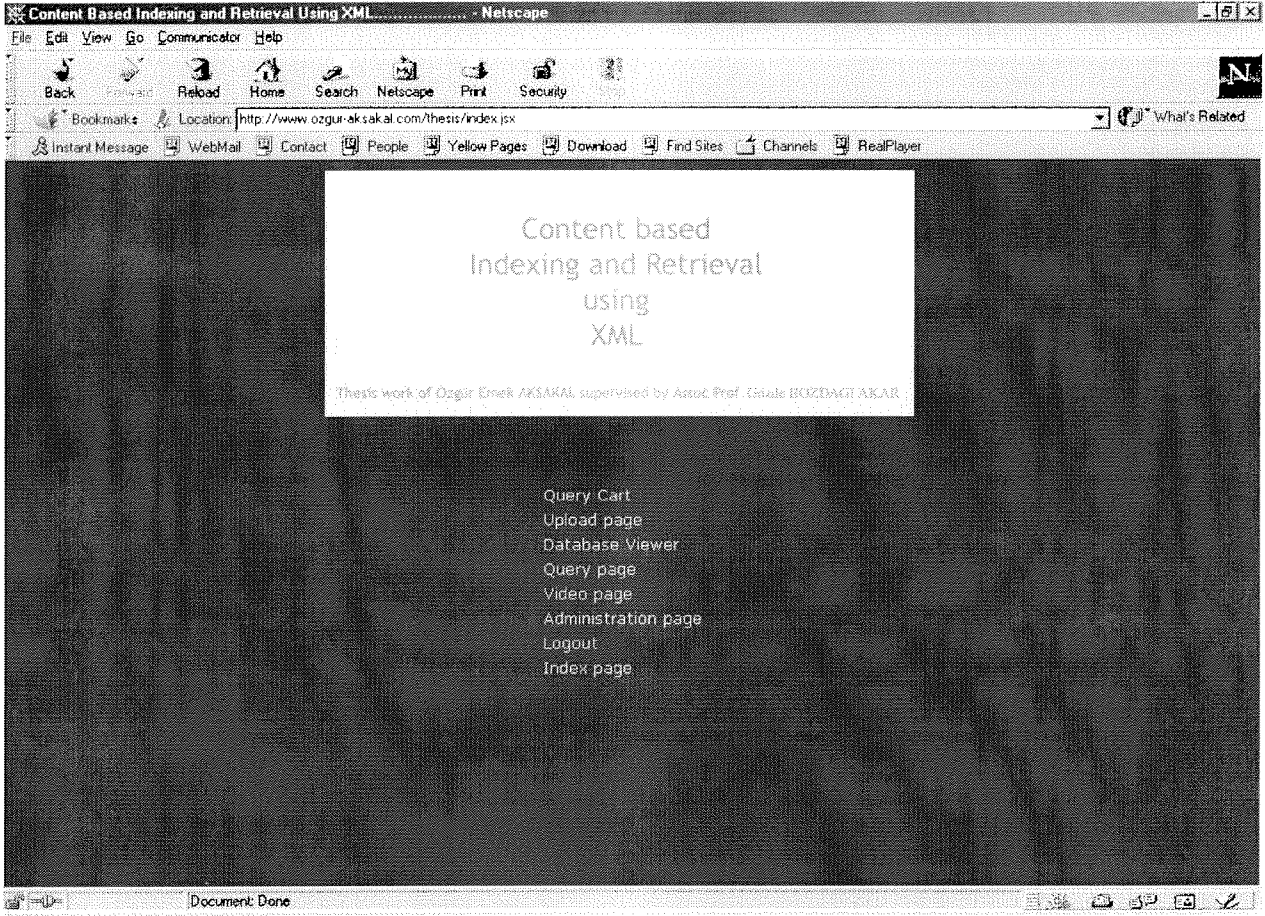
## SONUÇLAR VE DEĞERLENDİRME

Bu proje kapsamında çoğulortam verilerini modellemek için kullanılan MPEG-7 standardıyla uyumlu olarak çalışan bir veritabanı sistemi geliştirilmiştir. Veritabanında saklanacak olan veriler (renk verileri) otomatik olarak çıkarılarak yine MPEG-7 ile uyumlu olarak XML dosyalarında saklanmaktadır. Giriş bölümünde de belirtildiği gibi çoğulortam verilerindeki yapısal, zamansal ve mantıksal işikilerinden dolayı bu veriler veritabanında erişimi kolaylaştırmak için etkin bir biçimde saklanmalıdır. Bu proje kapsamında geliştirilen sistemde XML dosyaları ilişkisel bir veritabanında referans tabloları aracılığıyla saklanmaktadır. Bu tablolar veriler arası ilişkiyi XML standart kapsamında verilen ana başlıklar aracılığıyla saklamaktadır. Aynı zamanda derinlikler (sahip olduğu özellik sayısı) farklı olan nesneler arasında da karşılaştırmayı kolaylaştıran bir yapıdır. Proje kapsamında geliştirilen bu sistem kullanıcı kolaylığı, güvenlik ve her ortama uyumluluk gibi özellikler gözönünde bulundurularak (diğer sistemlerle karşılaştırmalar detaylı olarak giriş bölümünde verilmiştir) Java Servlet'ler kullanılarak gerçekleştirilmiştir.

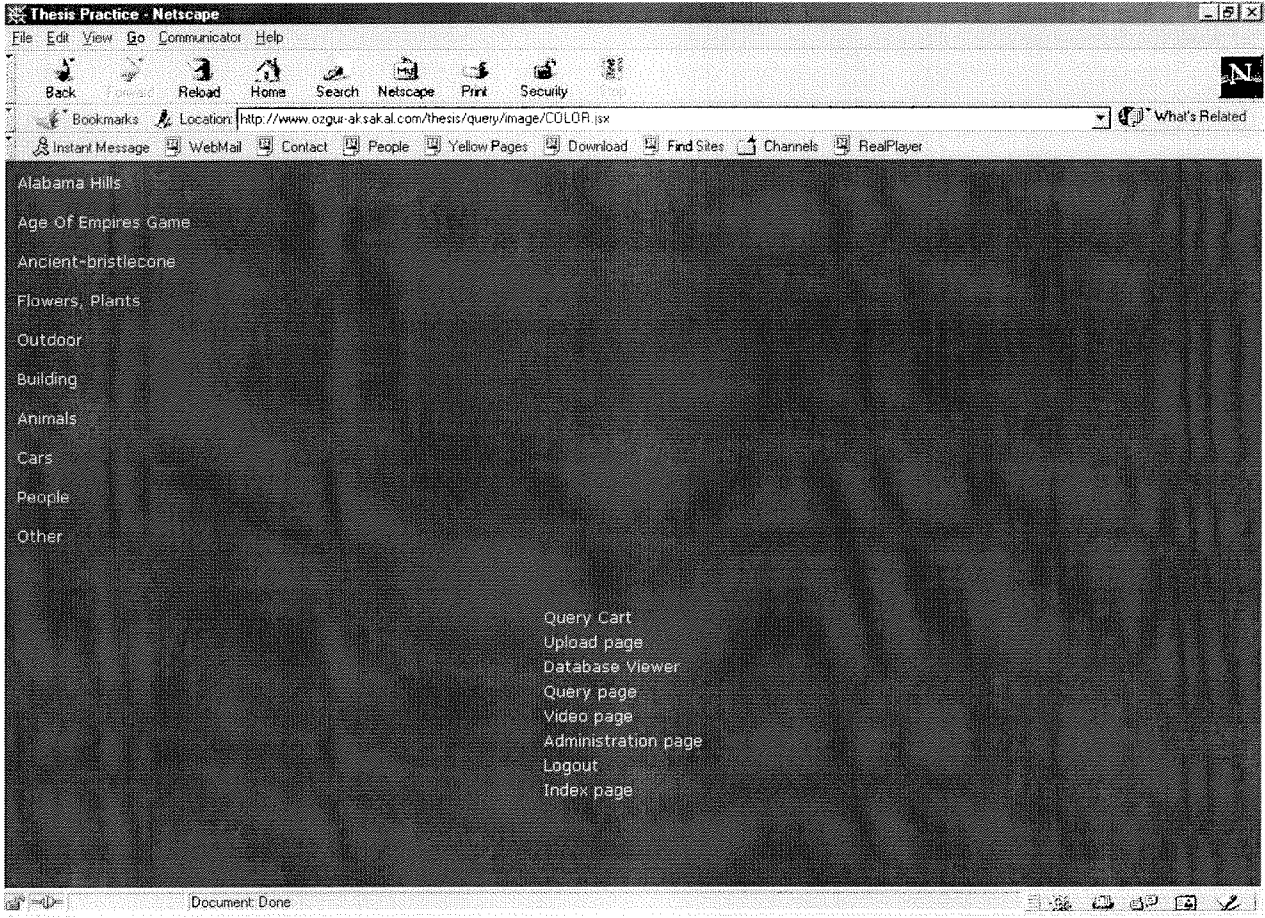
Sistemde kullanılan renk histogramı ve de baskın renk özellikleri, bu özelliklerin elde edilmesindeki ve karşılaştırmada kullanılan metodlardan dolayı literatürde varolan örnek sistemlere göre (örnek arama sonuçlarında belirtildiği gibi) daha iyi performans göstermektedir. Aynı zamanda kullanılan Servlet yapısı her kullanıcıda bir APPLLET'in çalışmasını engelleyerek hızı arttırmaktadır. Kullanılan veritabanı yapısı ise basit bir ilişkili veritabanını çoğulortam verileri için akıllı bir şekilde kullanılmasıyla gerçekleştirilmiştir. Daha ileriki çalışmalarda başka veri yapıları da incelenerek önerilen yapıya erişimdeki hız ve verim artışının daha detaylı olarak gösterilmesi planlanmaktadır.

İleriki çalışmalarda renk öznitelik vektörüne ek olarak MPEG-7 standardında belirtilen şekil, doku, metadata gibi özellikler de sisteme entegre edilecektir.

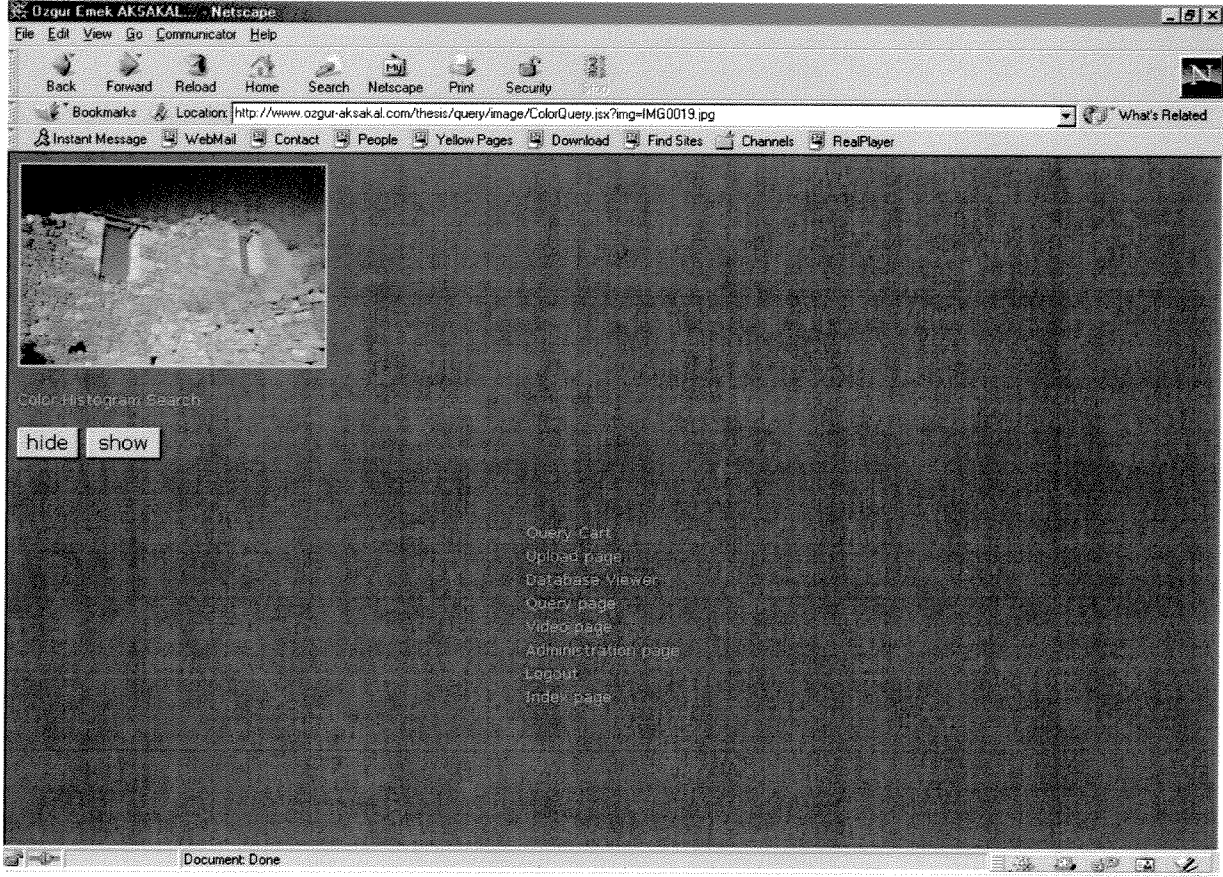
Java Servlet/JSP kullanarak ve yukarıda anlatılan veri yapısı altında geliştirilen sistemde yapılan örnek arama sonuçları aşağıda verilmiştir. WEB sunucuda renk bazlı arama için kullanılan servlet tabanlı örnek program ise Ek 1'de verilmiştir.



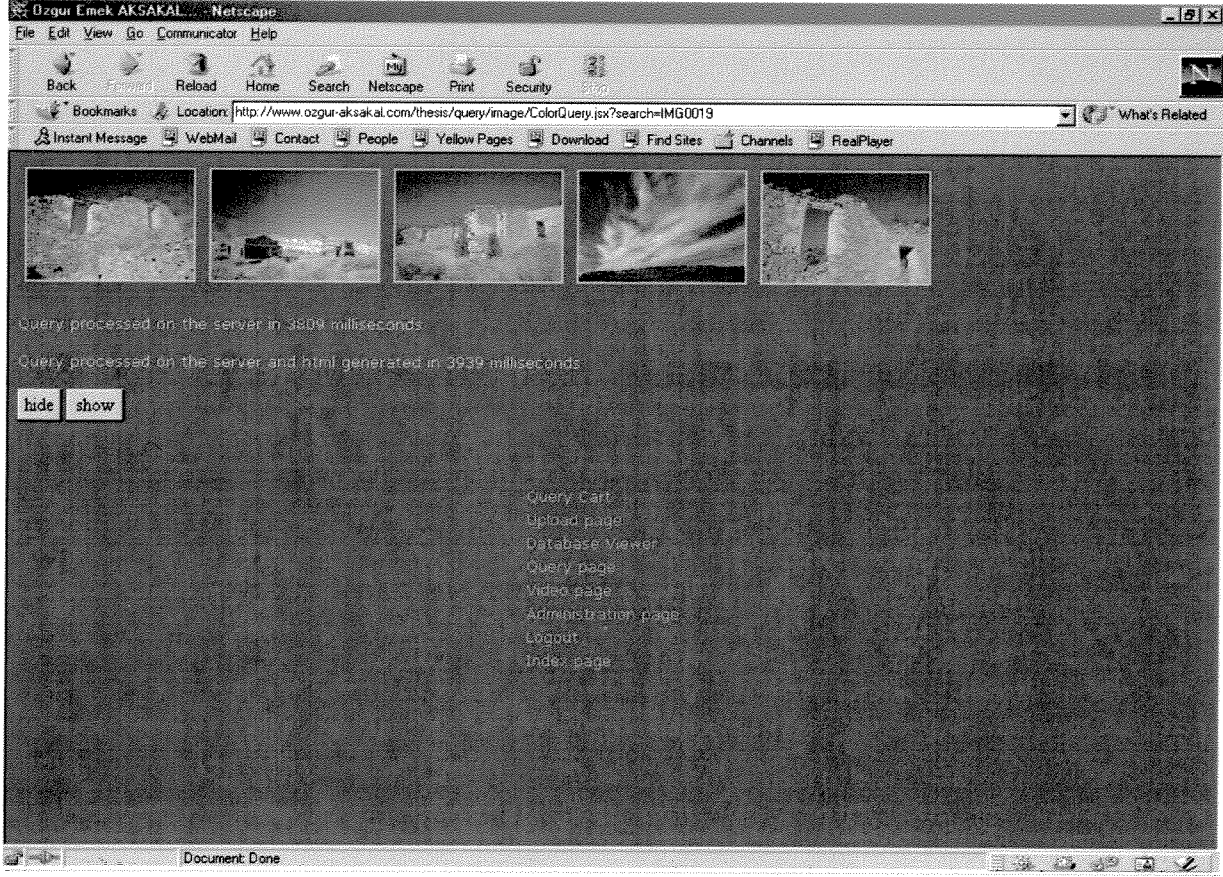
Şekil 7: Giriş sayfası



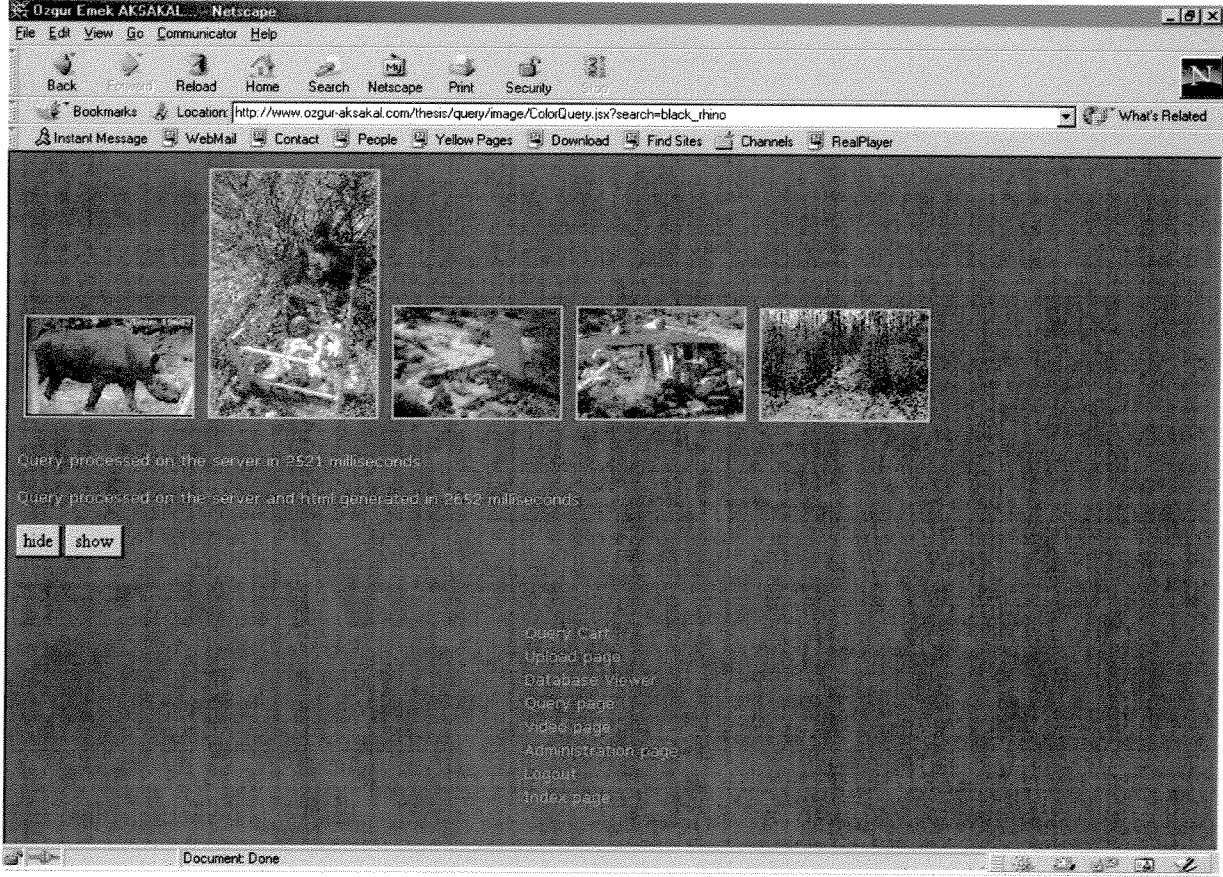
Şekil 8: Veritabanı tiplerinin gösterildiği sayfa.



Şekil 9: Üzerinde arama yapılacak olan imge.



Şekil 10: Arama sonuçları.



Şekil 11: Yeni bir imge üzerinde arama sonuçları.

Bu arama sonuçlarına ulaşmak için kullanılan örnek bir XML dosyası ve bu dosyanın veritabanında saklanması için gerekli PATH ve NODE tabloları ise aşağıdaki gibidir.

(Bu XML dosyasında "D:\Video\Kucuk\deliyurek.mpg.0.jpg" adlı imge incelenmiştir. İmgede bulunan en baskın renk %16 oranında piksele sahip R=5, G=15, B=16'dır)

```
<Mpeg7><DescriptionUnit type = "DescriptorCollectionType">
<Descriptor size = "1" type = "DominantColorType" input =
"D:\Video\Kucuk\deliyurek.mpg.0.jpg"><ColorSpace type = "YCRCB" colorReferenceFlag =
>false"/>
<SpatialCoherency>-1</SpatialCoherency>
<Values><Percentage>16</Percentage>
<ColorValueIndex>5 15 16 </ColorValueIndex>
</Values>
</Descriptor>
</DescriptionUnit>
</Mpeg7>
```

ID	VALUE	SCHEMA_ID
0	/DescriptorUnit	0
1	/DescriptorUnit/Descriptor	0
2	/DescriptorUnit/Descriptor /ColorSpacetype	0
3	/DescriptorUnit/Descriptor /SpatialCoherency	0
4	/DescriptorUnit/Descriptor /Values	0
5	/DescriptorUnit/Descriptor /Values/Percentage	0
6	/DescriptorUnit/Descriptor /Values/ColorValueIndex	0

ID	DOC_ID	PATH_ID	VALUE
0	0	1	DominantColorType
1	0	2	YCRCB
2	0	3	-1
3	0	5	16
4	0	6	5 15 16

**Tablo 3:** Örnek PATH ve NODE tablosu

### Performans

Geliştirilen sistemin performansı saklama ve arama fonksiyonları göz önünde bulundurularak değerlendirilebilir. Bu iki faktörden saklama daha az kullanıldığı için performans değerlendirmelerinde arama fonksiyonu dikkate alınacaktır.

Proje çerçevesinde kullanılan test sitemi aşağıdaki gibidir:

### Donanım :

Compaq Armada M700

650 Mhz P3 CPU

128 Mb RAM

10 Gb Hard Disk

**Yazılım:**

Oracle 8i (8.1.7) Personal Database

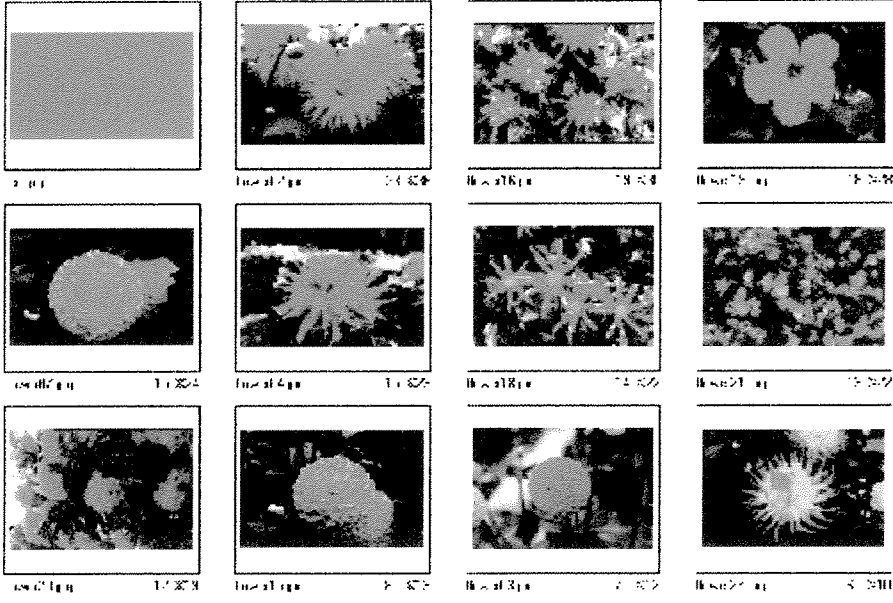
Apache Tomcat 3.2 Servlet & JSP Container

Oracle 8i JDBC 2 Driver

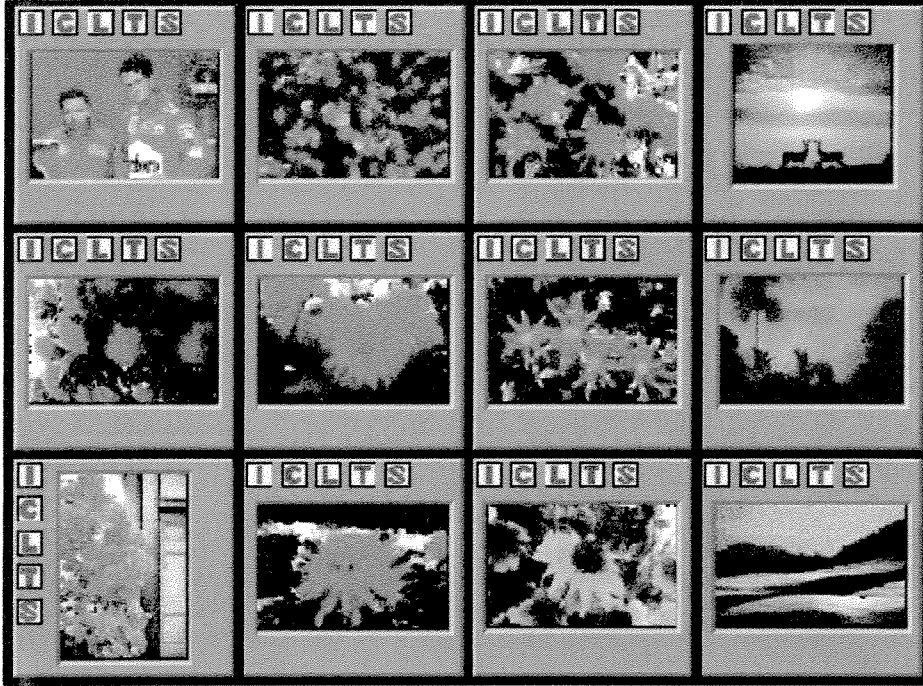
Örnek resimlerde de gösterildiği gibi histogram'a bağlı arama sonuçları 2-3 sn içerisinde tamamlanmaktadır. Bu 338 XML dosyasının Apache Xerces-J DOM Parser kullanarak ayrıştırılmasını da içermektedir. Sonuçların gösterilmesi için hazırlana HTML sayfaları ise ortalama 0.15 sn'de tamamlanmaktadır.

Var olan sistemlerle gürbüzlük açısından karşılaştırmak amacı ile de QBIC sistemi kullanılmıştır. Elde edilen sonuçlardan (Şekil 13) baskın renk performansının önerilen sistemde daha yüksek olduğu gözlemlenmiştir. Bu da hem baskın renk elde etme hem de karşılaştırma aşamasında kullanılan algoritmaların gürbüzlüğüyle doğru orantılıdır.

## Önerilen sistem



## QBIC



Şekil 12. Baskın renk özelliğine göre karşılaştırma.

## **PROJENİN GERÇEKLENMESİNDE KARŞILAŞILAN ZORLUKLAR**

Projede karşılaşılan en büyük zorluk gerekli eleman desteğinin sağlanamaması olmuştur. Projede çalışan elemanların MS çalışmalarını tamamlamadan yurtdışına gitmeleri, bilgi birikiminin diğer öğrencilere aktarımındaki zorluklar projede gecikmelere ve de yayın çıkmamasına neden olmuştur. Şu aşamada projenin daha da geliştirilmesi ile uğraşan (tüm MPEG-7 öznitelik vektörlerini destekleyecek şekilde) ve Ocak ayında tezini savunacak bir MS öğrencisi bulunmaktadır.

## REFERANSLAR

- [1] V.S. Subrahmanian, Principles of Multimedia Database Systems, Morgan Kaufmann, 1998.
- [2] A. Sloane, "Aspects of multimedia database technology for teleteaching", Proceedings 3rd IFIP Teleteaching conference, Trondheim, Norway, August 1993, pp 809-817.
- [3] D. A. Adjero, K. C. Nwosu, "Multimedia Database Management Systems - Requirements and Issues" IEEE MultiMedia, July - September 1997, Vol. 4, No. 3, pp 24-33.
- [4] I. L. Cheng, "Image Databases: A Content-Based Type System and Query By Similarity Match," Master's thesis, Department of Computing Science, University of Alberta, May 1999.
- [5] A. Fedorov, *et. al*, ASP 2.0, Wrox Pres. Ltd., 1998, UK.
- [6] T. C. Rakow, E. J. Neuhold and M. Lohr, "Multimedia Database Systems - The Notions and Issues," Datenbanksysteme in Büro, Technik und Wissenschaft, Dresden, March 1995, pp. 1-19.
- [7] U. Marder, VirtualMedia: Making Multimedia Database Systems Fit for World-wide Access, Proc. VII. Conference on Extending Database Technology, Germany, 31 March - 1 April, 2000, pp. 47-50
- [8] S. Marcus and V.S. Subrahmanian, "Foundations of multimedia database systems," Journal of the ACM, vol. 43, no. 3, May 1996, pp. 474-523.
- [9] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, and B. Dom et al. "*Query image and video content: the qbic system*," IEEE Computer, vol. 28, no. 9, 1995, pp. 23-32.
- [10] J. R. Smith and S.-F. Chang, "Visually Searching the Web for Content," IEEE Multimedia Magazine, Summer, Vol. 4, No. 3, 1997, pp.12-20.
- [11] S.-F. Chang, J. R. Smith, H. J. Meng, H. Wang, and D. Zhong, "Finding Images/Video in Large Archives," CNRI Digital Library Magazine, Feb. 1997

- [12] G. Speegle, X. Wang, and L. Gruenwald, "A Meta Structure for Representing Multimedia Databases", 16th British National Conference on Databases , Lecture Notes in Computer Science 1405 - Advances in Databases, July 1998, pp. 89-102.
- [13] MPEG Requirements Group, "MPEG-7 Requirements Document V.10", Doc. ISO/IEC TC1/SC29/WG11/N2996, October 1999.
- [14] The MPEG Home Page (<http://www.cselt.it/mpeg/>).
- [15] J. M. Martinez , "Overview of the MPEG-7 Standard ", ISO/IEC JTC1/SC29/WG11/ N3445, May/June 2000.
- [16] P.V. Beek, A. B. Benitez, J.Heuer, J.Martinez, P.Salembier, J.Smith, T.Walker, "MPEG-7 Multimedia Description Schemes WD", ISO/IEC/JTC1/SC29/WG11/N3247, March 2000.
- [17] J. Hunter, "DDL Working Draft 4.0", ISO/IEC/JTC1/SC29/WG11/N3575, July 2000.
- [18] MPEG DDL Group, "MPEG-7 Description Definition Language Document V 2" Doc. ISO/IEC JTC1/SC29/WG11/N2997, October 1999.
- [19] A. B. Benitez, S. Paek, S.F. Chang, A.Puri, Q. Huang, J.R. Smith, C.-S. Li, L.D. Bergman, C.N. Judice, "Object-Based Multimedia Content Description Schemes and Applications for MPEG-7," Signal Processing: Image Communication, Vol. 16, no. 1-2, 2000, pp. 235-269.
- [20] T. Bray, J. Paoli , C. M. Sperberg-McQueen , "Extensible Markup Language (XML) 1.0," Oct. 2000, (<http://www.w3.org/TR/REC-xml>).
- [21] T. Freter, "XML: It's the Future of HTML," (<http://www.sun.com/980602/xml/>).
- [22] N. Walsh, "What is XML?," Oct. 1998, (<http://www.xml.com/pub/98/10/guide1.html>)
- [23] S. St. Laurent, E. Cerami, Building XML Applications, McGraw-Hill Osborne Media, 1999.
- [24] E. Vlist, "Style-free XSLT", July 2000, (<http://www.xml.com/pub/2000/07/26/xslt/xsltstyle.html>)
- [25] XML Schema, (<http://www.w3.org/TR/2000/ND-xmlschema/>)

[26] A. M. Fedorchek, D. K. Rensin, ASP: Developing Distributed Java Applications with Remote Invocation, John Wiley and Sons Ltd ,1997.

[27] MSXML Parser (<http://msdn.microsoft.com/downloads/tools/xmlparser/xmlparser.asp>).

[28] Java Media Framework API (<http://java.sun.com/products/java-media/jmf/index.html>).

[29] J. Goodwill, Developing Java Servlets, Sams Publishing, June 1999.

[30] Apache Tomcat 3.2 Servlet ([jakarta.apache.org](http://jakarta.apache.org)).

## EKLER

### Ek 1

```
// Copyright (c) 2003 Kani Kerim GUNER kkguner@yahoo.com
package com.kkg.thesis.mpeg7;
/**
 * @author Kani Kerim GUNER
 */

//Gerekli kütüphaneler dahil ediliyor.
import com.kkg.thesis.util.*;
import java.util.*;
import java.sql.*;

public class DomColorQueryDB {
//Gerekli değişkenler tanımlanıyor.
    public DBConnectionPool2 pool; //Veritabanı bağlantı havuzu
    public Connection conn; //Veritabanı bağlantısı
    Statement stmt2 = null; //Sorgu yapıları
    PreparedStatement pstmtDC1;
    PreparedStatement pstmtDC2;
    PreparedStatement pstmtDCtemp;
    PreparedStatement pstmtDCtemp2;
    PreparedStatement pstmtcache;
    PreparedStatement pstmtcache2;
    int doc_id =0; //Veritabanındaki dokümanın numarası
    int doc_total =0; //Veritabanındaki toplam doküman sayısı
    int av_id =0; //Veritabanındaki imgenin numarası
    int[][][] domrgb; // orjinal imgenin dominant renkleri
    int[][][] domrgb2; // karşılaştırılan imgenin dominant renkleri
    int DC_total=0; //orjinal imgenin toplam dominant renk sayısı
    int DC_total2=0; //karşılaştırılan imgenin toplam dominant renk sayısı
    int DC_max; //orjinal imgedeki segmentler arasında en fazla dominant
    renk sayısı
    int DC_max2; //karşılaştırılan imgedeki segmentler arasında en fazla
    dominant renk sayısı
    int segment_total=0; //orjinal imgenin toplam segment sayısı
    int segment_total2=0; //karşılaştırılan imgenin toplam segment sayısı
    int[] similars; //orjinal imgeye benzerlikler
    ResultSet rstemp; //sorgular için gerekli sonuc yapısı
    int sgrabs=0; //aramanın square veya absolute mod olacağı
    int many=0; //aramanın single veya multi mod olacağı

    public DomColorQueryDB() {
// veritabanı bağlantı havuzu oluşturulup, veritabanına bağlanılıyor.
        pool = DBConnectionPool2.connectToPool();
        conn = pool.getConnection();
        try{
            stmt2 = conn.createStatement();
//imge icerisindeki toplam segment sayısını bulan sorgu
            String strSQL1 = "select max(segment_id) from node where doc_id=?";
//imge icerisindeki her segmentte kacar tane dominant renk oldugunu bulan
            sorgu
            String strSQL2 = "select segment_id,value from attribute where
            doc_id=?" +
                                " and pathattribute_id=0";
//imge icerisindeki dominant renklerin oranlarını bulan sorgu
```

```

        String strSQLtemp = "select segment_id,value_id,value from node where
doc_id=? and path_id=1";
//imge icerisindeki dominant renklerin bulan sorgu
        String strSQLtemp2 = "select segment_id,value_id,value from node where
doc_id=? and path_id=2";
//sorguların yapıları hazırlanıyor
        pstmtDC1 = conn.prepareStatement(strSQL1);
        pstmtDC2 = conn.prepareStatement(strSQL2);
        pstmtDCtemp = conn.prepareStatement(strSQLtemp);
        pstmtDCtemp2 = conn.prepareStatement(strSQLtemp2);

    }
    catch(Exception e){
        System.err.println("kkg: Error occurred during the database
connection");
        e.printStackTrace();
        try{
            stmt2.close();
        }
        catch(SQLException dummy){
        }
    }
}

//benzer imgeleri bulma islemi baslıyor
    public int[] findSimilar(String string_id,String strsqbrabs,String
strmany,int resultsize) throws Exception{
        try{
            sqbrabs = Integer.parseInt(strsqbrabs); //square veya absolute mod
            olacagina bakılıyor
            many = Integer.parseInt(strmany); //single veya multi mod olacagina
            bakılıyor
            av_id = Integer.parseInt(string_id); //imgenin numarası alınıyor
            doc_id=av_id;
            int cache_tokenno=0;
//Eger arama cache tablosunda varsa sonuçlar işlem yapılmadan tablodan
çagrılıyor.
            String strSQLcache = "select avobjectresults from cache where av_id=?
and queries_id=?";
            pstmtcache = conn.prepareStatement(strSQLcache);
            pstmtcache.setInt(1,av_id);
            pstmtcache.setInt(2,(2*sqbrabs+many));
            ResultSet rs = pstmtcache.executeQuery();
            similars =new int[resultsize];
            if(rs.next()){
                StringTokenizer cache_st = new StringTokenizer(rs.getString(1),"
");
                while (cache_st.hasMoreTokens()) {
                    similars[cache_tokenno] = Integer.parseInt(cache_st.nextToken());
                    cache_tokenno++;
                }
                return similars;
            }
        }

//original imge icerisindeki toplam segment sayısı bulunuyor
        pstmtDC1.setInt(1,doc_id);
        rs = pstmtDC1.executeQuery();
        if(rs.next()){
            segment_total = rs.getInt(1);
        }
    }
}

```

```

//her segmentteki dominant renk sayısını tutmak için bir dizi tanımlanıyor
ve içi dolduruluyor
int[] DC_segments=new int[segment_total];
pstmtDC2.setInt(1,doc_id);
rs = pstmtDC2.executeQuery();
while(rs.next()){
    DC_segments[rs.getInt(1)-1] =Integer.parseInt(rs.getString(2));
}

//orjinal imgedeki segmentler arasında en fazla dominant renk sayısı
bulunuyor
DC_max=0;
for(int i=0;i<segment_total;i++){
    if(DC_max<DC_segments[i])
        DC_max=DC_segments[i];
}
//orjinal imgenin toplam dominant renk sayısı bulunuyor
for(int i=0;i<segment_total;i++){
    DC_total=DC_total+DC_segments[i];
}
//veritabanındaki toplam doküman sayısı bulunuyor
String strSQL4 = "select max(id) from document";
rs = stmt2.executeQuery(strSQL4);
if(rs.next()){
    doc_total = rs.getInt(1);
}
// orjinal imgenin dominant renklerini tutacak dizi tanımlanıyor
int domrgb[][][] = new int[segment_total][DC_max][4] ;
//orjinal imgenin dominant renklerini diziye atmak üzere gerekli fonksiyon
çağırılıyor
domrgb = this.getDC(doc_id,segment_total,DC_segments,DC_max);
//Karsılaştırıcılar tanımlanıyor
DomColorComparatorSQRDB comparatorSQR = new
DomColorComparatorSQRDB();
DomColorComparatorABSDB comparatorABS = new
DomColorComparatorABSDB();
float[] diffVals = new float[doc_total]; // imgeler arası uzaklıkları
tutan dizi
int[] fileids = new int[doc_total]; //imgelerin numaralarını tutan
dizi

//veritabanındaki diğer imgeler için buraya kadarki işlemler tekrarlanıyor
for(int i=1;i<=doc_total;i++){
    pstmtDC1.setInt(1,i);
    rs=pstmtDC1.executeQuery();
    if(rs.next()){
        segment_total2 = rs.getInt(1);
    }
    int[] DC_segments2=new int[segment_total2];
    pstmtDC2.setInt(1,i);
    rs = pstmtDC2.executeQuery();
    while(rs.next()){
        DC_segments2[rs.getInt(1)-1] =Integer.parseInt(rs.getString(2));
    }

    DC_max2=0;
    for(int k=0;k<segment_total2;k++){
        if(DC_max2<DC_segments2[k])
            DC_max2=DC_segments2[k];
    }
    int domrgb2[][][] = new int[segment_total2][DC_max2][4];

```

```

        domrgb2 = this.getDC(i,segment_total2,DC_segments2,DC_max2);
//square veya absolute olmasına göre gerekli karşılaştırmacı çağrılıyor
//dönen sonuçlar diffVals dizisine aktarılıyor
        if(sqrabs==1){
            diffVals[i-1] =
comparatorSQR.compare(domrgb,domrgb2,segment_total,segment_total2,DC_segmen
ts,DC_segments2,many);
        }
        else if(sqrabs==2){
            diffVals[i-1] =
comparatorABS.compare(domrgb,domrgb2,segment_total,segment_total2,DC_segmen
ts,DC_segments2,many);
        }
        fileids[i-1] = i;
    }
//dönen sonuçlara göre benzerlik sıralaması yapılıyor
    similars = this.rank(fileids,diffVals,resultsize);
//sonuçlar cache tablosuna kaydediliyor
    String cache_similars =String.valueOf(similars[0]);
    for(int i=1;i<10;i++){
        cache_similars=cache_similars+" "+String.valueOf(similars[i]);
    }
    String strSQLcache2 = "insert into
cache(id,av_id,queries_id,avobjectresults)" +
        " values(?,?,?,?)";
    pstmtcache2 = conn.prepareStatement(strSQLcache2);
    pstmtcache2.setInt(1,IDGenerator.initCacheID());
    pstmtcache2.setInt(2,av_id);
    pstmtcache2.setInt(3,(2*sqrabs+many));
    pstmtcache2.setString(4,cache_similars);
    pstmtcache2.executeUpdate();
//gereksiz sorgu yapıları sonlandırılıyor
    stmt2.close();
    pstmtcache.close();
    pstmtcache2.close();
}

    catch(Exception e){
        System.err.println("kkg: Error occurred during the database
connection");
        e.printStackTrace();
        try{
            stmt2.close();
        }
        catch(SQLException dummy){
        }
    }
}
//sonuçlar web sayfasına gönderiliyor
    return similars;
}

//imgenin dominant renklerini bulan fonksiyon
public int[][][] getDC(int docid, int seg_total,int[] DCseg,int DCmax){
    int domrgbttemp[][][] = new int[seg_total][DCmax][4];
    String strdomrgbttempall[][] = new String[seg_total][DCmax];
    try{
//imgenin dominant renklerinin oranları veritabanından alınıyor
        pstmtDCtemp.setInt(1,docid);
        rstemp=pstmtDCtemp.executeQuery();
        String ratio="";
        while(rstemp.next()){
            ratio=rstemp.getString(3);

```

```

        domrgbtemp[rstemp.getInt(1)-
1][rstemp.getInt(2)][3]=Integer.parseInt(ratio);
    }
//imgenin dominant renkleri veritabanından aliniyor
    pstmtDCtemp2.setInt(1,docid);
    rstemp=pstmtDCtemp2.executeQuery();
    int tokenno;
    while(rstemp.next()){
        StringTokenizer st = new StringTokenizer(rstemp.getString(3)," ");
        tokenno=0;
        while (st.hasMoreTokens()) {
            domrgbtemp[rstemp.getInt(1)-1][rstemp.getInt(2)][tokenno] =
Integer.parseInt(st.nextToken());
            tokenno++;
        }
    }
}
catch(Exception e){
    System.err.println("kkg: Error occurred during the database
connection");
    e.printStackTrace();
    try{
        stmt2.close();
    }
    catch(SQLException dummy){
    }
}
return domrgbtemp;
}
//karsilastirma sonuclarini siraya koyan fonksiyon
public int[] rank(int[] fileids, float[] diffValues,int result_size){
    int size = diffValues.length;
    int[] results = new int[result_size];
    float[] diffValuesSorted = new float[size];
    for(int i=0;i<size;i++){
        diffValuesSorted[i] = diffValues[i];
    }
    Arrays.sort(diffValuesSorted);
    for(int i=0;i<result_size;i++){
        loop: for(int j=0;j<size;j++){
            if(diffValuesSorted[i] == diffValues[j]){
                results[i] = fileids[j];
                diffValues[j] = -999;
                break loop;
            }
        }
    }
    return results;
}
//imgelerin isimlerini veritabanından okuyan fonksiyon
public String getImage(int param){
    String getimageresult=null;
    try{
        PreparedStatement pstmtgetImage;
        String strSQL6 = "select name from avobject where id="+param;
        pstmtgetImage = conn.prepareStatement(strSQL6);
        ResultSet rsgetImage=pstmtgetImage.executeQuery();
        if(rsgetImage.next()){
            getimageresult=rsgetImage.getString(1);
        }
        pstmtgetImage.close();
    }
}

```

```

    }
    catch(Exception e){
        System.err.println("kkg: Error occurred during the database
connection");
        e.printStackTrace();
        try{
            stmt2.close();
        }
        catch(SQLException dummy){
        }
    }
    return getimageresult;
}
//web sayfasında, imgelerin altında dominant renklerini küçük kutularla
gösteren
//fonksiyon
public String[] getdomRGB(int param) throws Exception{
    pstmtDC1.setInt(1,param);
    ResultSet rsRGB = pstmtDC1.executeQuery();
    int segment_totaltemp=0;
    if(rsRGB.next()){
        segment_totaltemp = rsRGB.getInt(1);
    }
    int[] DCsegtemp=new int[segment_totaltemp];
    pstmtDC2.setInt(1,param);
    rsRGB = pstmtDC2.executeQuery();
    while(rsRGB.next()){
        DCsegtemp[rsRGB.getInt(1)-1]
=Integer.parseInt(rsRGB.getString(2));
    }

    int DC_maxtemp=0;
    for(int i=0;i<segment_totaltemp;i++){
        if(DC_maxtemp<DCsegtemp[i])
            DC_maxtemp=DCsegtemp[i];
    }

    int[][][] domrgb =
this.getDC(param,segment_totaltemp,DCsegtemp,DC_maxtemp);
    String[] domRGB = new String[3];
    String [] domtemp = new String[3];
    for (int i=0;i<3;i++){
        for (int j=0;j<3;j++){
            domtemp[j]=Integer.toHexString(domrgb[0][i][j]);
            if (domrgb[0][i][j]<16) {
                domtemp[j]="0".concat(domtemp[j]);
            }
            if (j==0) {
                domRGB[i]=domtemp[j];
            }
            if (j==1||j==2) {
                domRGB[i]=domRGB[i].concat(domtemp[j]);
            }
        }
    }
    return domRGB;
}
}

```

```

// Copyright (c) 2003 Kani Kerim GUNER kkguner@yahoo.com
package com.kkg.thesis.mpeg7;
/**
 * @author Kani Kerim GUNER
 */
public class DomColorComparatorABSDB extends Object {

    public DomColorComparatorABSDB() {

    }

//absolute modda karşılaştırma fonksiyonu
//*****
    public float compare(int[][][] RGBDom1, int[][][] RGBDom2, int st1, int
st2, int[] DCs1, int[] DCs2, int type){
        float Distance; //iki dominant renk arasındaki mesafe
        float Similar; //iki dominant renk arasındaki benzerlik
        float a=0,b=0,c=0;
        float[][] Similarity= new float[st1][st2]; //segmentlerin benzerliği
        float[][] distanceresults= new float[st1][st2]; // segmentler arası
        // mesafe

        float threshold=20; //esik degeri
//orjinal imgedeki k tane segment ve karşılaştırılan imgedeki m tane
//segment için segmentler arası mesafeler bulunuyor
        for(int k=0;k<st1;k++){
            for(int m=0;m<st2;m++){
                Similar=0;
                Distance=0;
                Similarity[k][m]=0; //orjinaldeki k. segmentin, bakılandaki m.
// segmente benzerliği orjinal imgedeki segmentin i tane rengi ve bakılan
// imgedeki segmentin j tane rengi için renkler arası mesafe bulunuyor
                for(int i=0;i<DCs1[k];i++){
                    for(int j=0;j<DCs2[m];j++){
//renklerin R,G,B bileşenleri arasındaki mesafe a,b,c olarak bulunuyor
                        a= RGBDom1[k][i][0] - RGBDom2[m][j][0];
                        b= RGBDom1[k][i][1] - RGBDom2[m][j][1];
                        c= RGBDom1[k][i][2] - RGBDom2[m][j][2];
//eğer a,b,c nin üçü de esik değgerinden küçükse karelerinin toplamının
karekökü bulunuyor
                        if
((Math.abs(a)<threshold)&&(Math.abs(b)<threshold)&&(Math.abs(c)<threshold))
{
                            Distance=a*a+b*b+c*c;
                            Distance = (float)Math.sqrt(Distance);
//mesafe normalize ediliyor, yani en büyük değer 1 olacak şekilde
ayarlanıyor
                            Distance = Distance/(((float)Math.sqrt(3))*threshold);
                        }
//eger a,b,c yukarıdaki şartı sağlamıyorsa mesafe en büyük mesafe olarak
//atanıyor
                        else {
                            Distance=1;
                        }
}
//iki renk arası benzerliği bulmak için renklerin oranlarından küçük olan,
//renkler arası mesafenin 1 den çıkarılmış haliyle çarpılıyor

```

```

Similar=((float)Math.min(RGBDom1[k][i][3],RGBDom2[m][j][3])/100)*(1-
Distance);
//eger single modda islem yapılıyorsa bulunan deger daha önce (diger
//renkler arası benzerlikler) bulunan degerlerle kıyaslanıyor, büyük olan
// segmentler arası benzerlik degeri oluyor
    if(type==1){
        if(Similarity[k][m]<Similar){
            Similarity[k][m] = Similar;
        }
    }
//eger multi modda islem yapılıyorsa bulunan deger daha önce (diger
renkler arası benzerlikler) bulunan degerlerle toplanıyor, toplam deger
segmentler arası benzerlik degeri oluyor
    else if(type==2){
        Similarity[k][m] = Similarity[k][m] + Similar;
    }
}
//single mod ise segmentler arası mesafe, segmentler arası benzerligin 1
// den çıkarılmasıyla bulunur
    if(type==1){
        distanceresults[k][m]=1-Similarity[k][m];
    }
//single mod ise segmentler arası mesafeyi bulmak için önce segmentler
//arası benzerlikler normalize edilir sonra 1 den çıkarılır
    if(type==2){
        Similarity[k][m]=Similarity[k][m] / (DCs1[k]*DCs2[m]);
        distanceresults[k][m]=1-Similarity[k][m];
    }
}
}
float Distanceresult=999999;
//iki imge arası mesafe segmentler arası mesafelerin en küçüğüdür.
//Bu sayade bir araba resmi ile duvarında araba posterı olan oda resmi
// benzer çıkar
    for(int p=0;p<st1;p++){
        for(int r=0;r<st2;r++){
            if(Distanceresult>distanceresults[p][r])
                Distanceresult=distanceresults[p][r];
        }
    }

    return Distanceresult;
}
}

```

MPEG-7 COMPLIANT ORDBMS BASED  
IMAGE STORAGE AND RETRIEVAL SYSTEM

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
THE MIDDLE EAST TECHNICAL UNIVERSITY

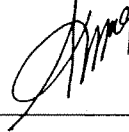
BY

KANİ KERİM GÜNER


IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE  
IN  
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2004

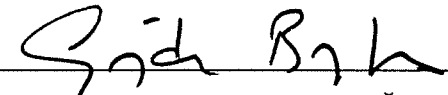
Approval of the Graduate School of Natural and Applied Sciences.

  
\_\_\_\_\_  
Prof. Dr. Canan ÖZGEN Y  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

  
\_\_\_\_\_  
Prof. Dr. Mübeccel DEMİREKLER  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

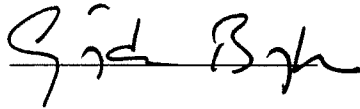
  
\_\_\_\_\_  
Assoc. Prof. Dr. Gözde BOZDAĞI AKAR  
Supervisor

Examining Committee Members

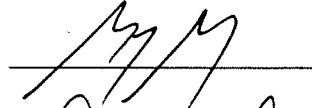
Prof. Dr. Semih BİLGİN

  
\_\_\_\_\_  
S. Bilgin

Assoc. Prof. Dr. Gözde BOZDAĞI AKAR

  
\_\_\_\_\_  
Gözde Bozdağı Akar

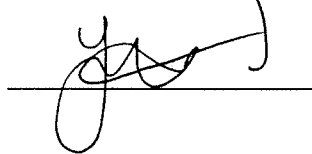
Assoc. Prof. Dr. A. Aydın ALATAN

  
\_\_\_\_\_  
A. Aydın Alatan

Asst. Prof. Dr. Cüneyt F. BAZLAMAÇCI

  
\_\_\_\_\_  
Cüneyt F. Bazlamaçcı

Yusuf KAVURUCU, M.S.

  
\_\_\_\_\_  
Yusuf Kavurucu

## ABSTRACT

### MPEG-7 COMPLIANT ORDBMS BASED IMAGE STORAGE AND RETRIEVAL SYSTEM

GÜNER, Kani Kerim

M.S., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Gözde BOZDAĞI AKAR

January 2004, 70 Pages

There is an accelerating demand to access and work over the visual content of documents. Because of the insufficiency of text-based techniques for storing this data, content-based image retrieval (CBIR) systems have become a promising field.

Due this fact, in this study a CBIR system is implemented that is Mpeg-7 compliant and ORDBMS based. The database contains images and their content summaries that are parsed from XML files. The summaries describe their dominant colors, color histograms, color spaces and labels, in order to be compliant with Mpeg-7. The query process requires only the summary not the image itself.

Software implementation of the system is based on JSP and servlet technologies using Oracle database and Tomcat web server. It is shown that the usage of these tools in the proposed architecture brings security, portability, and speed.

Keywords: MPEG-7, ORDBMS, content-based retrieval, servlet, image database.

## ÖZ

### MPEG-7 UYUMLU ORDBMS TABANLI GÖRÜNTÜ SAKLAMA VE ARAMA SİSTEMİ

GÜNER, Kani Kerim

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Gözde BOZDAĞI AKAR

Ocak 2004, 70 Sayfa

Dökümanların görsel içeriğine erişmek ve onunla ilgili çalışmak devamlı artan bir istektir. Metin bazlı tekniklerin bu veriyi saklamadaki yetersizliği nedeniyle, içerik tabanlı görüntü erişim (İTGE) sistemleri umut verici bir sektör haline geldi.

Bu nedenle, bu çalışmada Mpeg-7 uyumlu ve ORDBMS tabanlı bir İBRA sistemi gerçekleştirilmiştir. Veritabanı; görüntüleri ve görüntülerin XML dosyalarından alınan içerik özetlerini içermektedir. Özetler, Mpeg-7 uyumlu olmak için görüntülerin baskın renklerini, renk dağılımlarını, renk uzaylarını ve etiketlerini anlatır. Sorgu işlemi resmin kendisi değil sadece özet bilgisini gerektirir.

Sistemin yazılım uygulaması, JSP ve servlet teknolojilerinin Oracle veritabanı ve Tomcat web sunucusu ile birlikte kullanımı üzerine kuruludur. Bu araçların önerilen sistem yapısında kullanımının güvenlik, taşınabilirlik ve hız getirileri olduğu gösterilmektedir.

Anahtar Kelimeler: MPEG-7, ORDBMS, içerik tabanlı erişim, servlet, görüntü veritabanı.

## ACKNOWLEDGEMENTS

I would like to thank Assoc.Prof.Dr. Gzde BOZDAĐI AKAR for her valuable supervision and support throughout the development and improvement of this thesis.

I would also like to express my deep gratitude to all those who have supported and encouraged me in completing this thesis.

Finally, I would also like to thank my wife, Nuray and my family for all their support and patience during this study.

## TABLE OF CONTENTS

ABSTRACT.....	iii
ÖZ .....	iv
ACKNOWLEDGEMENTS .....	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES .....	ix
LIST OF ABBREVIATIONS.....	xi
1. INTRODUCTION .....	1
1.1. Content Based Image Retrieval (CBIR) .....	2
1.2. Related Work .....	3
1.3. Content and Outline of The Thesis .....	5
2. JAVA, TOMCAT SERVER, SERVLETS AND JSP.....	7
2.1. Java .....	7
2.1.1. Introduction to Java.....	7
2.1.2. The Java API.....	9
2.1.3. Java Database Connectivity (JDBC).....	9
2.1.4. Advantages of Java .....	10
2.2. Tomcat Web Server .....	11
2.2.1. The Structure of Tomcat Web Server .....	11
2.2.2. Advantages of Tomcat Web Server .....	12
2.3. Java Servlets and JSP.....	13
2.3.1. Server-Side Java Programming.....	13
2.3.2. The Overview of Java Servlets .....	14
2.3.3. Advantages of Servlets .....	14
2.3.4. Java Server Pages (JSP).....	15
2.3.5. Advantages of JSP .....	16

3. MPEG-7 AND XML.....	17
3.1. MPEG-7 .....	17
3.1.1. MPEG-7 Standard .....	18
3.1.2. MPEG-7 Descriptors.....	18
3.1.2.1. Dominant Color .....	19
3.1.2.2. Color Space.....	20
3.1.2.3. Scalable Color.....	20
3.1.2.4. Free Text Annotation .....	20
3.1.3. MPEG-7 Multimedia Description Schemes (DSs) .....	21
3.1.4. MPEG-7 Description Definition Language (DDL) .....	22
3.1.5. MPEG-7 System Tools .....	23
3.2. XML.....	23
3.2.1. The importance of XML .....	23
3.2.2. XML Documents .....	24
3.2.3. Document Type Definitions (DTD) and XML Schema .....	27
3.2.4. XML Tools.....	29
4. MULTIMEDIA DATABASES .....	30
4.1. Difficulties in Multimedia Databases .....	30
4.2. Multimedia Database Retrieval.....	31
4.3. Oracle .....	32
4.3.1. Information Storage .....	33
4.3.2. Properties of The Oracle ORDBMS .....	33
4.3.3. Oracle Database Structure.....	35
4.3.4. Oracle Database Objects .....	36
5. IMPLEMENTATION.....	39
5.1. Dominant Color Search.....	40
5.1.1. Dominant Color Search – Single Mode .....	41
5.1.2. Dominant Color Search – Joint Mode .....	41
5.1.3. Algorithm To Compute SVCs .....	42
5.2. Color Histogram Search.....	44
5.3. Color Label Search .....	44
5.4. Overview of The System .....	45

5.4.1. Web Pages Implementation .....	46
5.4.2. Web Server Implementation .....	47
5.4.3. Class Implementation.....	48
5.4.4. The Database Implementation .....	52
6. EXPERIMENTAL RESULTS AND CONCLUSION .....	58
REFERENCES .....	68

## LIST OF FIGURES

### FIGURES

1-1 Overview of The System .....	5
2-1 Java Components and Relations Between Them .....	8
2-2 Tomcat File System .....	12
2-3 Execution of a Java servlet .....	14
4-1 The Oracle Database Structure .....	35
5-1 The Possible Web Pages for Different Users.....	46
5-2 Relations Between All Classes in The Software of This System .....	51
5-3 Sample rows from ATTRIBUTE Table of The Database .....	52
5-4 Sample rows from AVOBJECT_CONSTANTS Table of The Database.....	52
5-5 Sample rows from AVOBJECT Table of The Database .....	53
5-6 Sample rows from CACHE Table of The Database .....	53
5-7 Sample rows from CACHE Table of The Database .....	54
5-8 Sample rows from DS Table of The Database.....	54
5-9 Sample rows from PATH Table of The Database .....	54
5-10 Sample rows from PATHATTRIBUTE Table of The Database .....	55
5-11 Sample rows from NODE Table of The Database.....	55
5-12 Sample rows from QUERIES Table of The Database.....	55
5-13 Sample rows from USERS Table of The Database .....	56
5-14 Relations Between All Tables in The Database of This System .....	57
6-1 Search results for (R, G, B) = (102, 255, 51) and label='ALL IMAGES' ....	59
6-2 ColorHistogram search results for 'untitled3.jpg' .....	60
6-3 Square and single mode dominant color search results for 'untitled3.jpg' ....	61
6-4 Square and single mode dominant color search results for 'bodie7.jpg' .....	61

6-5 Square and joint mode dominant color search results for 'bodie7.jpg' .....	62
6-6 Square and joint mode dominant color search results for 'pic3.gif' .....	63
6-7 Absolute and joint mode dominant color search results for 'pic3.gif' .....	63
6-8 Square and joint mode dominant color search results for 'pic3.gif' when threshold is increased .....	64
6-9 Absolute and joint mode dominant color search results for 'pic3.gif' when threshold is increased .....	64
6-10 Absolute and joint mode dominant color search results for 'pic2.gif' .....	65
6-11 Absolute and joint mode dominant color search results for 'pic2new.gif' ..	65
6-12 Comparison of Query Times.....	66

## LIST OF ABBREVIATIONS

API	Application Programming Interface
ASP	Active Server Pages
CBIR	Content-Based Image Retrieval
CGI	Common Gateway Interface
DDL	Description Definition Language
Ds	Descriptors
DSs	Description Schemes
DTD	Document Type Definitions
İTGE	İçerik Tabanlı Görüntü Erişimi
JDK	Java Development Kit
JVM	Java Virtual Machine
JSP	Java Server Pages
JDBC	Java Database Connectivity
MPEG	Moving Picture Experts Group
ORDBMS	Object Relational Database Management Systems
QBIC	Query By Image Content
SQL	Structured Query Language
SSI	Server-Side Includes
SVC	Similarity Value Candidates
URL	Universal Resource Location
VB	Visual Basic
XML	Extensible Markup Language
W3C	World Wide Web Consortium

# CHAPTER 1

## INTRODUCTION

It is increasingly easier to access digital multimedia information. Correspondingly, it has become increasingly important to develop systems that process, filter, search and organize this information, so that useful knowledge can be derived from the exploding mass of information that is becoming accessible.

The field of image retrieval has been an active research area for several years and has been paid more and more attention in nowadays as a result of the dramatic and fast increase in the volume of digital images. The development of Internet not only cause an explosively growing volume of digital images, but also give people more ways to get those images. The importance of an effective technique in searching and retrieving images from the huge collection can not be overemphasized. One approach for indexing and retrieving image data is using manual text annotations. The annotations can then be used to search images indirectly. But there are several problems with this approach. First, it is very difficult to describe the contents of an image using only a few keywords. Second, the manual annotation process is very subjective, ambiguous, and incomplete. Those problems have created great demands for automatic and effective techniques for content-based image retrieval (CBIR) systems. Most CBIR systems use low-level image features such as color, texture, shape, edge, etc., for image indexing and retrieval. It's because the low-level features can be computed automatically. [1]

## 1.1. Content Based Image Retrieval (CBIR)

Content-based image retrieval from image databases is an ongoing research for a long period. Digital image libraries are becoming more widely used as more visual information is produced at a rapidly growing rate. The technologies needed for retrieving and browsing this growing amount of information are still, however, immature and limited. [2]

The basic problem in CBIR is the gap between the high level semantic concepts used by humans to understand image content and the low-level visual features extracted from images and used by a computer to index the images in a database. Two most important research topics in CBIR are thus;

- The selection of the used features and the measure of similarity between them,
- The techniques for retrieving and browsing the images, i.e., how to select the images the system will display to the user.

When a CBIR system is implemented with description or summary based methods, each image in the database is transformed with a set of different feature extraction methods to a set of lower-dimensional descriptions in respective feature spaces. When the system tries to find images, which are similar to the requested images, it searches for those images whose distance to the requested images in some sense is minimal in any or all of the feature spaces. The distances between descriptions in the feature spaces can be defined in a multitude of ways, the Euclidean distance being the one used most. [3] How the distances in various feature spaces are weighted and combined in order to form a scalar suitable for minimization leaves a lot of ways for different techniques. It can be stated that in general there does not and will not ever exist one single 'correct' answer to this central question of CBIR.

## 1.2. Related Work

Some projects have been started in recent years to research and develop systems for content-based image retrieval. Query By Image Content (QBIC), VisualSeek, MIT's Photobook, FourEyes, Virage and Netra are known search engines for CBIR.

IBM's Visual Media Management research group developed the Query By Image Content (QBIC). It is a system primarily designed to query large online image databases. In addition to text-based searches, QBIC also supports access to imagery collections on the basis of visual properties such as color, shape texture and sketches. It allows users to pose queries using sample images (Query by Example) too. As a basis for content-based search, it supports color, shape, texture and layout. For example, it is possible to give a query such as 'Return the images that have blue at the top and red at the bottom', which is a color-based search with layout specification.

QBIC is one of the oldest CBIR systems, similar principles can still be found in most solutions today. The overall architecture of the QBIC system can be divided into two parts: database population and query. The purpose of the database population is to extract and store relevant information from images into a database. Database population is computationally intensive, and is therefore usually performed in an off-line fashion. The purpose of the query construction is to enable the user to compose queries and retrieve corresponding images from the database. QBIC used CGI and HTML technology. Nowadays, the Java applets are also used with QBIC.

QBIC provides some support for video data, as well; however, this support is limited to the features used for image queries: video is represented as an ordered set of representative frames (still images) and the content-based query operators used for images are applicable to video data through representative frames. Consequently, spatio-temporal relations between considerable objects and semantic content of video data are not taken into account for video querying.

VisualSEEK was created at the Image and ATV lab of Columbia University. This system allows the user to pose queries using low-level features of an image, such as color, spatial layout, and texture. These features are represented using Color Set and Wavelet Transform based texture. In VisualSEEK querying by texture feature is supported for images and video frames. The three texture measures, coarseness, contrast and orientation, are computed as a textural measure of the texture content of the objects residing in images and video frames. VisualSEEK uses the java applet technology to be reached from the web.

Photobook is developed at MIT. It is a tool for performing queries on image databases, based on image content. Basically, just like the other CBIR systems, it compares the features of images, not the images themselves. These features include color, shape, and texture. The largest difference between Photobook and other CBIR systems (e.g., QBIC) is that the Photobook system includes an interactive learning agent, called FourEyes. This agent selects and combines models based on instances from users.

FourEyes contains three stages: grouping generation, grouping weighting and grouping collection. In the generation stage, it produces many considerable groupings including both within image groupings and across image groupings induced by different models. The collection stage is guided by a rich example-based interaction with the user. The weighting stage adapts the collection stage's search space across uses, so that in later interactions, good groupings are found given few examples from the user. In spite of its many good characteristics, FourEyes system has two disadvantages. One is the use of region-to-region similarity measure, and the other is the reclustering of all the features when a new image is added. Thus it is not very scalable.

Virage is a commercial search engine that is developed by Virage Inc. It supports combination of queries. For example, users can request queries based on example to have half of the weighting ratio, while sketching and color determination each have a quarter of the weighting ratio.

Netra is developed at University of California. Netra has a fully automated segmentation algorithm. When the images are given as input to Netra, first their segments are found, then the color, shape and texture features are extracted for each segment. So it supports image retrievals based on image segments.

### 1.3. Content and Outline of The Thesis

The aim of this study is to create an image storage and retrieval system that is Mpeg-7 compliant and ORDBMS based. Overview of the system can be seen in Figure 1-1.

The system is compliant with the standard: The system can be used to upload an image with its XML file. Since the world standard for multimedia content description is MPEG-7, this XML file summarizes content of the image according to MPEG-7 descriptors.

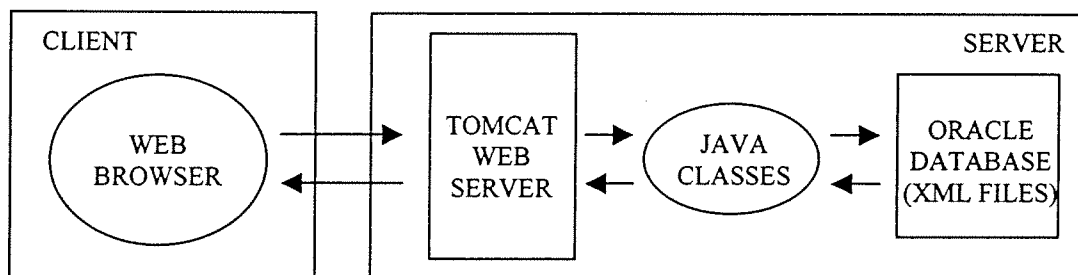


Figure 1-1 Overview of The System

The system is secure: The objects and XML files are stored in the database instead of file system to provide security. The system also supplies a session tracking mechanism for security which checks session information for each web page.

The system is fast: An ORDBMS is implemented to store objects. The 'multi-column indexing' property of this ORDBMS is used to provide fast queries.

The system is portable: Since all the components (servlets, JSPs, Tomcat web server, Oracle database) used in this system can work on other platforms the system is portable.

In chapter 2, the background components (Tomcat server, Java, Servlets, JSP) of the system are described. Their usages are explained with reasons.

In chapter 3, MPEG-7 and XML are presented in detail. The MPEG-7 descriptors used in this thesis are explained. The XML structure and its components are also shown.

In chapter 4, general information about image databases is given and specifically the Oracle database is explained in detail.

In chapter 5, the implementation of the system is described. The algorithms used in the system are given by formulas and figures.

In chapter 6, a brief illustration of the system is shown. The results of this study and comments on them are given. Finally the future work that may be done after this work is explained.

## CHAPTER 2

### JAVA, TOMCAT SERVER, SERVLETS AND JSP

In this chapter, first a programming language, Java will be described. Its components and properties will also be presented. Secondly the Tomcat web server, which is java-based web server plus servlet container, will be shown. Then the server-side applications of java, which are java servlets and Java Server Pages (JSP), will be depicted. In each part, the advantages of related topic to its traditional competitors will also be presented. By looking at Figure 1-1 the usages of these tools can be explained. Java language and servlets will be used for 'classes' part; Tomcat will be used for 'web server' part and JSP will be used for 'web browser' part.

#### 2.1. Java

##### 2.1.1. Introduction to Java

Java is an object-oriented programming language developed by Sun Microsystems. An extended definition of java can be like: "*Java is a programming language, a runtime system, a set of development tools, and an application programming interface (API)*". [4]

Java is the dominant technology bringing interactive content to the World Wide Web. A Java program is created as a text file with the file extension 'java'. It is compiled into one or more files of bytecodes with the extension '.class'. Bytecodes

are a set of instructions similar to the machine code instructions created when a computer program is compiled. The difference is that machine code must run on the computer system it was compiled for; bytecodes can run on any computer system equipped to handle Java programs. [5]

Bytecode is in a form that can be executed on the Java Virtual Machine (JVM), the core of the Java runtime system. The virtual machine can be thought as a microprocessor that is implemented in software and runs using the capabilities provided by your operating system and computer hardware. Since the JVM is not a real microprocessor, the Java bytecode is interpreted, rather than executed directly in the native machine instructions of the host computer. The Java runtime system consists of the virtual machine plus additional software, such as dynamic link libraries, that are needed to implement the Java API on the operating system and hardware.

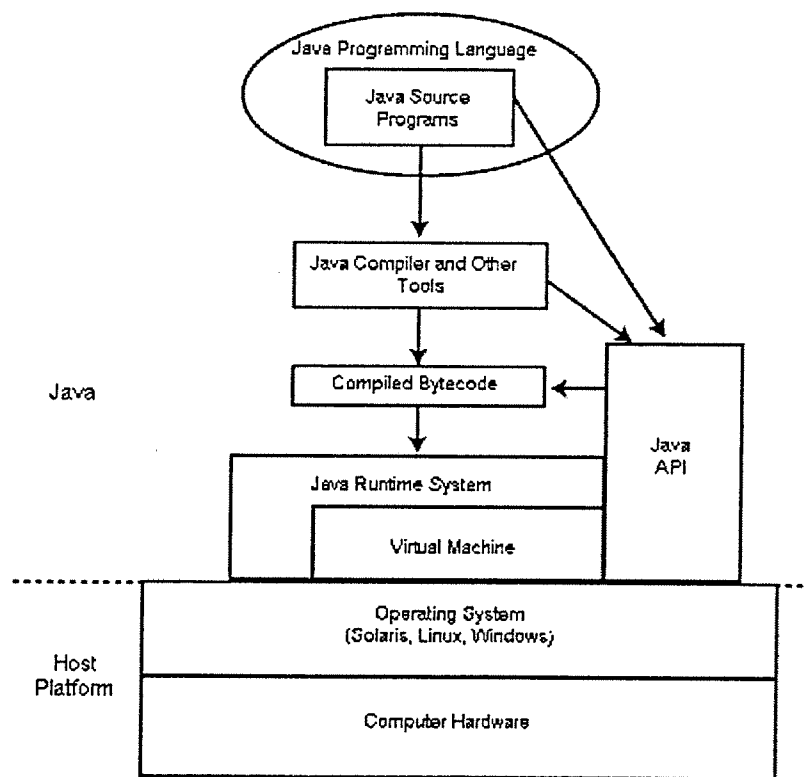


Figure 2-1 Java Components and Relations Between Them

The Java components and relations among them are illustrated in Figure 2-1. As shown in Figure 2-1, using predefined software packages of the Java API, a programmer can develop different kinds of programs in the Java language.

### **2.1.2. The Java API**

The Java Application Programming Interface (API) is a set of classes used to develop Java programs. These classes are organized into groups called packages. There are packages for the following tasks [5]:

- Numeric variable and string manipulation
- Image creation and manipulation
- File input and output
- Networking
- Windowing and graphical user interface design
- Applet programming
- Error handling
- Security
- Database access
- Distributed application communication
- JavaBeans components
- Servlet API, which creates applet-like Java programs that can run on a Web server

The powerful windowing, graphical user interface and networking features included in the Java API make it easier for programmers to develop software that is both attractive and platform independent. [7]

### **2.1.3. Java Database Connectivity (JDBC)**

During java programming sometimes the programmer needs to interact with a database. This issue is achieved by the call-level interface concept. The call-level

interface procedure is something like the following: Using standard library routines, programmer opens a connection to the database. Then JDBC can be used to send Structured Query Language (SQL) code to the database, and process the results that are returned. After the end of process, programmer closes the connection.

JDBC is a call-level programming interface, which allows connection to a database and external access to SQL manipulation and update commands. JDBC supports the integration of SQL commands into a general programming environment. This is achieved by the rich collection of library routines, which interface with the database simply and intuitively.

Since JDBC is a call-level interface, it does not require a precompilation route, which is done with Embedded SQL. For Embedded SQL, in the precompilation route the embedded SQL code is converted to the host language code. As a result with JDBC, portability increases and a cleaner client-server relationship occurs.

#### **2.1.4. Advantages of Java**

Java is simple. Most of Java's syntax is adapted from C++. The most complex parts of C++ were excluded from Java, such as pointers and memory management. Finding a pointer error in a large program is hard experience. Memory management occurs automatically in Java. [5]

Java is robust. Because the Java interpreter checks all system access performed within a program, Java programs cannot crash the system. Instead, when a serious error is discovered, Java programs create an exception. This exception can be captured and managed by the program without any risk of bringing down the system.

Java is secure. The Java system verifies all memory access. Since the Java language does not support pointers, programs cannot gain access to areas of the system for which they have no authorization. Another security level is the bytecode

verifier. Before a Java program is run, a verifier checks each bytecode to make sure that nothing suspicious is going on. [6]

Java is garbage collected. Java programs do their own garbage collection, which means that programs are not required to delete objects that they allocate in memory. This relieves programmers of virtually all memory-management problems.

Java is platform independent. Platform independence is the capability of the same program to work on different operating systems; Java is completely platform independent. A Java '.class' file of bytecode instructions can execute on any platform without alteration. The Java Development Kit (JDK) is a set of command-line tools that can be used to create Java programs. [5]

## **2.2. Tomcat Web Server**

### **2.2.1. The Structure of Tomcat Web Server**

*"A web server is an application that responds to HTTP requests by returning 'web' resources (e.g., HTML files, images, servlets, CGI output) over the Internet".* [8] Tomcat web server is written in java and so supports java elements such as servlets and JSP.

There are two possible execution modes for Tomcat. The first one is the default mode for Tomcat that is called as standalone. The second one is called as Out-of-process add-on mode. In this mode, Web server plug-in opens JVM outside web server; plug-in and JVM communicate using IPC mechanism (TCP/IP sockets and special protocol).

Tomcat has two important configuration files. First one is 'server.xml' file. This is the global configuration file of Tomcat. The ports for connections and the directories, which will be served, are defined in this file. The second one is 'web.xml' file that is the deployment descriptor for Tomcat. This file is used to

configure Tomcat web applications (contexts) by mapping webapp components and properties. The Tomcat file system is shown in Figure 2-2. The 'webapp' is the base directory that contains web applications. The 'classes' directory contains unpacked classes for web applications. The 'lib' directory contains classes packed in JAR files.

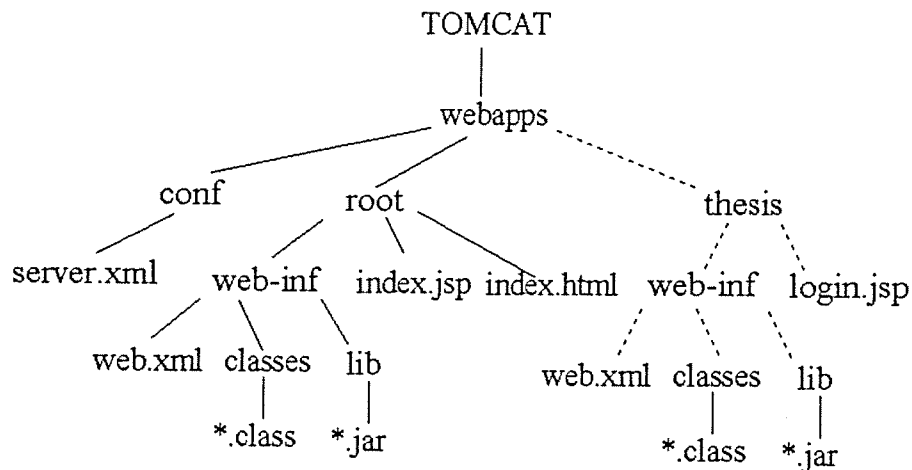


Figure 2-2 Tomcat File System

### 2.2.2. Advantages of Tomcat Web Server

Tomcat is compliant with Sun's Java standards and the open source nature of its code. It also has an ardent industry following, as evidenced by its strong support from the likes of Sun Microsystems and IBM.

A typical Tomcat installation in the server environment occupies less than 1MB. This small installation footprint keeps things running efficiently and reserves more space for web site content. The RAM consumption of Tomcat is also quite reasonable (around 620KB), so a typical managed server is capable of running hundreds of individual Tomcat JVMs without RAM contention.

As the reference implementation of Sun's Servlet and JSP specifications, Tomcat is guaranteed to support new industry products. Tomcat 3.x supports JSDK 2.2 and JSP 1.1, while Tomcat 4.x supports JSDK 2.3 and JSP 1.2.

It is the fact that Tomcat has benefited from the rigorous debugging and optimization because Tomcat is a project that is contributed by a large number of programmers.

## **2.3. Java Servlets and JSP**

### **2.3.1. Server-Side Java Programming**

When the Java language was first introduced by Sun Microsystems Inc., its purpose was to embed greater interactivity into Web pages. Java has accomplished this through the use of applets. Applets add functionality to Web pages, but because of compatibility and bandwidth issues, business has started moving to server-side Java.

Java applets are programs that are embedded directly into Web pages. When a browser loads a Web page, which contains a reference to an applet, the applet byte-code is downloaded to the client computer and executed by the browser. This is fine for small applets, but as applets grow in size the download times become unacceptable. Applets also have compatibility problems. To run an applet you must have a compatible browser. If your customer does not have a compatible browser, applets will not be presented with the proper content.

Server-side Java solves the problems that applets face. When the code is being executed on the server side, no issues arise with browser compatibility or long download times. The Java application on the server sends to the client only small packets of information, including HTML, WML, XML, and so on that it can understand. [9]

### 2.3.2. The Overview of Java Servlets

Servlets are generic extensions to Java-enabled servers. Their most common use is to extend Web servers, providing a very secure, portable and easy-to-use replacement for CGI. A servlet is a dynamically loaded module that services requests from a Web server. It runs entirely inside the JVM. Because the servlet is running on the server side, it does not depend on browser compatibility. Figure 2-3 graphically depicts the execution of a Java servlet. [9]

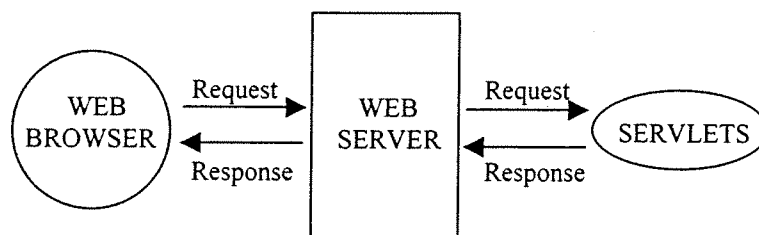


Figure 2-3 Execution of a Java servlet

### 2.3.3. Advantages of Servlets

a. Efficiency: With traditional Common Gateway Interface (CGI), a new process is started for each HTTP request. If the CGI program does a relatively fast operation, the overhead of starting the process can dominate the execution time. With servlets, the JVM stays up, and each request is handled by a lightweight Java thread, not a heavyweight operating system process. Similarly, in traditional CGI, if there are N simultaneous request to the same CGI program, then the code for the CGI program is loaded into memory N times. With servlets, however, there are N threads but only a single copy of the servlet class. Servlets also have more alternatives than do regular CGI programs for optimizations such as caching previous computations, keeping database connections open, and the like.

b. Power: Java servlets let you easily do several things that are difficult or impossible with regular CGI. For one thing, servlets can talk directly to the Web server (regular CGI programs can't). This simplifies operations that need to look up images and other data stored in standard places. Servlets can also share data among each other, making useful things like database connection pools easy to implement. They can also maintain information from request to request, simplifying things like session tracking and caching of previous computations.

c. Portability: Servlets are written in Java and follow a well-standardized API. Consequently, servlets written for Apache Tomcat Server or Microsoft IIS can run virtually unchanged on another server. Servlets are supported directly or via a plugin on almost every major Web server.

d. Cost: There are a number of free (Tomcat) or very inexpensive Web servers available that are good for servlet-included Web sites. Nevertheless, once you have a Web server, no matter the cost of that server, adding servlet support to it (if it doesn't come preconfigured to support servlets) is generally free or cheap. [10]

#### **2.3.4. Java Server Pages (JSP)**

JSPs are HTML or XML pages with embedded Java code to generate dynamic contents. They are text-based documents, which describe how to process a request and to generate a response. They both include template data in HTML/XML and some dynamic actions in Java.

JSPs are processed on the server side. When the first time a JSP is invoked JSP's are compiled (by a JSP compiler) to servlets. The clients receive the processed results of JSP's, not the source. [11]

### 2.3.5. Advantages of JSP

a. Active Server Pages (ASP): ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.

b. Server-Side Includes (SSI): SSI is a widely supported technology for including externally defined pieces into a static Web page. JSP is better because it lets you use servlets instead of a separate program to generate that dynamic part. Besides, SSI is really only intended for simple inclusions, not for 'real' programs that use form data, make database connections, and the like.

c. Static HTML: Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

d. JavaScript: JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment. With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like. [10]

## **CHAPTER 3**

### **MPEG-7 AND XML**

As previously mentioned, there is a large collection of digital imagery information in the world. Since this information comes from different sources, a system to manage it gets importance. In this chapter, an international standard of ISO/IEC, namely MPEG-7, for this issue will be described. MPEG-7 parts are constructed by means of MPEG-7 Description Definition Language. This language uses a schema based on Extensible Markup Language (XML). So the XML, its importance and its parts will also be presented in this chapter.

#### **3.1. MPEG-7**

For content-based retrieval of digital imagery some systems have been developed. But there is an inconsistency over these systems because each system uses its own features, comparison algorithms. For instance, it is possible to find similar images to the original one in one system, but it is impossible to compare the results with the images in other systems with the same parameters as they all use different feature extraction and comparison algorithms. In October 1996, MPEG (Moving Picture Experts Group) started a new work item to provide a solution for the urging problem of generally recognized descriptions for audio-visual content, which extend the limited capabilities of proprietary solutions in identifying content that exist today. [12]

MPEG-7 is the first complete work for description of multimedia data and it is become an international standard of ISO/IEC in September 2001. MPEG-7 is a work to define a standard set of descriptors for various types of multimedia data and methods to define other descriptors as well as structures of descriptors and their relationships. Although MPEG-7 is not aimed at any particular application area, one of its main applications areas will be searching and retrieving multimedia content. In image retrieval, a natural step towards MPEG-7 is to use MPEG-7 content descriptors suitable for still images in retrieval systems. [13]

### **3.1.1. MPEG-7 Standard**

The MPEG-7 standard has four important parts. These are Descriptors (Ds), Description Schemes (DSs), Description Definition Language (DDL) and System Tools. MPEG-7 specifies a standard set of Descriptors to describe various types of multimedia information. MPEG-7 also standardizes Description Schemes to define other Ds as well as DSs for the structure of Ds and their relationships. This description (i.e. the combination of descriptors and description schemes) is associated with the content itself to allow fast and efficient searching for material of a user's interest. MPEG-7 standardizes a language to specify description schemes too, i.e. a Description Definition Language (DDL). The DDL is used to construct new Ds, DSs and specify the schemes for encoding the descriptions of multimedia content. [14] Finally MPEG-7 standardizes System Tools, to support efficient storage and transmission mechanisms, multiplexing of descriptions, synchronization of descriptions with content, etc.

### **3.1.2. MPEG-7 Descriptors**

MPEG-7 Descriptors define the syntax and semantics for the representation of features. The most important (primary) descriptor categories of visual items cover the features of color, texture, shape and motion. [15]

### *Color*

Color provides valuable information about the nature and illumination of objects. To distinguish different identically shaped objects one from another, it is in general sufficient to know color histogram and some of the dominant colors of an object.

### *Texture*

The material an object is made of is visually translated into a texture. The material characteristics are specified by perceptual texture descriptions. The actual object material is given by its texture image.

### *Shape*

Shape is an important visual feature for object recognition and classification both by humans and machines. But while a simple shape description usually suffices for classification, in unambiguous object recognition the original shape has to be matched closely.

### *Motion*

Motion represents the trajectory of objects. The action of a video scene is given by its perceptual motion description. To move rigid objects, it suffices to know the trajectory of one representative point. [16]

The components of MPEG-7 DSs which are used in this thesis are Dominant Color, Color Space, Scalable Color (Color Histogram) and Free Text Annotation.

#### **3.1.2.1. Dominant Color**

This color descriptor is most suitable for representing local (object or image region) features where a small number of colors are enough to characterize the color information in the region of interest. Whole images are also applicable, for example, flag images or color trademark images. Color quantization is used to extract a small

number of representing colors in each region/image. The percentage of each quantized color in the region is calculated correspondingly. A spatial coherency on the entire descriptor is also defined, and is used in similarity retrieval.

#### **3.1.2.2. Color Space**

The color space is generally used in conjunction with other color descriptors (i.e. Dominant Color). Possible color spaces include RGB, YCrCb, HSV, HMMD and monochrome. If some linear combinations of RGB are used, extra color spaces can be included.

#### **3.1.2.3. Scalable Color**

The Scalable Color Descriptor is a Color Histogram in HSV Color Space, which is encoded by a Haar transform. Its binary representation is scalable in terms of bin numbers and bit representation accuracy over a broad range of data rates. The Scalable Color Descriptor is useful for image-to-image matching and retrieval based on color feature. Retrieval accuracy increases with the number of bits used in the representation. The similarity between two images is the hamming distance between their Scalable Color descriptors.

#### **3.1.2.4. Free Text Annotation**

Text annotation is an important component of DSs. MPEG-7 provides a number of different basic constructions for textual annotation. The most flexible text annotation construction is the data type for free text. Free text annotation allows the formation of an arbitrary string of text, which optionally includes information about the language of the text. [17]

### 3.1.3. MPEG-7 Multimedia Description Schemes (DSs)

Description Schemes (DSs) specify the structure and semantics of the relationships between their components, which may be both Descriptors and Description Schemes. The DSs provide a standardized way of describing in XML the important concepts related to AV content description and content management in order to facilitate searching, indexing, filtering, and access. The DSs are defined using the MPEG-7 Description Definition Language (DDL), which is based on the XML Schema Language, and are instantiated as documents or streams. The resulting descriptions can be expressed in a textual form (i.e., human readable XML for editing, searching, filtering) or compressed binary form (i.e., for storage or transmission).

The MPEG-7 DSs are designed primarily to describe regions, segments, objects, events which are high-level AV features and also some constant metadata like title, author, creation date. By combining Ds, DSs and presenting relationships between them complex descriptions can be defined. In MPEG-7, the DSs are categorized according to their related media such as audio, visual domain or multimedia. Typically, the multimedia DSs describe content consisting of a combination of audio, visual data, and possibly textual data, whereas, the audio or visual DSs refer specifically to features unique to the audio or visual domain, respectively. [17]

The DSs can be grouped into 5 different classes according to their functionality.

- a. Content description: Representation of perceivable information
- b. Content management: Information about the media features, the creation and the usage of the AV content;
- c. Content organization: Representation the analysis and classification of several AV contents;
- d. Navigation and access: Specification of summaries and variations of the AV content;

e. User interaction: Description of user preferences and usage history pertaining to the consumption of the multimedia material.

#### **3.1.4. MPEG-7 Description Definition Language (DDL)**

For MPEG-7, one of the main works was to define a language (Description Definition Language) to describe the Ds and DSs. During the design process of Description Definition Language (DDL), the designers concerned to achieve following points by DDL:

- Prevention of being sophisticated in other words satisfying the main features and also ability to be extended;
- Coverage of MPEG-7 requirements and usage of MPEG-7 Ds, DSs;
- Usage of XML structure and interoperability with XML Schema.

The DDL is the language which allows the creation of MPEG-7 DSs and Ds. A DDL schema (a DDL file) specifies the constraints that a valid MPEG-7 description should respect. It is encoded in XML.

The DDL uses XML Schema for interoperability reasons. However because multimedia content descriptions require specific features that are not defined in XML Schema (such as array and matrix datatypes), the DDL adds those features to the language. Hence the MPEG-7 DDL adopts most the of the XML Schema specification and adds MPEG-7-specific mechanisms on top of it.

Some of the important issues, which are supported by DDL, are extended XML Schema, structural and datatype constraints over Ds and DSs, usage of Ds, DSs in other DSs, group definitions like Attribute Group for simple DSs definitions.

With the DDL one can express structural constraints and data type constraints. Structural constraints specify the rules that a valid description should respect in terms of inclusion of elements. Data type constraints specify the type and

the possible values for data within the description. The DDL allows defining complexTypes and simpleTypes. The complexTypes specify the structural constraints while simpleTypes express datatype constraints. Moreover the DDL allows reusing the existing complexTypes or simpleTypes in a way similar to inheritance in object-oriented programming. [18]

### **3.1.5. MPEG-7 System Tools**

Some applications of MPEG-7's system tools are support binary coded representation for efficient storage and transmission, transmission mechanisms (both for textual and binary formats), error resilience, multiplexing of descriptions, synchronization of descriptions with content, management and protection of intellectual property in MPEG-7 descriptions. These issues result in the ability to search, browse, and filter MPEG-7 descriptions.

## **3.2. XML**

A standard format, Extensible Markup Language (XML), was developed by an XML Working Group formed under the auspices of the World Wide Web Consortium (W3C) in 1996. XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents. [19] XML is a set of rules for designing text formats that let you structure your data. XML makes it for a computer to generate data, read data, and ensure that the data structure is unambiguous. [20]

### **3.2.1. The importance of XML**

In XML, rather than specifying where to display content, Web builders can specify the content of the document. XML source document can be written once, then displayed in a variety of ways: on a computer monitor, within a cellular-phone

display, translated into voice on a device for the blind, and so forth. It'll work on any communications devices that might be developed; an XML document can thus outlive the particular authoring and display technologies available when it was written. [21]

XML provides a facility to define tags and the structural relationships between them. Since there's no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by style sheets. HTML, as we have already discussed, comes bound with a set of semantics and does not provide arbitrary structure. SGML provides arbitrary structure, but is too difficult to implement just for a web browser. Full SGML systems solve large, complex problems that justify their expense. Viewing structured documents sent over the web rarely carries such justification. [22]

### **3.2.2. XML Documents**

XML documents are plain text files stored in ASCII file format. Thus it very easy to pass data between different modules of a complex application in the form of XML files. XML files can be easily accessed in a platform independent and language independent manner. That is an XML document can be parsed by a Java program with no coordination with the C program that produced it. Thus XML is likely to become the standard for data exchange between applications. [23]

XML documents simply define the data representation and do not deal with the presentation. Like HTML, XML makes use of tags (words bracketed by '<' and '>') and attributes (of the form name="value"). But unlike HTML, XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it. XML 1.0. is the specification that defines what 'tags' and 'attributes' are. XML documents should begin with an XML declaration, which specifies the version of XML being used. For example, "<?xml version='1.0'?>" can be an acceptable first line of a XML document. [20]

Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a 'root' or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup.

No part of the root, or document element appears in the content of any other element. For all other elements, if the start-tag is in the content of another element, the end-tag is in the content of the same element. More simply stated, the elements, delimited by start- and end-tags, nest properly within each other. As a consequence of this, for each non-root element C in the document, there is one other element P in the document such that C is in the content of P, but is not in the content of any other element that is in the content of P. P is referred to as the parent of C, and C as a child of P. [19] Below, a possible XML document that is used in this study to summarize the object information is given:

```
<?xml version="1.0" ?>
<Mpeg7>
  <DescriptionUnit type="DescriptorCollectionType">
    <Descriptor size="3" type="DominantColorType" input ="agri_hills.jpg">
      <ColorSpace type="RGB"/>
      <Values>
        <Percentage>70</Percentage>
        <ColorValueIndex>108 96 88</ColorValueIndex>
      </Values>
      <Values>
        <Percentage>14</Percentage>
        <ColorValueIndex>168 116 92</ColorValueIndex>
      </Values>
      <Values>
```

```

<Percentage>8</Percentage>
<ColorValueIndex>164 176 216</ColorValueIndex>
</Values>
</Descriptor>
<Descriptor numberOfCoefficients="16" type="ColorHistogramType">
<Coefficients>0.010023907 0.019006148 0.01675205 0.051827185 0.1551571
0.31615436 0.20520833 0.1148224 0.06372951 0.34528688 0.010672 0.00184
2.5614753E-4 1.7076503E-5 0.208755 0.0</Coefficients>
<Coefficients>0.130023607 0.01507618 0.10385206 0.071927186 0.13759501
0.21415839 0.10727533 0.1049224 0.1038247 0.017588698 0.030952 0.10164
1.8604743E-4 2.9075503E-5 0.0 0.0</Coefficients>
<Coefficients>0.119271867 0.038007108 0.51947905 0.152797185 0.2527878
0.26384936 0.15269353 0.269264 0.186972951 0.14822669 0.479283 0.14274
3.7293753E-4 6.9262905E-5 0.8863789 0.21567789</Coefficients>
</Descriptor>
</DescriptionUnit>
<DescriptionMetadata>
<Comment>
<FreeTextAnnotation>Hills</FreeTextAnnotation>
</Comment>
</DescriptionMetadata>
</Mpeg7>

```

In the sample XML document the root is declared to be <Mpeg7>. It is also a document element since there exists a </Mpeg7> expression. Be careful that every element that is constructed by <element>, should be ended by </element>. Realize the case sensitivity of XML. If the element is declared by <ElemENt> then it should be ended by </ElemENt> not <Element> or </ELEMENT>. In the <ColorSpace type="RGB"/> expression 'type' is the attribute of ColorSpace and its value is 'RGB'. The ColorSpace attribute gives extra information of Descriptor element.

### 3.2.3. Document Type Definitions (DTD) and XML Schema

A Document Type Definition is the grammar of an XML page. It contains the elements, attributes, entities, and notations used in the XML document. XML is a markup language to create other markup languages, and the DTD is the tool you use to create and describe your language. Once a DTD is created and a document written based on that DTD, the document is then compared to the DTD. This is called validation. If your XML document follows the rules listed in the DTD, then the document is said to be valid. XML documents that do not follow the rules in their DTDs are called invalid. Here is part of a simple DTD. It states that there is a root element called 'family' that has two possible elements within it: 'parent' and 'child':

```
<!DOCTYPE family [  
  <!ELEMENT parent (#PCDATA)>  
  <!ELEMENT child (#PCDATA)>  
>
```

This would be a valid XML document:

```
<family>  
  <parent>Judy</parent>  
  <parent>Layard</parent>  
  <child>Jennifer</child>  
  <child>Brendan</child>  
</family>
```

But if some extra text is added outside of the <parent> or <child> tags, the document would be invalid:

```
<family>  
This is my family:  
  <parent>Judy</parent>  
  <parent>Layard</parent>
```

```
<child>Jennifer</child>
<child>Brendan</child>
</family>
```

XML Schema help developers to precisely define their own XML-based format structures. XML allows you to define a new document format by combining and reusing other formats. Since two formats developed independently may have elements or attributes with the same name, care must be taken when combining those formats (does '`<p>`' mean 'paragraph' from this format or 'person' from that one?). XML Schema is designed to mirror this support for modularity at the level of defining XML document structures, by making it easy to combine two schemas to produce a third which covers a merged document structure. [20]

String, boolean, decimal, float, double, duration, dateTime, time, date, hexBinary, anyURI are the primitive data types defined in XML Schema. NMTOKEN, NMTOKENS, token, language, integer, NonPositiveInteger, long, short, Byte, unsignedLong, positiveInteger are the built-in data types in XML Schema, which are generated from the primitive data types. A new data type can be defined from an existing data type (called the 'base' type) by specifying values for one or more of the optional facets for the base type. The string primitive datatype has six optional facets: length, minLength, maxLength, pattern, enumeration, whitespace. Here is an example of creating a new data type by specifying facet values:

```
<xsd:simpleType name="TelephoneNumber">
  <xsd:restriction base="xsd:string">
    <xsd:length value="8"/>
    <xsd:pattern value="\d{3}-\d{4}"/>
  </xsd:restriction>
</xsd:simpleType>
```

This creates a new data type called 'TelephoneNumber'. Specify its elements to hold string values. But the string length must be exactly 8 characters long and the string must follow the pattern: ddd-dddd, where 'd' represents a 'digit'. [25]

#### 3.2.4. XML Tools

Beyond XML 1.0, 'the XML family' is a growing set of modules that offer useful services to accomplish important and frequently demanded tasks. Xlink describes a standard way to add hyperlinks to an XML file. XPointer and XFragments are syntaxes in development for pointing to parts of an XML document. An XPointer is a bit like a URL, but instead of pointing to documents on the Web, it points to pieces of data inside an XML file. CSS, the style sheet language, is applicable to XML as it is to HTML. XSL is the advanced language for expressing style sheets. It is based on XSLT, a transformation language used for rearranging, adding and deleting tags and attributes. The DOM is a standard set of function calls for manipulating XML (and HTML) files from a programming language. It is a kind of parser. [20]

XML documents are manipulated programmatically by transforming XML documents into Objects that conform the Document Object Model or DOM, specified by the W3C. Once a DOM representation of an XML document is created, it can be manipulated through any Object based (javascript, VB etc) or Object-Oriented language. Parsers are available that convert an XML text file into a DOM object. Sun provides the Java API for XML or JAXP, which provides complete support working with XML documents. This support comes in the form of nine packages. Between these packages the most important ones are the javax.xml.parsers, org.w3c.dom for parsing , creating and navigating through XML documents. [23]

## **CHAPTER 4**

### **MULTIMEDIA DATABASES**

The rapid growth of the World Wide Web and the use of digital data in the preparation of paper documents mean that millions of people now access multimedia documents daily. Multimedia documents contain audio, image and video. There is thus a need for systems that allow users to create, manage and query multimedia databases in an efficient and accurate manner.

#### **4.1. Difficulties in Multimedia Databases**

The first difficulty about the multimedia databases is storage. A multimedia file usually contains more data when compared to text files. In recent years some compression techniques are discovered for this issue. So some new formats occurred namely jpg, gif, tif, mp3, mpg, etc. But nowadays another question is in minds. 'Is it secure to save the image or other data as a file in file systems?'. To come over this problem the databases now support saving multimedia data (very large data) in.

Another difficulty occurs for the user interface and query designs. [26] Since the knowledge about a multimedia data is large, the user interface is generally complex to understand. If the programmer gives the query language control to the user it is more complex then. So to be well remembered, a program should include simple but efficient user interface and query language.

May be the most difficult thing during the construction of a multimedia database is how to tell the computer 'What is the content of object stored in it?'. An image storage process to the computer is a good example for this condition. The computer can understand when an image is stored in it. Then it can also know the name, creation time, author, path, format, size, width, height, and resolution of that image. But with this information it cannot separate a car picture from a human picture. The traditional method is to enter a list of keywords about the pictures. When a program searches over all the images actually it only searches over the keyword list of the images. It does not care content of the image. So if two image both containing cars parked in a parking lot may be entered as 'cars', 'parking lot' etc. However it is possible to have red cars in first image and blue ones in the second.

The attachment of text labels to multimedia is inadequate, since identical multimedia data can be described in different ways. The users may enter a word for searching that is not included in the database. Also it is possible that the programmer would not describe the multimedia data sufficiently. Actually, vocabulary indexing is now deemed insufficient even in text retrieval systems. [27]

Consequently, there is great interest in content-based retrieval systems. Because, a content-based system retrieves multimedia data from a database based on the content similarities.

#### **4.2. Multimedia Database Retrieval**

Multimedia Database Management Systems (MDBMS) aim to store large collections of multimedia data, and to support efficient content-based retrieval of this data. Like any other DBMS, a MDBMS has a storage facility, the data is arranged according to some data model, and it has a query facility, the multimedia data is retrieved according to the queries. [28] Generally the selection queries are used. For example, selecting a few images from the database, based on their content. Imagine a query facility that loads all the images from the database and checks each image to see if it fits the selection criterion. In all but extreme cases this query facility would

be inefficient. The only other option is to have more information about the images stored in the database so it can be used for evaluating the selection criterion. This may be called indexing. When a multimedia data is stored in the database, an index is constructed and stored as well for that data. Queries are executed using only the information in the indexes. Multimedia is retrieved only after we check that the index conforms to the selection criterion. This means that any content information we want to query must be in the index. Therefore, the type of information in the index limits the expressiveness of any MDBMS query language.

Efficient processing of queries in a database requires sophisticated indexing techniques. Content-based querying of multimedia databases requires development of indexing techniques that are significantly different than traditional DBMSs. In general, two types of indexes occur: text indexes to facilitate queries of the type "Find all the images that are labeled as 'car'" and content similarity indexes that facilitate queries of the type 'Find all the images that are similar, in color, to this sample image'. [29]

When an MDMBS is wanted to be constructed, some extra data types (arrays, images, etc.) are required that are not in the standard relational data model. So the need of an object relational data model arises with the ability to store large uninterpreted blobs (for arrays, images, etc.). This type of DBMSs are called Object-Relational DBMS namely ORDBMS. One famous and powerful ORDBMS is Oracle.

#### **4.3. Oracle**

The information age has ushered in a new competitive era where business opportunities are discovered and exploited more quickly than ever before. To be successful in this new environment, businesses are challenged to deploy information solutions that are both powerful and flexible. [32] (Two characteristics that have traditionally been competing objectives)

#### **4.3.1. Information Storage**

Every organization has some information needs. A library keeps a list of members, books, due dates, and fines. A company needs to save information about employees, departments, and salaries. These pieces of information are called data. Organizations can store data on various media and in different formats; for example, a hard-copy document in a filing cabinet or data stored in electronic spreadsheets or in databases.

A database is an organized collection of information. To manage databases, you need database management systems (DBMS). A DBMS is a program that stores, retrieves, and modifies data in the database on request. There are four main types of databases: hierarchical, network, relational, and more recently object relational.

An object-relational DBMS is composed of three basic components: Collections of objects or relations that store the data, a set of operators that can act on the relations to produce other relations, data integrity for accuracy and consistency. To access the database, you execute an SQL statement. The language contains a large set of operators for partitioning and combining relations. The programmer can modify the database by means of SQL statements.

#### **4.3.2. Properties of The Oracle ORDBMS**

On January 17, 1995, Oracle Corporation announced the release of Personal Oracle, which becomes the most popular relational database in the world. At that time Oracle decided to distribute a trial version of Personal Oracle to its potential customers by making Personal Oracle available for downloading via Oracle Corporation's Web site (<http://www.oracle.com>). The company has since expanded this concept with a 'try and buy' option that allows existing and potential customers to experiment with trial versions of many Oracle products. [30]

Oracle is the most popular relational database management system in the world. Its popularity is the result of several factors: Oracle is available on a wide range of computer systems. Oracle Corporation continues to add powerful features to each new release of the product. The exponential growth in the quantity of data maintained by organizations requires convenient and reliable data repositories.

Oracle is a multi-user database management system. A software package specializing in managing a single, shared set of information among many concurrent users. Oracle is one of the database servers that can be plugged into a client/server equation. Oracle works to efficiently manage its resource, a database of information, among the multiple clients requesting and sending data in the network. [31]

Oracle software runs on network computers, personal digital assistants, set-top devices, PCs, workstations, minicomputers, mainframes and massively parallel computers. [32]

Oracle ORDBMS provides a new engine that brings object-oriented programming, complex datatypes, complex business objects, and full compatibility with the relational world. Operating within the Network Computing Architecture (NCA) framework, it supports client-server and Web-based applications that are distributed and. Oracle ORDBMS can scale tens of thousands of concurrent users, support up to 512 petabytes (equals approximately  $65536 * 8192$  GB), and can handle any type of data, including text, spatial, image, sound, video, and time series as well as traditional structured data. [33]

Oracle ORDBMS includes a robust, integrated, and scalable JVM within the server (Jserver), thus supporting Java in all tiers of applications. This eliminates the necessity of recompiling or modifying Java code when it is to be deployed on a different tier. An efficient JDBC driver is also included in Oracle database.

Oracle database has additional features that are usually not supported by other famous databases. 'Foreign Key Clauses', synonyms (aliases for tables,

objects, etc.), languages for writing stored procedures (PL/SQL, Java), roles (to control access rights of groups), incremental backups, multi-column indexes (up to 32 columns), and automatic recovery from failures are some examples of these features. [34]

### 4.3.3. Oracle Database Structure

The Oracle database has two sections called physical and logical structures. The physical side is viewed from the operating system. This includes datafiles, controlfiles, and logfiles. The logical side includes tablespaces, tables, views, sequences, synonyms and users. The logical side can be viewed only by connecting to oracle. Every object in the database belongs to a user. An object is logically stored in its user's tablespace and physically stored in a datafile of that tablespace. But this datafile cannot be viewed by the operating system. In Figure 4-1 structure of the Oracle database is shown.

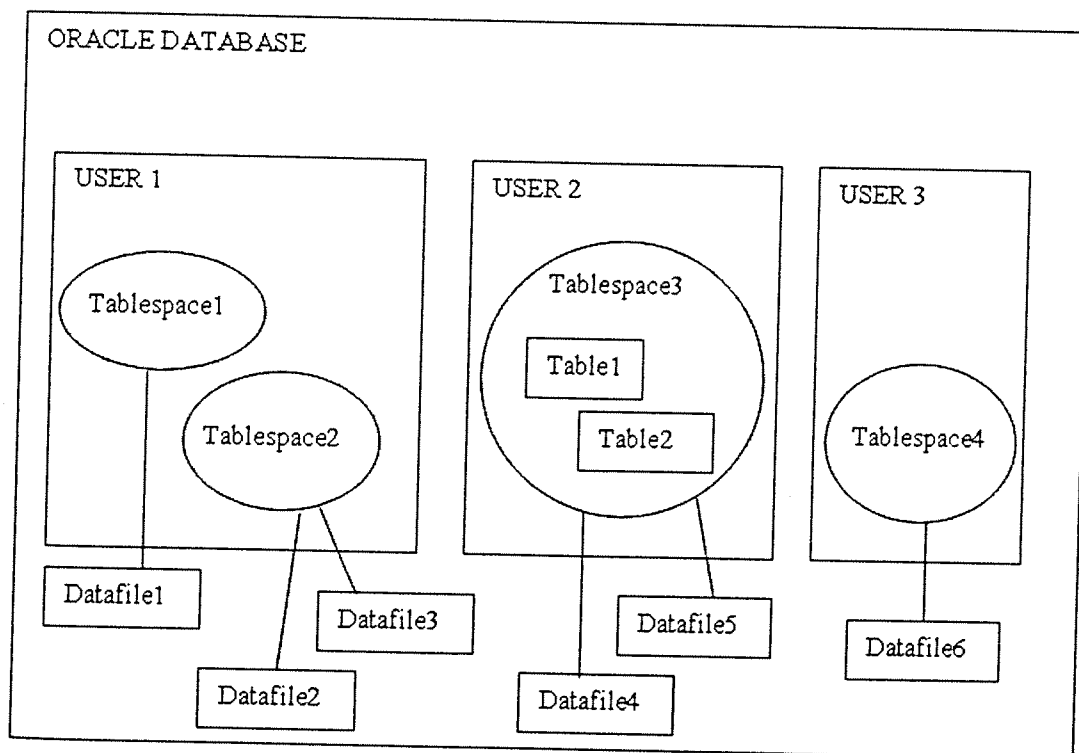


Figure 4-1 The Oracle Database Structure

#### 4.3.4. Oracle Database Objects

Some of the important Oracle database objects are described below.

##### *Tablespaces*

A database is divided into one or more logical storage units called tablespaces. Every Oracle database contains a tablespace named SYSTEM, which Oracle creates automatically when the database is created. The USER\_DATA tablespace includes all the objects of users. Tablespaces are divided into logical units of storage called segments, which are further divided into extents. A segment is a section of the database for an object using space in the database. An extent is the space that an object takes to increase its total space in the database. Every object has a space in the database that is defined during the construction of the database.

When the space of an object becomes insufficient for that object, that object takes an extent and increases its space. This process can be occurred until the number of extents reaches the number of 'max extent' that is defined during the segment definition.

##### *Datafiles*

A tablespace in an Oracle database consists of one or more physical datafiles. A datafile can be associated with only one tablespace and only one database. Oracle creates a datafile for a tablespace by allocating the specified amount of disk space plus the overhead required for the file header. A database's data is collectively stored in the datafiles. The names of datafiles are look like 'data1.ora, data2.ora'.

##### *Control Files*

Every Oracle database has a control file. A control file contains entries that specify the physical structure of the database. For example, it contains the following types of information: database name, names and locations of a database's datafiles and redo log files, time stamp of database creation. Oracle allows the control file to

be multiplexed for protection of the control file. Every time an instance of an Oracle database is started, its control file is used to identify the database. If the physical makeup of the database is altered (for example, a new datafile is created), the database's control file is automatically modified by Oracle to reflect the change. [35]

### *Tables*

A table is the basic unit of data storage in an Oracle database. The tables of a database hold all of the user-accessible data. Table data is stored in rows and columns. Every table is defined with a table name and set of columns. Each column is given a column name, a datatype (such as CHAR, DATE, or NUMBER), and a width (which may be predetermined by the datatype, as in DATE) or scale and precision (for the NUMBER datatype only). Once a table is created, valid rows of data can be inserted into it. The table's rows can then be queried, deleted, or updated. To enforce defined business rules on a table's data, integrity constraints and triggers can also be defined for a table.

### *Sequences*

A sequence generates a serial list of unique numbers for numeric columns of a database's tables. Sequences simplify application programming by automatically generating unique numerical values for the rows of a single table or multiple tables. For example, assume two users are simultaneously inserting new employee rows into the EMP table. By using a sequence to generate unique employee numbers for the EMPNO column, neither user has to wait for the other to input the next available employee number. The sequence automatically generates the correct values for each user. Sequence numbers are independent of tables, so the same sequence can be used for one or more tables. After creation, various users can access a sequence to generate actual sequence numbers.

### *Users*

The owners of the objects in the Oracle database are called users. The users construct tables, use them and delete them. When the Oracle database is installed there exist three default users namely SYS, SYSTEM, SCOTT. The first two users

are the system users. In default they have the ability to use all objects of all users. The third one, Scott, is the owner of the training tables of Oracle database. The postdefined users can use the tables of Scott.

## CHAPTER 5

### IMPLEMENTATION

In this thesis study, an ORDBMS based image storage and retrieval system is implemented. As previously mentioned, since the multimedia content description standard in the world is MPEG-7, the system is MPEG-7 compliant.

The objects and their XML files are stored in the database to supply a security level because datafile of the database can be seen only from the ORDBMS. The system creates a session for each user with some attributes in order to supply another security level. These attributes are checked when a web page is requested.

This system is portable because all technologies used in this system are platform independent. If the 'configuration.xml' file is modified, this system can work on another platform.

One-column indexing (primary key) is not enough for fast queries over this system. Consequently, 'multi-column indexing' property of Oracle ORDBMS is used to provide fast queries.

For each visual object a separate XML file is considered. This file can be called as a summary, a description or an index of that object. In this XML file; dominant colors, color histograms, labels, colorspace, etc. are stored. A sample XML file can be seen in Section 3.2.2. This sample file is created by means of the

system proposed in [36]. However other feature extractor systems can also be used as input to the system of this thesis. [37]

The search and retrieval process is done by means of the contents of these XML files. This process can be done in three ways: Dominant color, Color histogram and Color label search. The important point here is the similarity measure between images. The color histogram search is done like other systems. But in Color label and dominant color searches some experiments are done and they will be shown in the following pages. In all of the three search techniques each image is given a similarity value that measures its similarity to the requested image. According to this similarity value a list is prepared and the most similar images are shown.

Since the ORDBMSs are used in these types of studies at an increasing rate, an ORDBMS is prepared. All of the information in XML files is parsed using Xerces parser and it is stored in this database. The search techniques mentioned above are processed over this database.

An image is thought as a composition of segments. Each object that attracts attention may be a segment. So an image may include several segments. But even if there is nothing to care specially, the image has one segment that corresponds to it.

The following search algorithms run over these segments (images). In other words each segment is taken as a new image and so it is compared to segments of all other images.

### **5.1. Dominant Color Search**

Dominant color search tries to find the similarity value between two images by means of dominant color information stored in the database. All of the images in the database are summarized with a number of dominant colors and also the size (color ratio) of their regions in the image. For the first step these values are read from the database that belong to original image. The second step is to read these values for

another image and compare them by original image. The similarity value computed is stored until the entire process end. The third step includes the above processes for all of the images in the database. In other words, the dominant colors and color ratios of all images are read and compared to the original image. After these steps, we have the similarity values. These values are sorted in increasing order. The most similar images are shown as the search result. The dominant colors of them are displayed for each one. These dominant colors can be used for Color label search by clicking them.

According to computation of similarity values, dominant color search can be divided in to two types. The first one is 'Dominant Color Search – Single Mode' and the second one is 'Dominant Color Search – Joint Mode'.

#### **5.1.1. Dominant Color Search – Single Mode**

This can also be called as single mode. This mode is useful for images having dominant colors with large color ratios. In single mode, following steps are done to compute the similarity vale between two images:

- a. Count the number of segments for original and compared images:  $SG_o$ ,  $SG_c$
- b. Count the number of dominant colors for each segment in original and compared images:  $DCSG_o[i]$ ,  $DCSG_c[j]$ ,  $i: 1 \dots SG_o$ ,  $j: 1 \dots SG_c$
- c. Make the comparisons segment by segment, i.e.  $i^{th}$  segment from original image and  $j^{th}$  from compared image. For each segment comparison actually there are  $DCSG_o[i] \times DCSG_c[j]$  dominant color comparisons resulting  $DCSG_o[i] \times DCSG_c[j]$  similarity value candidates (SVC).
- d. Among these candidates select the maximum one. This one is called similarity value between these two images.

#### **5.1.2. Dominant Color Search – Joint Mode**

Similar to single mode, this can be called as multi mode. In joint mode the similarity value is computed similar to single mode except item d. In item d, the

computed similarity value candidates are summed for their segments then they are normalized and renamed as segment candidates. In other words the similarity value candidates are now for segments. Afterwards, the maximum segment candidate is selected as the similarity value between these two images. This mode is useful for images having dominant colors with medium color ratios. The item c in both single and joint mode is called as 'Algorithm To Compute SVCs'.

### 5.1.3. Algorithm To Compute SVCs

When an image is being compared with another image, there are totally

$$\sum_i^{SG_o} \sum_j^{SG_c} DCSG_o[i] \times DCSG_c[j]$$

SVCs for this comparison. In SVC computation the L2 norm (squared root) is used. We represented the dominant color components of the original and compared images as  $R_o$ ,  $G_o$ ,  $B_o$  and  $R_c$ ,  $G_c$ ,  $B_c$ . We also denote the dominant color ratios of original and compared images with  $CR_o$  and  $CR_c$ . A SVC can be calculated like in the following formula:

For ( $1 \leq i \leq SG_o$ )

For ( $1 \leq j \leq SG_c$ )

For ( $1 \leq m \leq DCSG_o[i]$ )

For ( $1 \leq n \leq DCSG_c[j]$ )

$$SVC [i][j][m][n] = ( |R_o - R_c|^2 + |G_o - G_c|^2 + |B_o - B_c|^2 )^{1/2}$$

$$SVC [i][j][m][n] = ( SVC [i][j][m][n] ) / thresholdSQR$$

$$SVC [i][j][m][n] = (1 - SVC [i][j][m][n]) \times (\min(CR_o, CR_c)) / 100$$

#### *Threshold Mechanism*

In order to prevent large comparison time and for a reasonable result a threshold mechanism is done during SVC calculation. The threshold mechanism checks the distance between dominant color components (R, G, B) of original and

compared images. In this study two threshold condition is worked out: Absolute threshold and Square threshold.

In absolute threshold condition, the mechanism works in the following way: If the distance between anyone of the dominant color components is larger than the threshold (i.e.  $|R_o - R_c| > \text{threshold}$ ) then SVC is not computed for that pair of dominant colors. In other words the SVC is computed only if all of the dominant color components are closer to each other than the threshold. This condition can be formulated like below:

$$|R_o - R_c| \leq \text{thresholdABS} \ \&\& \ |G_o - G_c| \leq \text{thresholdABS} \ \&\& \ |B_o - B_c| \leq \text{thresholdABS}$$

This condition ensures that each component close enough that two colors can be seen as similar. For example if R and G components of dominant colors are same (i.e.  $|R_o - R_c| = |G_o - G_c| = 0$ ) but B components differ by a value of 40. Actually these two colors are not seen as similar by human eye but unfortunately if absolute threshold mechanism did not processed, the SVC would be calculated and it would be a high value. This condition can be modeled as a cube with its center ( $R_o, G_o, B_o$ ).

In square threshold condition, the mechanism works in the following way: The distances are squared for each color component. Then they are summed. The square root of this sum is taken as the result. The SVC is calculated only if this result is smaller than the threshold. This can be formulated like below:

$$\text{ThresholdSQR} = \sqrt{3} \times \text{thresholdABS}$$

$$\text{Result} = (|R_o - R_c|^2 + |G_o - G_c|^2 + |B_o - B_c|^2)^{1/2} \leq \text{thresholdSQR}$$

This mechanism does not investigate the components of colors but the colors themselves. Other words not each component distance but altogether color distance is important. Thinking in 3-D, if the distance between colors is less than the threshold they are accepted as similar and SVC is calculated for them. This condition

ensures that colors are close enough that they can be seen as similar. This condition can be modeled as a sphere that has its center as  $(R_o, G_o, B_o)$ .

## 5.2. Color Histogram Search

Color Histogram Search uses the histogram information in RGB domain that is stored in database. Namely, each image is summarized with its R, G and B histograms in the database. For the sake of simplicity and efficiency, the histograms are quantized with 16. As a result there are 16 values for each histogram. Firstly, these values are read from the database for the original image. Secondly, for compared image these values are read. Now, as in the Dominant Color Search process a threshold mechanism is carried out. When the threshold condition is satisfied, the distance between images is calculated. This process is shown below:

$$A = \left[ \sum_{i=0}^{16} (HR_o[i] - HR_c[i])^2 \right]^{1/2} \quad B = \left[ \sum_{i=0}^{16} (HG_o[i] - HG_c[i])^2 \right]^{1/2}$$

$$C = \left[ \sum_{i=0}^{16} (HB_o[i] - HB_c[i])^2 \right]^{1/2}$$

if  $(A \leq \text{threshold} \ \&\& \ B \leq \text{threshold} \ \&\& \ C \leq \text{threshold})$

Distance =  $A + B + C$

Finally, these steps are carried out for all of the images in the database. These distance values are sorted in increasing order and the images corresponding to the smallest 10 distance values are shown as the result.

## 5.3. Color Label Search

Each image in the database is labeled. There are 11 possible labels. So, every image must have one of these labels. For Color label search, there are 216

choices of colors and 12 choices of labels. Here, the extra label stands for 'All Images'. When the Color label search is run, first the chosen color is decomposed to its R, G and B components. Then these values are compared with the values that are stored in the database and have the same label. Finally similar to previous searches the results are sorted and first 10 images are shown as the result. When the results are shown, the dominant colors of them displayed at the bottom of each one. These dominant colors can be also used for Color label search by clicking them and running Color label search again.

#### **5.4. Overview of The System**

The system is developed in a Windows 2000 operating system environment. As database Oracle 8i is used. Borland JBuilder 7.0 is used for programming tool. The system works on web browsers. This property supplies usage of the system all over the world. The system is composed of four parts: Web pages for browsing, a Web server, Class files and the Database. Web pages act as an interface for users. They listen the requests of users and send them to the web server. The web server runs on the database (server) side, takes requests from the browser and directs them to the class files. It also sends the results to the browser. The class files work on the requests, communicate with the database and give results to web server. The database stores the information about images, users, queries, etc. In Figure 1-1 this processes can be seen.

The system uses an XML file called 'Configuration.xml' to take information about the system itself. The servlet addresses, the database connection information and JSP file locations are stored in this XML file. This file is located at the file system directory of the web server. The usage of this file gives flexibility to the administrator of the web server. For instance if the location of a JSP file is changed, it is not required to change all links to that file, only a modification to this XML file is sufficient. Also, when the web server and database are installed on another computer the only thing to do is to change the corresponding line in this XML file.

### 5.4.1. Web Pages Implementation

The system has a lot of web pages. Some of them are login, query, upload and user management pages. For the sake of security the system starts with a login page comes with a username/password query. The usernames and passwords are stored in database and they are grouped in to two: Administrator and Normal users. If the user trying to login is the administrator, it is oriented to the administrator page. If the user is listed in the normal users, the main page is shown. If the user is not listed in one of these groups he/she cannot login to the system. This security check is carried out before every new request to the web browser. This step ensures unauthorized people not to use the system after an authorized person exits his/her session. After the login process some shortcuts are seen that are different for administrator and the normal users. According to the requests from users the web pages may pass through between other web pages. The possible web pages to visit are shown in Figure 5-1.

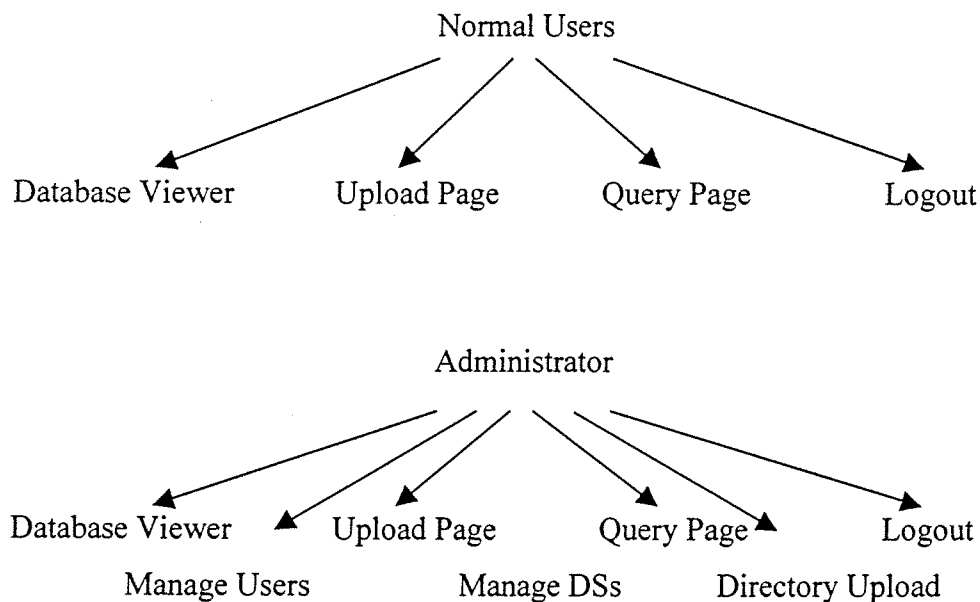


Figure 5-1 The Possible Web Pages for Different Users

The usages of these web pages are described below.

- a. Database Viewer: Lists the images and their related xml files that are stored in the database.
- b. Upload Page: Takes an image and its xml file, stores them in to database and parses the xml file in to the database.
- c. Query Page: First, takes a color with/without a label to search over the database. Then dominant color, color histogram or color label searches can be done.
- d. Logout: Ends the user session and opens the login page.
- e. Manage Users: Displays the information about users. Users can be deleted or their password can be given here.
- f. Manage DSs: Displays the information about Description Schemes in the database. DSs can be deleted or modified here.
- g. Directory Upload: Uploads all files in the temp directory of the system. Instead of uploading files one by one administrator can use this page.

#### 5.4.2. Web Server Implementation

In this thesis, the web server runs on the database (server) side, takes requests from the browser and directs them to the class files. It also sends the results to the browser. For this system Apache Tomcat web server (in standalone mode) is used. The reasons to use this web sever is analyzed in Chapter 2. Here we will investigate how it is used.

A directory called 'thesis' is created under the webapps directory. Then it is put in the server.xml file. Also the port is defined to be '80' in server.xml for the sake of simplicity when writing URLs. The modified lines in the server.xml file are:

```
<Context path="/thesis" docBase="thesis" debug="0" reloadable="true" />
<Connector className="org.apache.catalina.connector.http.HttpConnector"
port="80" />
```

The web server also needs the definitions of the environmental variables that are called JAVA\_HOME and CATALINA\_HOME. CATALINA\_HOME is the directory where the Tomcat and its subdirectories are stored. JAVA\_HOME is the directory where the Java (JDK) is stored.

The last configuration rule is about class and JSP locations. The class files must be located in the predefined directory and the JSP files must be located in the directory that is defined in server.xml file as 'docBase':

- All class files reside in: \$CATALINA\_HOME/webapps/thesis/WEB-INF/classes/
- All jsp files reside in: \$CATALINA\_HOME/webapps/thesis/

In normal operation, when the computer in server side is shut down then the web server should be restarted. For the sake of simplicity this is done automatically in this thesis. A service is constructed for this operation and this service restarts the Tomcat web server while the startup process of computer in server side.

### 5.4.3. Class Implementation

Almost all calculations, comparisons and processes over database are done by means of classes. There are totally 22 classes in the software of this system. The important classes are described below. Also the relations between these 22 classes are shown in Figure 5-2 at the end of this section.

- a. Configuration: Parses the configuration.xml file and sends the paths of class or JSP files when they are needed.
- b. DBConnectionPool2: Takes the database connection strings from the configuration class and sets a connection between the database and the system.
- c. IDGenerator: Generates the next value for primary keys of image, document, path, pathattribute, node, attribute, users, ds and cache tables of the database.

- d. `AuthenticationServlet`: Checks whether the user trying to login is a registered user. It also checks whether the user is administrator or not.
- e. `AVObjectXMLDownloadServlet`: Takes the name of image files then downloads them to the local machine from to the database.
- f. `XMLandObjectUploadServlet`: Takes the addresses of XML and image files then stores them in to the database.
- g. `XMLToDatabase`: After an XML file is stored in to the database this class parses that file and stores acquired information again in to the database.
- h. `AllObjectstoDatabase`: This class is for administrator use only. It takes all the files from the default directory, TEMP, and stores them in the database. It also calls `XMLToDatabase` class for all XML files to parse them in to the database.
- i. `ColorUtilities`: Makes conversions between RGB, YCrCb and HSB colorspaces. This provides comparisons between images having different colorspaces.
- j. `ColorQueryDB`: Reads histogram values of images from the database, sends them to `HistogramComparatorDB` class, takes the comparison results, sorts the results and responds to the browser by giving the first 10 results.
- k. `DomColorQueryDB`: Reads dominant color components of images from the database, sends them to comparator classes according to their parameters, takes the comparison results, sorts the results and responds to the browser by giving the first 10 results.
- l. `LabelQueryDB`: Takes R, G and B values of the selected color and also the label from the browser, sends them to `DomColorComparatorABSDB` class according to their parameters, takes the comparison results, sorts the results and responds to the browser by giving the first 10 results.
- m. `XMLandAVListingBean`: Lists all objects and their documents in the database.
- n. `UserManagementServlet`: Lists all approved and unapproved users with their usernames, names, surnames and e-mails in the database. Users can be approved or deleted by means of this class.

- o. DSManagementServlet: Shows all DSs included in the database. The paths of a DS can be displayed, modified or deleted. Also new paths and DSs can be added.
- p. HistogramComparatorDB: Takes two groups of histogram values which corresponds to different objects. Calculates the similarity between them and sends the comparison result back.
- q. DomColorComparatorSQRDB: Takes two groups of dominant color values which corresponds to different objects. Calculates the similarity between them by means of its square threshold mechanism and sends the comparison result back.
- r. DomColorComparatorABSDB: Works similar to DomColorComparator-SQRDB. But this class uses absolute threshold mechanism. Also this class can calculate similarities from labels with only one dominant color.

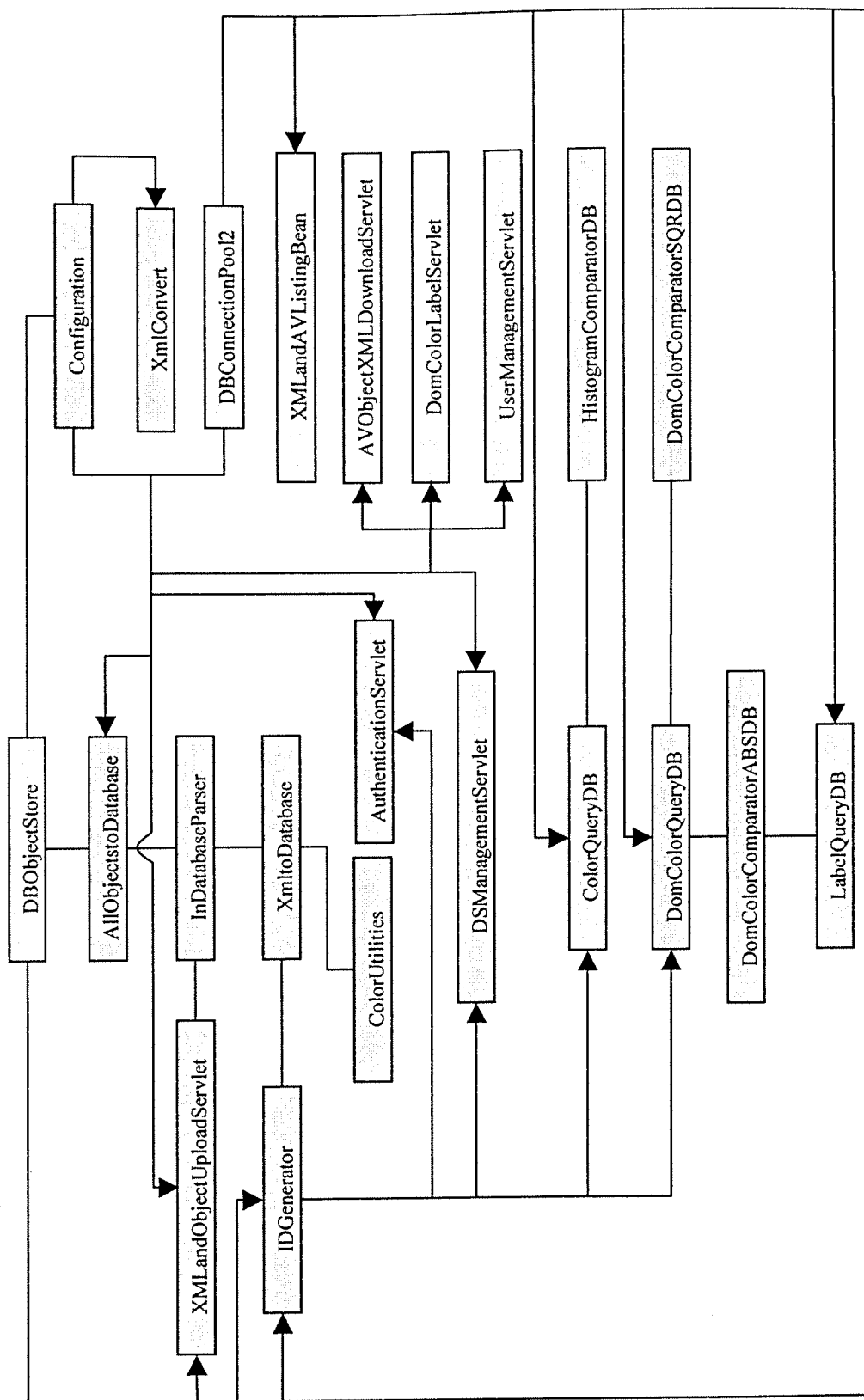


Figure 5-2 Relations Between All Classes in The Software of This System

#### 5.4.4. The Database Implementation

Oracle 8i (8.1.5) is used for the database part of this system. In the database the images and their text-based summaries are stored. But the database is designed in a way that in the future, users will be able to store other multimedia objects. So the term image is replaced with object from now on. There are 11 tables in the database. The table names and their usage purposes are explained in the following pages. Some rows of the corresponding table are presented in each attached figure.

a. **ATTRIBUTE:** For attributes of MPEG7 documents. The path, segment, document and node that an attribute belongs to are also stored.

ID	DOC_ID	SEGMENT_ID	PATH	ATTRIBUTE_ID	VALUE
279	140	1		0 3	
280	140	1		1 center1.gif	
281	141	1		0 3	
282	141	1		1 center2.gif	
283	142	1		0 3	
284	142	1		1 chang.gif	
285	143	1		0 3	
286	143	1		1 cheetah.jpg	
287	144	1		0 3	
288	144	1		1 circuit1.gif	
289	145	1		0 3	
290	145	1		1 cottage.jpg	
291	146	1		0 3	
292	146	1		1 devils-postpile-15.jpg	
293	147	1		0 3	
294	147	1		1 devils-postpile-16.jpg	

Figure 5-3 Sample rows from ATTRIBUTE Table of The Database

b. **AVOBJECT\_CONSTANTS:** For types of multimedia objects. Description of the object is stored.

ID	DESCRIPTION
1	image

Figure 5-4 Sample rows from AVOBJECT\_CONSTANTS Table of The Database

c. AVOBJECT: For multimedia objects. The name and type of the objects are also stored. A new object can be uploaded over the web. The administrator can upload a whole directory to the database.

ID	VALUE	TYPE	NAME	CONTENTTYPE
140	(BLOB)	1	center1.gif	image/gif
141	(BLOB)	1	center2.gif	image/gif
142	(BLOB)	1	chang.gif	image/gif
143	(BLOB)	1	cheetah.jpg	image/pjpeg
144	(BLOB)	1	circuit1.gif	image/gif
145	(BLOB)	1	cottage.jpg	image/pjpeg
146	(BLOB)	1	devils-postpile-15.jpg	image/pjpeg
147	(BLOB)	1	devils-postpile-16.jpg	image/pjpeg
148	(BLOB)	1	eagles-pt-muriefeld-03.jpg	image/pjpeg
149	(BLOB)	1	fer1.jpg	image/pjpeg
150	(BLOB)	1	fer10.jpg	image/pjpeg

Figure 5-5 Sample rows from AVOBJECT Table of The Database

d. CACHE: For the results of latest searches. Query ids are also stored. When a new object is stored this table is cleared.

ID	AV_ID	QUERIES_ID	AVOBJECTRESULTS
1	140	3	140 141 142 145 199 212 245 246 247 286
2	141	3	141 142 145 212 245 247 286 287 302 319
3	141	4	141 246 142 145 212 245 247 286 287 302
4	142	2	142 140 287 286 141 312 195 145 90 302
5	142	5	142 145 212 245 247 287 302 332 286 87
6	142	4	142 287 302 145 212 245 247 332 319 286
7	145	6	145 89 88 212 245 247 287 302 332 142
8	145	2	145 90 302 246 319 332 25 88 212 245
9	145	5	145 212 245 247 287 302 332 142 286 87
10	212	4	212 245 286 247 287 302 332 299 99 87

Figure 5-6 Sample rows from CACHE Table of The Database

e. DOCUMENT: For XML files of objects. Related multimedia object, XML file name and the parse status are also stored. A new XML file can be uploaded over the web. . The administrator can upload a whole directory to the database.

ID	AV_ID	VALUE	NAME	ISPARSED	CONTENTTYPE
140	140	(BLOB)	center1.xml	1	text/xml
141	141	(BLOB)	center2.xml	1	text/xml
142	142	(BLOB)	chang.xml	1	text/xml
143	143	(BLOB)	cheetah.xml	1	text/xml
144	144	(BLOB)	circuit1.xml	1	text/xml
145	145	(BLOB)	cottage.xml	1	text/xml
146	146	(BLOB)	devils-postpile-15.xml	1	text/xml
147	147	(BLOB)	devils-postpile-16.xml	1	text/xml
148	148	(BLOB)	eagles-pt-muriefeld-03.xml	1	text/xml
149	149	(BLOB)	fer1.xml	1	text/xml
150	150	(BLOB)	fer10.xml	1	text/xml

Figure 5-7 Sample rows from CACHE Table of The Database

f. DS: For Description Schemes. The name of DS is stored. A new DS can be added or an existing DS can be modified over the web.

ID	NAME
0	Mpeg7

Figure 5-8 Sample rows from DS Table of The Database

g. PATH: For all possible node paths of XML file. The DS that a path belongs to is also stored.

ID	VALUE	DS_ID
0	/Mpeg7/DescriptionUnit/Descriptor/ColorSpace	0
1	/Mpeg7/DescriptionUnit/Descriptor/Values/Percentage	0
2	/Mpeg7/DescriptionUnit/Descriptor/Values/ColorValueIndex	0
3	/Mpeg7/DescriptionUnit/Descriptor/Coefficients	0
4	/Mpeg7/DescriptionMetadata/Comment/FreeTextAnnotation	0

Figure 5-9 Sample rows from PATH Table of The Database

h. PATHATTRIBUTE: For all possible attribute paths of XML file. The DS that a pathattribute belongs to is also stored.

ID	ATTRNAME	VALUE	DS_ID
0	size	/Mpeg7/DescriptionUnit/Descriptor	0
1	input	/Mpeg7/DescriptionUnit/Descriptor	0

Figure 5-10 Sample rows from PATHATTRIBUTE Table of The Database

- i. NODE: For nodes of XML file. The path, segment and document that a node belongs to are also stored.

ID	DOC_ID	SEGMENT_ID	PATH_ID	VALUE_ID	VALUE
1530	140	1	0	0	RGB
1531	140	1	4	0	Building
1532	140	1	1	0	30
1533	140	1	1	1	25
1534	140	1	1	2	18
1535	140	1	2	0	152 128 152
1536	140	1	2	1	252 252 252
1537	140	1	2	2	204 212 152
1538	140	1	3	0	3.7104834E-5 0.0 0.0 0.15845619 0.0 0.0 0.185
1539	140	1	3	1	0.0 0.0 0.084642306 0.0 0.0 0.20191213 0.0 0.
1540	140	1	3	2	0.0 0.0 0.0 0.18264236 0.0 0.0 0.2103906 0.0
1541	141	1	0	0	RGB
1542	141	1	4	0	Building
1543	141	1	1	0	47
1544	141	1	1	1	27
1545	141	1	1	2	12
1546	141	1	2	0	100 128 100
1547	141	1	2	1	252 252 252
1548	141	1	2	2	204 168 100
1549	141	1	3	0	7.2291313E-4 0.0 0.0 0.19565356 0.0 0.0 0.239
1550	141	1	3	1	0.0 0.0 0.053246073 0.0 0.0 0.23064768 0.0 0.
1551	141	1	3	2	0.0 0.0 0.0 0.13647705 0.0 0.0 0.3236284 0.0

Figure 5-11 Sample rows from NODE Table of The Database

- j. QUERIES: For all possible query types in cache table. The query string is stored.

ID	QUERYSTRING	COMMENTS
2	ColorQuery.jsp?histSearch=av_id	Color Histogram Search
3	ColorQuery.jsp?domSearch=av_id&sqr=1&many=1	DominantColor Search with Square and Single
4	ColorQuery.jsp?domSearch=av_id&sqr=1&many=2	DominantColor Search with Square and Many
5	ColorQuery.jsp?domSearch=av_id&sqr=2&many=1	DominantColor Search with Absolute and Single
6	ColorQuery.jsp?domSearch=av_id&sqr=2&many=2	DominantColor Search with Absolute and Many

Figure 5-12 Sample rows from QUERIES Table of The Database

k. USERS: For users of the system. The name, surname, password and e-mail of the user are also stored.

ID	USERNAME	PASSWORD	NAME	SURNAME	E_MAIL
0	admin	tez	kani	guner	kkguner@yahoo.com
1	kkguner	kkg	kani	guner	kkguner@yahoo.com
2	gbozdagi	gba	gozde	bozdagi	g.bozdagi@ieee.org.tr
3	un_approv	dunayword	denemel	soyad_denemel	mail@denemel.com.tr
4	deneme2	dunayword	dene	soyad_deneme2	mail@deneme2.org

Figure 5-13 Sample rows from USERS Table of The Database

The relations between these 11 tables are shown in Figure 5-14.

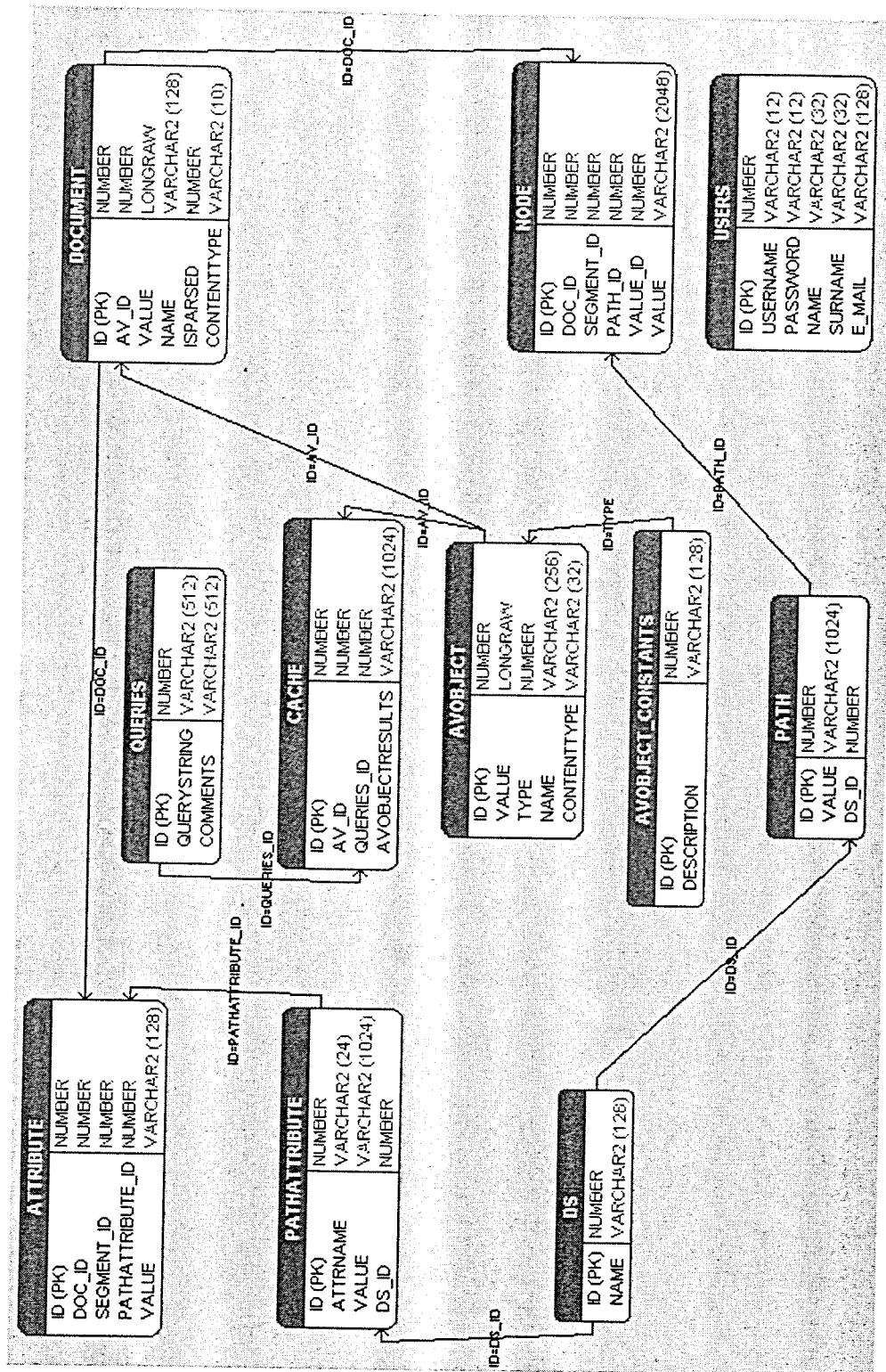


Figure 5-14 Relations Between All Tables in The Database of This System

## CHAPTER 6

### EXPERIMENTAL RESULTS AND CONCLUSION


Throughout this study, an Mpeg-7 compliant and ORDBMS based CBIR system is worked out. The Oracle database is used in order to store images and XML files. This approach supplies system security. There is also a session tracking mechanism for web pages, which prevent misuse of the system. 'Four-column indexing' is done over the database to achieve fast queries. Since all the system components can work on other platforms the system is portable. Finally, MPEG-7 standard Ds and DSs are used to be consistent with related systems.

The retrieval process of the system is based on six different search types. Five of them work as query by example and remaining one (ColorLabelQuery) works as label query with only one dominant color. In this chapter, five different search results are compared. The images are thought as collections of segments. So, some segments of images are inserted to other images and comparisons are done again. The successful aspects of each search technique are explained.

A similar system is constructed, which is based on XML but not on a database, to make comparisons between performances. Same amount of objects are stored in both systems. In average, 100 images with their 100 summary documents are stored in to the database in 10.359 seconds. But the comparisons are done for retrievals because storage process is done only once. The comparisons are made for 100, 200, 300, 400 and 500 images in turn. Same image is used for all tests. A

computer with AMD 1700+ CPU, 256 MB RAM and 20 GB 7200 rpm is used for the test. The test results and their graphical representations will be displayed.

At the end of this chapter some developments for this system that can be done in the future are described and the reasons and results for them are also stated.

The system starts with ColorLabelQuery, which takes a label and only one dominant color to search over the database. There are 216 choices of dominant colors to choose. There is also a label 'ALL IMAGES' to search over not a specific label but all labels in the database. In Figure 6-1, the search result for  $(R, G, B) = (102, 255, 51)$  and label = 'ALL IMAGES' is shown. This query is done in 3.281 seconds over 338 objects. The sample searches that are shown in this chapter are done at the server side. The appearance of color with  $(R, G, B) = (102, 255, 51)$  is: 

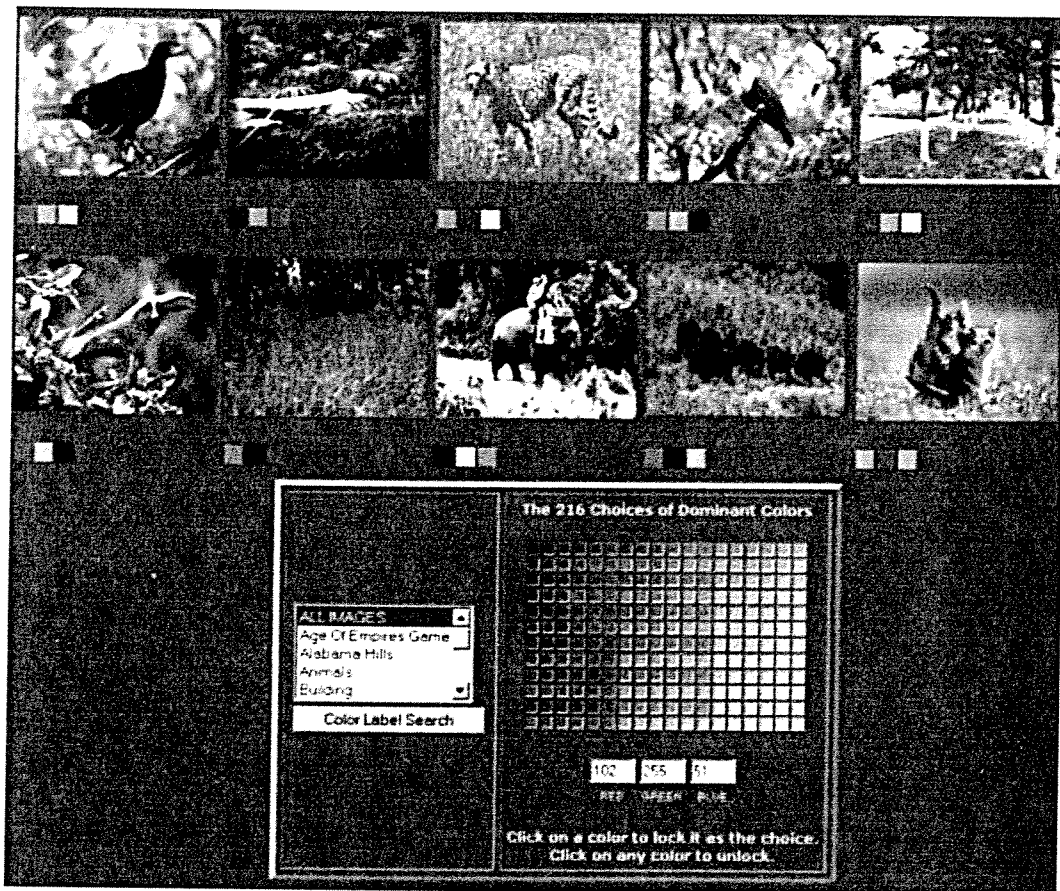


Figure 6-1 Search results for  $(R, G, B) = (102, 255, 51)$  and label='ALL IMAGES'

The labels seen on this page are taken from the database and the 216 choices of colors are defined in ColorLabelQuery page. After the ColorLabelQuery results are seen, it can be run again with another label and dominant color or other search techniques can be thought alternatively. When the image to be searched by other five searches is clicked by mouse, a menu showing alternative searches is seen.

For example, if Color Histogram Search is chosen to search about image 'untitled3.jpg', the results are displayed in Figure 6-2. This search does not deal with the dominant colors of images but with distribution of all colors over image. The first image in this figure is 'untitled3.jpg' itself. Since distribution of the histogram of 'untitled3.jpg' is near to colors white and dark green, the histograms of results are also similar.



Figure 6-2 ColorHistogram search results for 'untitled3.jpg'

In general people do not deal with the distribution of R, G, and B components of colors throughout the image. Instead, they want to find images having similar colors without concerning distribution of them. If dominant color search in single and square mode is run for the same image 'untitled3.jpg', the results are not very similar as seen in Figure 6-3. This is because dominant color searches deal with only dominant colors but not the whole image.

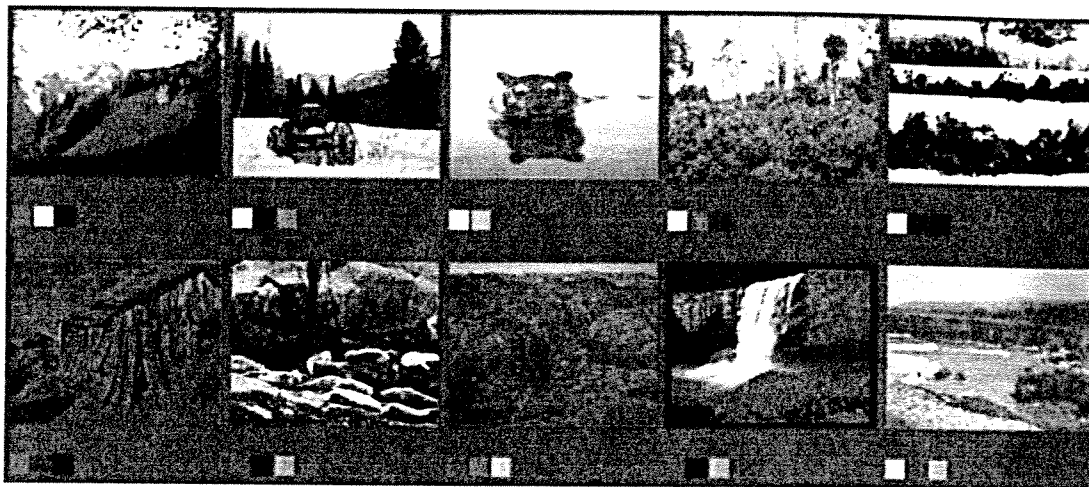


Figure 6-3 Square and single mode dominant color search results for 'untitled3.jpg'

The difference between single and joint mode of dominant color search can be realized with the search about image 'bodie7.jpg'. In Figure 6-4 and Figure 6-5 search results of single and joint mode are shown. The ninth image in Figure 6-4 becomes third image in Figure 6-5. Because the first dominant colors of first and third images in Figure 6-4 have large ratios and they are very similar. Although the second and third dominant colors are not very similar, the similarities between them are neglected. However in joint mode, since the sum of similarities is considered, the third image in Figure 6-5 becomes more similar than others.



Figure 6-4 Square and single mode dominant color search results for 'bodie7.jpg'

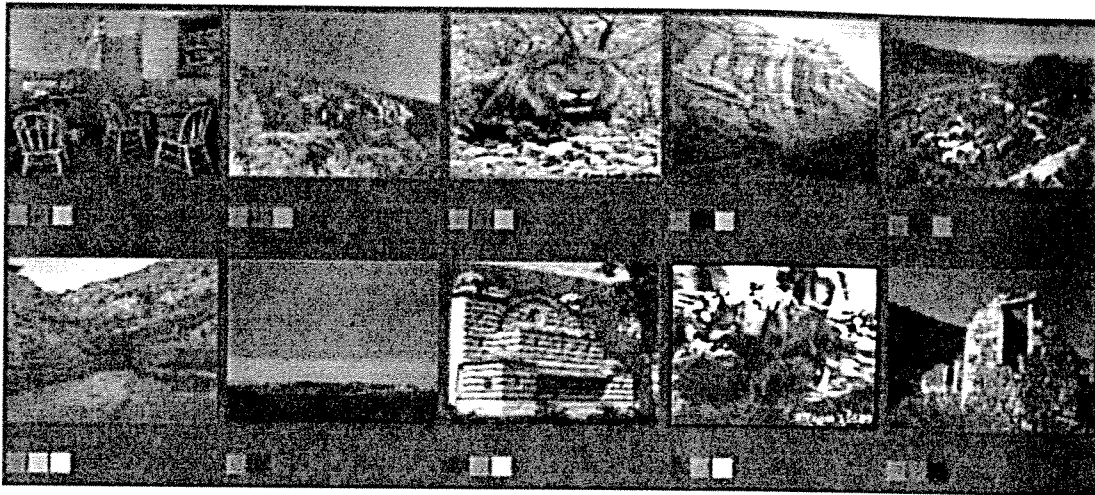


Figure 6-5 Square and joint mode dominant color search results for 'bodie7.jpg'

The variation between square and absolute mode of dominant color search can be seen with the joint mode search about image 'pic3.gif'. In Figure 6-6 and Figure 6-7, search results of square and absolute mode are shown respectively. The second image in Figure 6-6 does not exist in Figure 6-7. Because first dominant color of this image occupies a large area (around 70%) in image and unfortunately its distance of R component to R components of dominant colors of 'pic3.gif' is slightly larger than the threshold. As a result, absolute threshold mechanism does not calculate similarity values for the first dominant color of the second image and it is not displayed in the results. But since square threshold mechanism looks for the total distance of colors but not the components it calculates the similarity value and because of the area of first dominant color it is a high value. Consequently, the second image in Figure 6-6 becomes more similar than others.

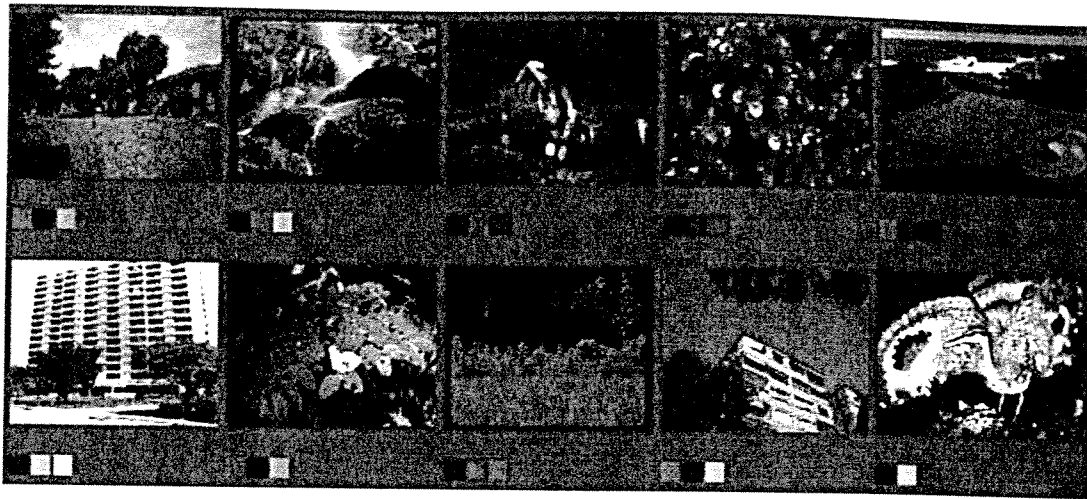


Figure 6-6 Square and joint mode dominant color search results for 'pic3.gif'

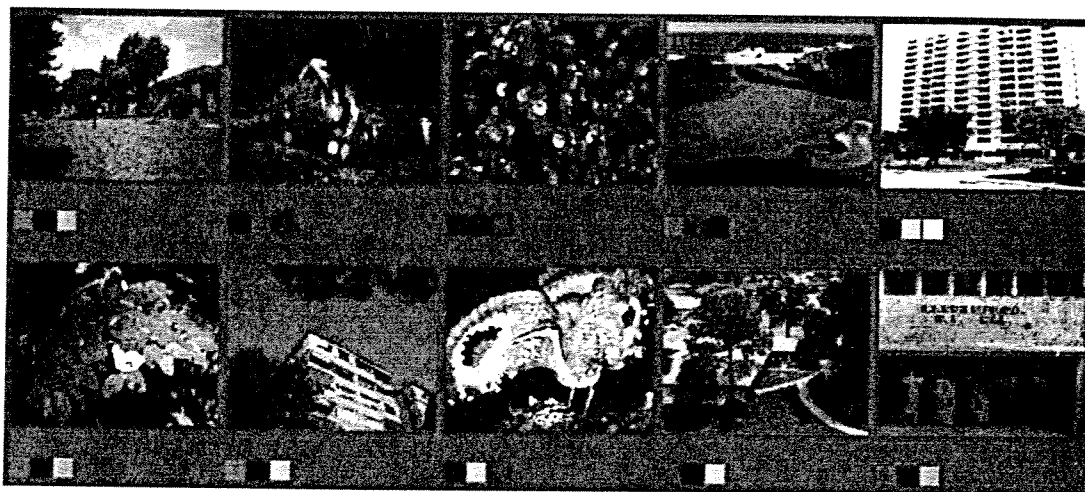


Figure 6-7 Absolute and joint mode dominant color search results for 'pic3.gif'

The importance of threshold mechanism can be understood if the threshold is increased and the above search is repeated. Now, the component distances between the first dominant color of second image and dominant colors of 'pic3.gif' is smaller than the threshold. As a result, absolute threshold mechanism calculates similarity values for the first dominant color of the second image and finds high values since its color ratio is very high. So, the second image in Figure 6-8 also exists in Figure 6-9.

Since some neglected similarities between colors are now calculated, this affects similarity values between images. So, some images in both search results are also changed.

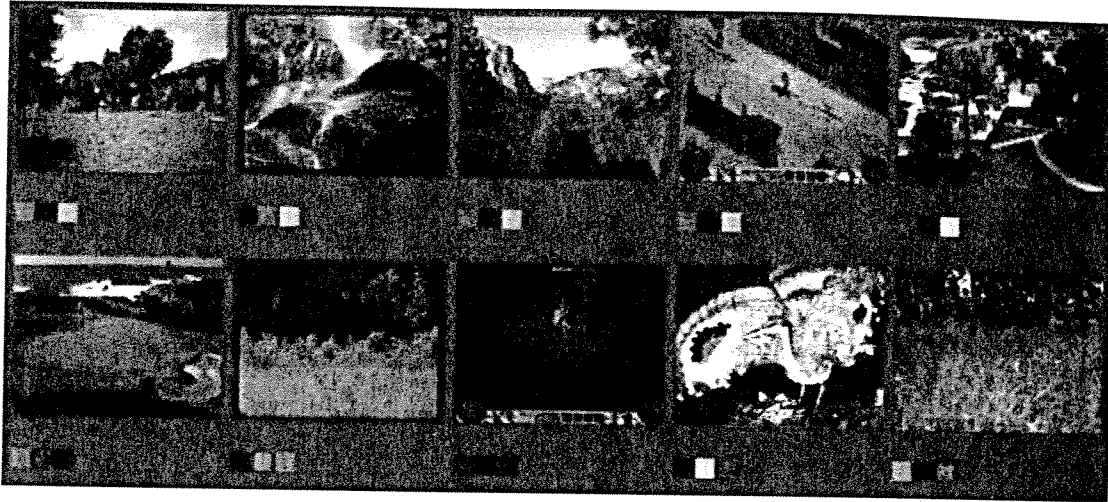


Figure 6-8 Square and joint mode dominant color search results for 'pic3.gif' when threshold is increased

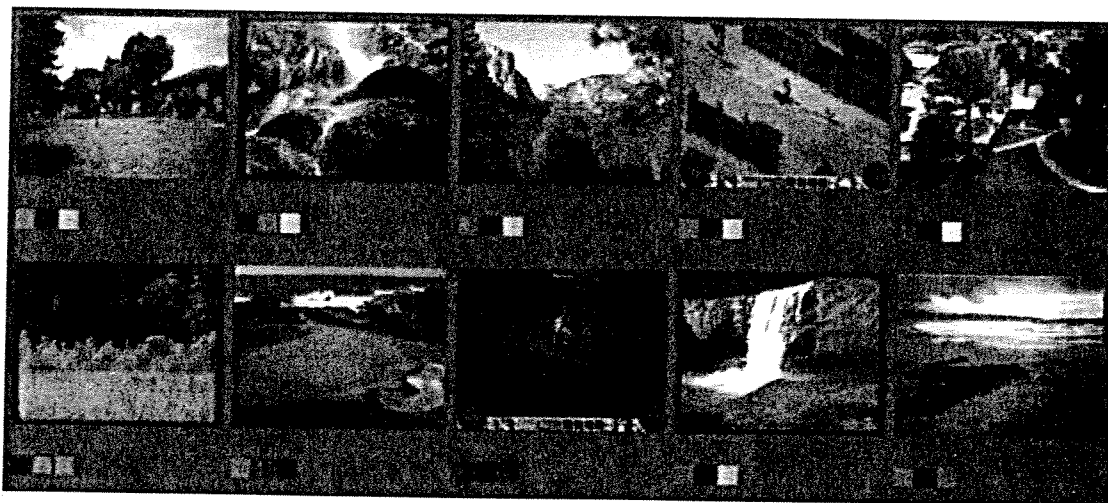


Figure 6-9 Absolute and joint mode dominant color search results for 'pic3.gif' when threshold is increased

The effect of object segmentation can be realized when the results in Figure 6-10 and Figure 6-11 are compared. In the results of first search 'fer9.jpg' is seen as the 10<sup>th</sup> image. If 'fer9.jpg' is inserted in to the right-bottom part of 'pic2.gif' as a segment and renamed as 'pic2new.gif', 'fer9.jpg' becomes the most similar image to 'pic2new.gif'. Because 'fer9.jpg', 'pic2new.gif' similarities to 'pic2new.gif' are equal. But since 'fer9.jpg' has a smaller id in database it is in the first position.



Figure 6-10 Absolute and joint mode dominant color search results for 'pic2.gif'

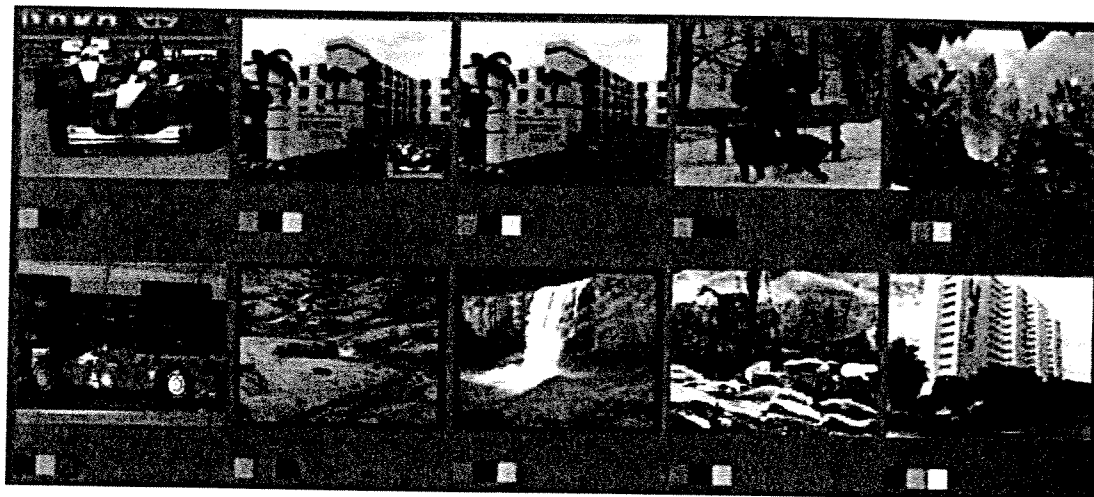


Figure 6-11 Absolute and joint mode dominant color search results for 'pic2new.gif'

The typical query times of database system and XML based system are compared at a step of 100, starting from 100 images up to 500 images. All query tests are done at the server side for the same image between the same groups of images to provide accuracy. Dominant color searches are used for these tests. The test results are shown in Figure 6-12.

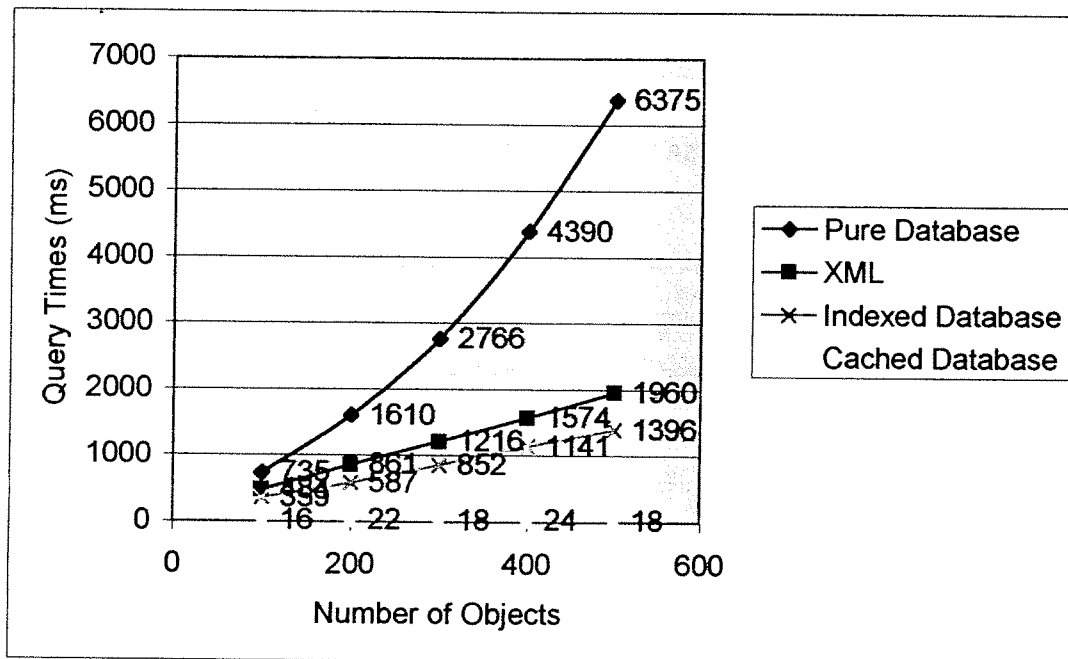


Figure 6-12 Comparison of Query Times

When object summaries are stored but no indexing is done, the database is denoted by 'Pure Database'. This state of the database is not faster than XML-based system. Also it is getting slower when new objects are stored. 'Indexed Database' stands for the database after indexing technique is applied. The indexing process is done when a new object is uploaded. So it takes a little more time to upload an object (i.e. 100 images with their 100 summary documents are stored in to the database in 11.284 seconds). But since query time is the important criteria, it is faster than 'Pure Database' and XML-based system. 'Cached Database' denotes the database when the CACHE table is activated. The cached queries have the best performances since they

do not require any comparisons between objects. They have results of queries before the queries and quickly give response to requests. There are at most five rows in the cache table for each object in the database corresponding to five different search techniques. So if  $N_o$  denotes the number of objects, there are at most  $5.N_o$  rows in the cache table. But the increase in  $N_o$  does not affect the query time very much as shown in Figure 6-12. The reason of this situation can be stated as: For queries selecting the same number of rows from an indexed table (without considering total number of rows), the response times are same. Actually this situation stands for 'Indexed Database' too, but since comparisons are done between objects in its queries its query times increase a little with increasing  $N_o$ .

The loading times of totally 10 images are calculated at the client side with an ISDN 128K internet connection in both server and client sides. They are around 1300 ms for database and around 700 ms for XML based system.

### ***Future Work***

The system built during this study can be improved with a segmentation tool that allows users to select their own segments. The segmentation process may be done automatically too.

The users can be given the right to select their choice between uploading objects with or without XML files. This can be done when the system has its own feature extraction procedure.

The threshold value can be modified automatically for the object during the query process. This property can supply more accurate results.

The path management of the database can be provided partially to all users of the system. So they can construct their own schemas and add, delete or modify their paths in these schemas. Perhaps they can make their queries according to their schemas.

## REFERENCES

- [1] J.Huang, Color-Spatial Image Indexing and Applications, Ph.D. Thesis, Cornell University, August 1998.
- [2] E. Oja, J. Laaksonen, M. Koskela, "Self-Organizing Maps for Content Based Image Retrieval", Proceedings of IJCNN'99, Washington, DC, July 1999.
- [3] E. Oja, J. Laaksonen, M. Koskela, "Comparison Of Techniques For Content-Based Image Retrieval", Proceedings of SCIA 2001, Norway, June 2001.
- [4] J. Jaworski, Java Developer's Guide, SAMS, CD-ROM Edition, August 1997.
- [5] M. Morrison, R. Fanta, Java Unleashed, SAMS, 2nd Edition, December 1996.
- [6] C. Walnum, Java By Example, MacMillan Computer Publication, CD-ROM Edition, June 1996.
- [7] M. Cohn, B. Morgan, M. Morrison, M. T. Nygard, Java Developer's Reference, SAMS, November 1996.
- [8] M. Perry, "Components for Web Applications", DSD Dept. Meeting, Ernest Orlando Lawrence Berkeley National Laboratory, August 2001.
- [9] J. Goodwill, B. Morgan, Developing Java Servlets, Pearson Education, 2nd Edition, May 2001.
- [10] C. Wang, "Content-Based Image Retrieval System", A Report Submitted to Dr. Raghavan, Louisiana University, 2001.
- [11] J. Xiaoping, Lecture Notes of Object-Oriented Enterprise Application Development Course, DePaul University, 2002.
- [12] MPEG Requirements Group, "MPEG-7 Requirements Document V.10", Doc. ISO/IEC TC1/SC29/WG11/N2996, Melbourne (Australia), October 1999.
- [13] J. M. Martinez, "Overview of the MPEG-7 standard (version 5)", ISO/IEC JTC1/SC29/WG11/N4031, Singapore, March 2001.

- [14] S. Paek, A.B. Benitez, S. Chang, "Self-Describing Schemes for Interoperable MPEG-7 Multimedia Content Descriptions", Columbia University, 1999.
- [15] J.R. Ohm, F. Bunjamin, W. Liebsch, B. Makai, K. Müller, A. Smolic, "MPEG-7 Description Scheme for Visual Objects", WIAMIS'99, Berlin, May 1999.
- [16] O. Steiger, A. Cavallaro, "MPEG-7 Description for Scalable Video Reconstruction", submitted to the SPIE/IS&T Journal of Electronic Imaging, 2002.
- [17] J. M. Martinez, "MPEG-7 Overview (version 8)", ISO/IEC JTC1/SC29/WG11, Klagenfurt, July 2002.
- [18] MPEG-7 DDL FAQ, <http://archive.dstc.edu.au/mpeg7-ddl>
- [19] T. Bray, J. Paoli, "Extensible Markup Language (XML) 1.0 (Second Edition)", October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>
- [20] XML in 10 points, <http://www.w3.org/XML/1999/XML-in-10-points>
- [21] T. Gorman, "20 Questions on XML", <http://builder.com/5100-31-5076848.html>
- [22] N. Walsh, "What is XML?", <http://www.xml.com/pub/a/98/10/guide1.html>
- [23] S. Rajan, "JSP and XML",  
[http://www.geocities.com/sundar\\_rajani/java/jsp/jspxml1.html](http://www.geocities.com/sundar_rajani/java/jsp/jspxml1.html)
- [24] What is a DTD?, <http://webdesign.about.com/library/week/aa101700a.htm>
- [25] R. L. Costello, "XML Schema Tutorial - XML Technologies Course",  
<http://www.xfront.com/xml-schema.html>
- [26] M.E.Dönderler, Data Modeling And Querying For Video Databases, Ph.D. Thesis, Bilkent University, July 2002.
- [27] D. McG. Squire, "Learning a similarity-based distance measure for image database organization from human partitioning of an image set", WACV'98, Princeton, NJ, October 1998.
- [28] U. Shaft, R. Ramakrishnan, "Data Modeling and Querying in the PIQ Image DBMS", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 19(4):28-36, 1996.
- [29] N.S. Grant, "DISIMA: A Distributed Image Database Management System", Final Report – Nserc Strategic Grant, University of Alberta, April 2000.
- [30] S. Straiger, T. Bumbalough, Developing Personal Oracle, SAMS, 2nd Edition, 1997.

- [31] Oracle Overview, <http://members.tripod.com/mdameryk/OrclOverview.htm>
- [32] [http://www.idevelopment.info/html/ORACLE\\_overview.shtml](http://www.idevelopment.info/html/ORACLE_overview.shtml)
- [33] Oracle Corporation, "Slides of Introduction to Oracle Course", 1999
- [34] [http://spdoc.cineca.it/mysql/mysql-ora/mysql\\_compare.html](http://spdoc.cineca.it/mysql/mysql-ora/mysql_compare.html)
- [35] Oracle Corporation, "Oracle8i Concepts Release 8.1.5", <http://docs.oracle.com>
- [36] Timuçin Noyan, XML Based Image Retrieval System, M.S. Thesis, METU, September 2000.
- [37] Medeni Soysal, Combining Image Features For Semantic Descriptions, M.S. Thesis, METU, September 2003.