XML VERİTABANLARINDA KELİME VE KESME DİZİNLEMELERİNİ KARŞILAŞTIRMA

COMPARISON OF TEXT AND CUT INDEXING IN XML DATABASES

YÜCE PİŞKİNSÜT

Başkent Üniversitesi
Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin
BİLGİSAYAR Mühendisliği Anabilim Dalı İçin Öngördüğü
YÜKSEK LİSANS TEZİ
olarak hazırlanmıştır.

2007

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma, jürimiz tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan :….....
Öğr. Gör. Güven KÖSE

Üye :….....
Prof. Dr. Hayri SEVER

Üye : ….....
Doç. Dr. Erdoğan DOĞDU

ONAY

Bu tez ...../...../..... tarihinde Enstitü Yönetim Kurulunca belirlenen yukarıdaki jüri üyeleri tarafından kabul edilmiştir.

...../...../.....
Prof. Dr. Emin AKATA
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde beni yönlendiren, tez danışmanım Öğr. Gör. Güven Köse'ye, yeni yaklaşımlar geliştirme konusunda sürekli fikir ve kaynak sağlayan, Prof. Dr. Hayri Sever'e, çalışmalarımda desteklerini esirgemeyen Prof. Dr. Ümit Karakaş'a, Doç. Dr. Erdoğan Doğdu'ya, bana hep destek olan sevgili annem Sema Pişkinsüt'e, babam Feyyaz Pişkinsüt'e, yardımcı olan kardeşlerim Ece ve Gülce Pişkinsüt'e, anneannem Kadriye Tutar'a, manevi olarak hep yanımda olan dedem Mustafa Tutar'a teşekkür ederim.

ÖZ

XML VERİTABANLARINDA KELİME VE KESME DİZİNLEMELERİNİ KARŞILAŞTIRMA

YÜCE PİŞKİNSÜT

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Günümüzde yaygın bir şekilde kullanılmaya başlanılan XML, sadece web uygulamalarında değil kendine özgü özellikler ve fonksiyonlar içeren veritabanları sayesinde popüler bir işaretleme dili haline gelmiştir. Bununla birlikte içerik yönetiminde (CM) belgelerin veritabanlarında saklanılmasında da kullanılmaya başlanılmıştır. Bugün Doküman Yönetim Sistemlerinin (DYS) belge saklama formatında XML kullanılmaktadır. Teknolojik imkanlar geliştikçe belgelerin yapılarında ve içeriklerininde de değişiklik meydana gelmektedir. İçerik bakımından gelişen belgelerin veritabanlarında saklanması ve sorgulanmasında kullanılan yöntemlerde de gelişmeler meydana gelmektedir. Bu tez çalışmasında, XML'in genel kullanımı hakkında detaylı bilgi edinilmiş, XML belgelerinin veritabanlarında saklama ve sorgulamanın performansını iyileştirilmesinde kullanılan dizinleme yötemleri incelenmiştir. Bununla birlikte yazılan bir arayüz sayesinde aynı içeriğe sahip farklı yöntem ile saklanan XML belgelerinin kelime dizinleme ve kesme dizinleme kullanılarak sorgulama performansları ortaya çıkarılmaya çalışılmıştır.

Bu tez çalışmasında amaç, içerik bakımından gelişen XML dökümanlarının sayıca fazlalaşması durumunda veritabanlarında kullanılan sorgulama performansının artırması hedeflenen yeni dizinleme yöntemleri için alt yapı oluşturmaktır. Bu nedenle dizinleme yöntemleri üzerine odaklanılmıştır. Gelecekte XML belgelerinin sorgulama performanslarını artırarak veritabanlarında daha fazla sayıda ve geniş içerikte XML belgelerinin saklanabilmesi hedeflenmiştir.

ANAHTAR SÖZCÜKLER: XML (Genişletilebilir Metin Dili), XML Database (XML'e Özgü Veritabanı), XMLTYPE , XQuery, Dizinleme

ABSTRACT

COMPARISON OF TEXT AND CUT INDEXING IN XML DATABASE

YÜCE PİŞKİNSÜT

Başkent University Institute of Science

The Department Of Computer Engineering

Nowadays, XML has become one of the most popular mark up language not only by using web applications but also it contains databases which includes special features and functions. In addition to this, it has been started to use in CM for the aim of saving all documents in databases. Today XML is using as the format of document, for storing at database in Document Management System (DMS). As the technologic opportunities develop, some changes occur in the structure and contents of documents. Also it can be seen that there are some developments in methods which are used in storing and searching the documents in databases. In this thesis work, information about general usage of the XML is obtained , indexing methods for upgrading search and store XML data in databases. In addition to this, text and cut indexing have compared to stored same content of document in database with interface.

The goal of this thesis work, to show the new indexing methods when the number of XML databases with rich content is increased, then to reach the aim of incerased searching performance. Therefore, this work is focused on indexing methods. In future, the basic aim is to store XML documents which includes lots of number and large of contents via increasing the performance of XML documents.

KEY WORDS: XML, XML Database, XMLTYPE , XQuery, Indexing Methods

İÇİNDEKİLER LİSTESİ

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

KISALTMALAR

ANSI : American National Standart Institute (Amerikan Ulusal Standart Enstitüsü)

API : Application Programming Interface (Uygulama Programlama Arayüzü)

ASCII : American Standard Code for Information Interchange

SGML : Standardized Generalized Markup Language (Standartlaştırılmış Genelleştirilmiş İşaretleme Dili)

ISO : The International Organization for Standardization (Uluslararası Standartlaştırma Organizasyonu)

XML : Extensible Markup Language (Genişletilebilir İşaretleme Dili)

GML : Generalized Markup Language (Genelleştirilmiş İşaretleme Dili)

W3C : World Wide Consontium (Dünya Çapında Konsorsiyum)

DTD : Document Type Definition (Belge Tipi Tanımlama)

WML : Wireless Markup Language (Kablosuz İşaretleme Dili)

WAP : Wireless Application Protocol(Kablosuz Uygulama Protokolü)

SAX : The Simple API for XML (Genişletilebilir İşaretleme Dili için Basit Uygulama Programlama Arayüzü)

URI : Uniform Resource Identifier (Tek Kaynak Tanımlayıcısı)

DB : Database (Veritabanı)

HTML : Hyper Text Markup Language (Hiper Metin İşaretleme Dili)

DOM : Document Object Model (Doküman Nesne Modeli)

FTP : File Transfer Protocol (Dosya Aktarma Protokolü)

JDBC : Java Database Connectivity (java veritabanı bağlanırlığı)

CLOB : Character Large Object

PL : Programming Language

SQL : Structured Query Language

HTTP : Hyper Text Transfer Protocol (Yüksek Metin İletişim Protokolü)

B2B : Business to Business (İşletmeden İşletmeye)

LOB : Large Object

BÖLÜM 1

GİRİŞ

1.1 Problemin Tanımı

XML hakkında geniş bilgiye sahip olabilmemiz için öncelikle İşaretleme Dilinin (Markup Language) ne olduğunu açıklamamız gerekir. Yazılı metinlerin baskı makinelerinde ne şekilde yayınlanacaklarını belirlemek için kullanılan semboller ve notlar "işaretleme (markup)" olarak ifade edilebilir. Basılı yayının gelişmesi ile birlikte yayıncılar yazılı metinlerde bu ifadeleri kullanmaya başlamışlardır. Bu ifadelerin kullanılmasındaki amaç, yazılı metnin belirlenen kısımlarına özel bir anlam kazandırmaktır. Bu amaçla kullanılan işaretler, kurallar ve gramer kümesi "İşaretleme Dili" olarak tanımlanır. Benzer işaretleme yapılarına birçok alanda rastlanabilmektedir. Örnek verecek olursak, ASCII kodlama standardının içerdiği birçok kontrol karakteri ve işaretlemeler veri iletişimi için kullanılmaktadır. Kelime işlem programları, metin kısmına gömülmüş olan birçok işareti barındırırlar. Bu işaretler sayesinde metnin yazı tipi, biçimi ve stilleri belirlenir, farklı kısımlar birbirlerinden ayrılır. Ayrıca sembol ve işaretler; programlama dillerinde, veri yapılarını ve verileri ayırmak için de kullanılmaktadır. Taşınabilir ve paylaşılabilir bir uygulama geliştirmek için standart ayıraçların işaret veya etiketler kümesi ile birlikte kullanılması gerekmektedir.

1960'lı yılların sonlarına doğru IBM tarafından yapılan araştırmalar ve çalışmalar sonucunda, ilk işaretleme dili olan GML (Generalized Markup Language) geliştirilmiş ve bu sayede metin ve belgelerin kolay bir şekilde taşınabilmesi, paylaşılabilmesi ve işlenebilmesi sağlanmıştır. GML, ANSI (American National Stardard Institute) tarafından oluşturulan bir grup ile 1978 yılında geliştirilmiş ve bu geliştirilen yeni dile SGML (Standardized Generalized Markup Language) adı verilmiştir. 1986 yılında SGML, ISO (the International Organization for Standardization) kurumunca uluslararası bir standart olarak kabul edilmiştir [1]. SGML bir metin veya belge kümesinde kullanılan dilin gramer ve sözlük yapısını belirtmek amacıyla kullanılan bir işaretleme dildir. Amerikan hükümeti

kuruluşlarında, havacılık, otomotiv ve benzeri büyük endüstri kuruluşları ile yazılı basın endüstrisinde bir belgeleme standardı olarak SGML kullanılmaktadır. SGML diğer dilleri göre daha karmaşık bir yapıya sahiptir. Ayrıca uygulama geliştirme maliyetinin de yüksek olduğu bilinmektedir. Bu iki olumsuz özelliği sebebi ile SGML dili yaygın olarak kullanılamamıştır. Buna rağmen SGML dili gerek yapısı gerekse sayfa geliştiricisine sağlamış olduğu yüksek kontrol olanağı ile çok güçlü bir işaretleme dili olmuştur [1].

1989 yılında Tim Berners-Lee ve Anders Berlung tarafından, internet ortamında belgelerin paylaşılabilirliğini arttırmak ve kolaylaştırmak amacı ile günümüz WEB uygulamalarının temel öğesi olan HTML (Hypertext Markup Language) dili geliştirilmiştir. SGML, meta dil [1] özelliğine sahiptir. HTML dili, SGML dilinden türetilmesine rağmen meta dil özelliğine sahip değildir. Yani HTML dilinden yeni diller türetilememektedir. HTML dilinin başarılı olmasının ana sebebi çok basit ve anlaşılabilir bir yapıda olmasıdır. HTML dili bir belgenin içerdiği başlık, yazı biçimi, resim ve tablo gibi bilgileri bilgisayar ortamında standart bir şekilde görüntülemek ve biçimlendirmek için geliştirilmiş bir dildir. Belgenin istenen biçimde sunulması etiket (tag) olarak ifade edilen işaretler ile sağlanmaktadır. Belgelerin standart bir biçimde görüntülenerek sunulabilmesi için HTML dili geliştirilmiştir. Fakat HTML dilinin sadece WEB tarayıcıları için sunum ve görüntüleme amaçlı olması ve gelişen WEB uygulamaları için gerekli olan birçok özelliği kendinde barındıramaması nedeni ile XML dili geliştirilmiştir.

Word Wide Consortium (W3C, http://www.w3.org) 1996 yılında SGML'nin gücünü ve esnekliğini içeren fakat daha basit ve anlaşılabilir bir işaretleme dili geliştirmek amacıyla XML dilini tasarlamaya başlamıştır. XML 1.0 bir standart olarak W3C tarafından Şubat 1998'de yayınlanmıştır. XML dili, SGML dilinin bir alt kümesidir. XML de SGML gibi bir meta dildir. Bir başka ifade ile farklı dillerin yapısını tanımlamak için kullanılan bir dil özelliğine sahiptir.

[1] Meta dil, kendisinden yeni diller türetilebilen bir dil olma özelliğidir.

Günümüzde popüler olarak kullanılan XML, bilgisayarlar arası belge paylaşımını kolaylaştırmıştır. Gelişmiş WEB uygulamalarında kendinde barındırdığı birçok özellik sayesinde kullanımı yaygınlaşmıştır. Gerek üretici firmalar gerekse yazılım geliştirenler tarafından kolaylıkla benimsenmiş ve teknolojik gelişmeler sayesinde günümüzün vazgeçilmez işaretleme dili haline gelmiştir. İlerleyen zaman içerisinde XML'in sadece WEB uygulamalarında değil, geniş tabanlı içerik yönetiminde de söz sahibi olması beklenmektedir. Bu bağlamda, tez çalışmasının konusu olarak XML belgelerinin XML'e özgü veritabanlarında saklanma biçimleri ve sorgulama performanslarının arttırılması için uygulanacak dizinleme yöntemleri seçilmiştir.

1.2 Tez Çalışma Planı

Bu Yüksek Lisans Tezine XML dilinin yapısı, özellikleri ve kullanım şekilleri araştırılarak başlanılmıştır. Daha sonrasında XML belgelerinin gerek ilişkisel veritabanında gerekse XML'e özgü veritabanlarında saklama biçimleri ile kullanılan veritabanı yapıları araştırılmış, özellikleri incelenmiştir. XML belgelerinin sorgulama performanslarının araştırılması için bir uygulama yazılmış bu uygulama ile farklı alanlarda ve farklı yöntemlerle saklanan belgeleri üzerinde çeşitli dizinlemeler kullanılarak ulaşılan sorgulama performansları karşılaştırılmıştır. Son olarak da gelecekte yapılması düşünülen çalışmalar belirtilmiştir.

1.3 Tez Düzeni

Bu tez çalışmasının birinci bölümünde, tez içeriğinin temel özellikleri, tez konusu ve çalışma planı hakkında temel anlamda giriş bilgileri verilmiş, tezin genel çerçevesi oluşturulmuş, XML'e temel olan işaretleme dilleri ve tarihçesi hakkında bilgiler verilmiş, işaretleme dillerinin özellikleri belirtilmiştir.

İkinci bölümde, XML'in ne olduğu, kullanım şekilleri, genel özellikleri, yararları, genel söz dizilimi ve ilgili teknolojik biçimler kapsamlı olarak araştırılmıştır.

Üçüncü bölümde, XML belgelerinin tutulduğu veritabanlarından Oracle 9i XML veritabanı mimarisi, bu veritabanının genel kullanım bilgileri ile klasik ilişkisel veritabanlarında kullanılan sorgulama dili olan SQL ile XML veritabanı sorgu dili olan XQuery arasındaki benzerlik, farklılık ve dönüşümler incelenerek, geniş ve kapsamlı bilgiler verilmiştir.

Dördüncü bölümde, Oracle 9i XML destekli veritabanında iki farklı şekilde tutulan aynı içeriğe sahip belgeler üzerindeki sorgu performans hızlarını etkileyecek olan dizinleme yöntemlerinin kullanılması sonucu yapılacak karşılaştırmalar için bir uygulama programı yazılmış, ilgili ara yüzler oluşturulmuş ve çeşitli XML ve metin belgeleri ile performans analizleri yapılmıştır.

Son bölümde ise, tez çalışmasında elde edilen sonuçlar ve gelecekte yapılması düşünülen fikirlere yer verilmiştir.

BÖLÜM 2

XML'İN GENEL ÖZELLİKLERİ VE KULLANIMI

2.1 XML Nedir?

XML, HTML diline çok benzer bir yapıya sahip olan bir işaretleme dilidir. Verilerin tanımlanabilmesi ve tarif edilebilmesi için kullanılmaktadır. HTML'de kullanılacak olan etiketlerin önceden tanımlanmasına rağmen XML'de böyle bir tanımlama mevcut değildir. XML belgesinin yapısı tamamı ile kullanıcı tarafından oluşturulmaktadır. Verilerin tarif edilmesi için Belge Tipi Tanımlama (Document Type Definition - DTD) adı verilen yapılar kullanılmaktadır. DTD yapıları XML belgesinde kullanılacak etiketlerin ve yapıların önceden belirlendiği kütüklerdir. Böylelikle XML ve DTD'nin birlikte kullanılması ile belgeler kendini tarif eden bir yapı halini almaktadırlar. XML, SGML'den türetilmiş bir dildir. SGML kendisinden başka diller yaratılmasını sağlayan bir meta-dildir. SGML'den türetilmesine rağmen XML'de bir meta-dildir. Dolayısı ile XML'den de yeni diller türetilebilmektedir. Örneğin mobil cihazlar için kullanılan ve WAP ortamlarının dili olan Kablosuz İşaretleme Dili (Wireless Markup Language- WML) XML'in bir türevidir. XML'in HTML'den en büyük farkı HTML'inde SGML'den türetilmiş bir dil olmasına rağmen meta-dil özelliği göstermemesidir. Bununla birlikte XML verinin kendisi ile ilgilenirken HTML verinin sunumuyla ilgilenmektedir. HTML belgeleri veriye ilişkin şekillendirme bilgileri içerirken, XML belgeleri verinin tanım bilgilerini içermektedir. Verinin taşınması XML'in tasarım amaçlarından biridir. Bütün bu özellikler XML'in HTML'in yerine geçecek bir dil olması yerine HTML'in tamamlayıcısı olmasını sağlamaktadır. Günümüz bilişim dünyasına bakılacak olursa hemen hemen her yerde XML'e rastlamamız mümkündür. XML'in bu kadar yaygın kullanılması, geleceğin WEB dili olacağı beklentisini yaratmaktadır. XML, bileşik belgeler yaratmak için çoklu kütüklerin birleştirilebilmesi, metin kütüklerine istenilen biçimde ve yerde resim eklenebilmesi, programa bilgi denetim işlemini sağlayabilmesi, metin harici açıklama satırlarının eklenebilmesi gibi özellikleri kullanıcılara sunmaktadır. Bütün bu özelliklerine ek olarak, HTML'deki gibi daha önceden tanımlanmış etiketlerin ve belgelerin üretilmesi için gerekli olan standart şablonların tanımlanmasına ihtiyaç

duyulmamaktadır. W3C, XML'i internet üzerinde çalışabilmesi, tüm uygulamalar tarafından desteklenebilmesi, SGML ile uyumlu olması, üreten programların oldukça kolay olması, seçmeli özelliklerin çok az olması, belgelerin okunabilirliliğinin yüksek olması, tasarımının biçimlendirilmiş ve kısa olması, belgelerin yaratılmasının basit ve anlaşılır olması, belgelerde anlam belirsizliklerinin bulunmaması gibi amaçlar doğrultusunda tasarlamıştır.

2.2 XML Kullanım Şekli

XML veri taşımak amaçlı tasarlanmıştır. XML sayesinde veri modülerlik kazanmakta verilerin içeriği ile ilgilenilmektedir. Bu sayede verilerin içerik, yapı ve sunum kısımları ayrı parçalar halinde farklı XML belgelerinde tutulabilmektedir. XML belgeleri Veri Adaları (Data Islands) adı verilen teknik sayesinde HTML sayfaları içerisinde de depolanabilmektedir. Bu teknik sayesinde verilerin sadece sunumu ile ilgilenebilmek mümkün olmaktadır. Böylece HTML sayfasının içine sadece HTML etiketleri değil, aynı zamanda XML komutları da yazılabilmektedir. Veri adaları HTML sayfalarına iki değişik şekilde gömülür. Bunlardan birincisi, doğrudan XML verisinin HTML'in içine gömülmesidir. Aşağıdaki örnekte XML verisi doğrudan HTML'in içine gömülmüştür.

```
<XML ID="XMLID">
<iller>
        <ilkod>06</ilkod>
        <ilad>ANKARA</ilad>
</iller>
</XML>
```

İkinci yöntem ise HTML sayfasından XML'i referans göstererek veri adasının eklenmesidir. Aşağıdaki örnekte ise XML verisi HTML sayfasından referans gösterilmiştir.

```
<XML ID="XMLID" SRC="iller.xml"></XML>
```

Dolayısı ile HTML sayfalarına gömülen XML sayesinde verinin alışveriş işlemleri gerçekleştirilmektedir. XML yapısının esnekliği sayesinde birbirine uyumlu olmayan sistemler arasında veri alışverişi rahatlıkla gerçekleştirilmektedir.

Günümüz bilişim dünyasında bilgisayar sistemleri ve veritabanlarının genellikle birbirine uyumsuz sistemler içerebildiklerini görmekteyiz. Bundan dolayı uygulama geliştiriciler internet üzerinden bu tip uyumsuz verilerin alışveriş işlemini gerçekleştirmek zorundadırlar. Verinin XML biçimine çevrilmesi ile farklı sistemler ve uygulamalardaki verilerin karmaşıklık derecesi indirgenerek, alışveriş işleminin kolaylaştırılması sağlanır.

XML ile finansal bilgilerin internet üzerinden alışverişi de sağlanabilmektedir. Günümüzde artık sıklıkla duyulan elektronik iş kavramı açısından incelenecek olduğunda, XML'in önemli işlevleri yerine getirdiği görülmektedir. Bahsedilen bu işlevler ile XML geleceğin elektronik iş dili olarak da yeni bir görevi üstlenmektedir.

XML, verilerin paylaşımını da kolaylaştırmaktadır. XML, veriyi düz metin (plain text) biçiminde saklamasından dolayı veriyi paylaştırma konusunda hem yazılım hem de donanımdan bağımsız hareket edebilme özelliğini sunmuştur. Bu özelliği sayesinde farklı uygulamalarda hareket eden farklı veri tipleriyle çalışmak daha da kolaylaşmaktadır. XML, ayrıca işletim sistemlerinin yükseltilmesi, sunucu, uygulama ve benzeri dışsal faktörlerin yenilenmesi gibi kullanıcı için problem yaratabilecek olumsuzluklardan en az düzeyde etkilenmesini sağlamaktadır.

XML ile verilerin depolanması sağlanmaktadır. XML, verilerin dosyalarda veya veritabanlarında saklanması için de kullanılmaktadır. XML, yazılım, donanım ve uygulamalardan bağımsız olduğu için verinin daha elverişli olarak kullanımını sağlamaktadır. Yani başka istemci (client) veya uygulamalar, tıpkı veri kaynaklarına erişiyormuş gibi XML kütüklerine rahatlıkla erişebilirler.

2.3 XML'in Yararları

XML'in diğer yararları aşağıdaki şekilde sıralanabilir;

- Yalınlık; XML'de doğal dile yakın anlamlı etiketler insanlar tarafından daha kolay anlaşılır dolayısı ile XML'de kodlanan bilgilerin okunması ve anlaşılması kolaydır.
- Açıklık; XML W3C standardıdır ve yazılımda lider firmalar tarafından da kolaylıkla onaylanmıştır.
- Genişleyebilirlilik; XML' de sabit etiketler kümesi bulunmamaktadır. İstenildiğinde yeni, farklı etiketler yaratılabilmektedir.
- Kendi kendini tanımlama; veritabanlarında genellikle veri kayıtları, veritabanı sorumlusu tarafından belirlenen ve kurulan şemalara ihtiyaç duymaktadır. XML belgelerinde bu tür tanımlamalara ihtiyaç duyulmamaktadır. XML etiket ve özelliklerden oluşan bir meta içermektedir. Dolayısı ile ayrıca bir şema tanımlanmasına gerek duyulmamaktadır.
- Makine tarafından anlaşılabilen bağlam bilgisi içerir; etiketler, özellikler ve eleman yapıları, etkin arama motorlarında, içerik anlamlarının yorumlanmasında, yeni olasılıklar oluşturmada, akıllı veri işlemede (Data Mining) bağlam bilgisi sağlar. HTML veya düz metinde bu bağlam bilgisinin değerlendirilmesi mümkün değildir. XML ile birlikte bağlam bilgisinin değerlendirilebilmesi, XML'e HTML ve düz metine oranla büyük bir avantaj sağlamaktadır.
- İçeriği gösterimden ayırır; HTML görünüm ile ilgilenirken XML anlamla ilgilenmektedir. Dolayısı ile bir belgenin görünümünün, belgenin bağlamına dokunmadan değiştirilebilmesi sağlanmaktadır. Aynı içeriğe sahip farklı görünümlerde belgeler ve sunumlar oluşturulabilmektedir.
- Çok dilli belgeleri ve Evrensel Kodu (Unicode) destekler; Evrensel Kod Unicode Consortium organizasyonu tarafından geliştirilen ve her karaktere bir sayı değeri karşılığı atayan bir standarttır. XML uygulamalarının uluslararası anlaşılabilirliği için bu özellik çok önemlidir.

- Verilerin kümelenmesi ve karşılaştırılmasını kolaylaştırır. XML belgeleri ağaç yapısı halinde oluşturulduğundan belgenin elemanları etkin bir şekilde kümelenmiş ve karşılaştırılmasına imkân sağlanmıştır.
- Çoklu veri türleri ile kullanılabilir. Çoklu verilerden (resim, ses, video, vb.) aktif bileşenlere (ActiveX, Java Appletleri, vb.) kadar birçok olası veri türü ile birlikte kullanılabilmektedir.
- Var olan veri türleri ile kullanılabilir. İlişkisel veritabanlarının XML'le ilişkilendirilmesi mümkündür.
- Dağıtılmış veriler için tek bir sunucudan görünüş sağlar. XML'in kullanmış olduğu birçok veri değişik veritabanları içerisinden sağlanabilir. XML sayesinde bu verilere tek bir sunucu üzerinden rahatlıkla erişilebilmektedir.
- Sanayi tarafından hızlıca benimsenmiştir. Açıklık özelliği sayesinde sektörün önde gelen IBM, Sun, Microsoft, Netscape, SAP gibi firmaları tarafından XML desteklenmektedir.

2.4 XML'in Sözdizimi

XML'in sözdizimi basit kurallar üzerine oluşturulmuştur. Aşağıdaki gibi bir XML belgesi incelendiğinde;

```
<?xml version="1.0"?>
<not>
        <kime>Yüce</kime>
        <kimden>Ece</kimden>
        <baslik>Kutlama</baslik>
        <ileti>Nice Mutlu Yıllara!!</ileti>
</not>
```

XML belgesinin ilk satırında yapılan bir bildirim ile XML sürümü belirlenir. Hemen altındaki satırda DTD bildirimi de yer alabilir. Sonraki satırda ise XML belgesinin kök (root) elemanı açıklanır. Kök elemanından sonra gelen elemanlar XML sıra düzenine uyacak şekilde kök elemanına bağlı durumdadırlar. XML belgelerinin yapısı ağaç yapısına benzemektedir. Kök elemanından sonraki dört satır, dört adet çocuk (child) elemanıdır. Bu çocuk elemanlar birbirleri ile kardeştir (sibling). Kök elemanı da çocuk elemanların atası (parent) durumundadır. XML belgelerinde kök, çocuk ve kardeş elemanlarının sıra düzensel (hierarchical) ve iç içe geçmemiş bir

yapıda olması gerekmektedir. Yukarıdaki örnekte <not> kök eleman, <kime>, <kimden>, <başlık> ve <ileti> <not>'un çocuğu, <not> bu dört çocuk elemanın atasıdır. Ayrıca dört çocuk eleman birbirleri ile kardeştir. İstenildiğinde çocuk elemanlara, kendilerine ait çocuk elemanlar da eklenebilmektedir. XML belgelerinde, başlangıç ve bitiş etiketleri birbirleri ile sıkı bir bağ içerisindedir. XML belgelerinde kullanılan etiketler büyük / küçük harf duyarlıdırlar (case-sensitive). Dolayısıyla bir elemanın başlangıç ve bitiş etiketlerinin aynı harf duyarlılığında olması gerekmektedir. XML belgeleri sıra düzensel bir söz dizimine sahiptirler. Sıra düzensel olmayan bir söz dizimi hatalı olarak nitelendirilir. XML elemanları kendilerine ait özelliklere sahip olabilirler.

<not tarih="20/02/2002">

Yukarıda bahsedildiği gibi, XML söz dizimi açısından katı kurallara sahip olmasına rağmen, verinin saklanması ve temsil edilmesi bakımından çok esnek bir yapıya sahiptir.

2.5 DTD (Document Type Definition)

XML belgelerinde kullanılan etiketler kişilerden kişilere farklılık gösterebilir. Bu yüzden, kullanılan etiketin XML kütüğü içindeki işlevi DTD'nin içinde yer alan tanımlarla açıklanmaktadır. DTD, XML kütüğünün içinde (içsel alt küme olarak), dışında (dışsal alt küme ve genellikle .dtd uzantılı olarak) ya da hem içsel alt küme hem de dışsal alt küme olarak bulunabilir. XML'in esnek bir dil yapısı olması ile XML belgesinde kullanıcının belirlediği özelliklere sahip bir element, yine kullanıcının belirlemiş olduğu şekilde kullanılabilir. Bu nedenledir ki, XML belgeleri düzensizlik ve dağınıklık potansiyeline de sahiptirler. XML belgelerinde yer alan dağınıklık ve düzensizliği önlemek amacıyla mantıksal ve pratik kısıtlar konularak var olan esnekliğin dizginlenmesi gerekmektedir. Bu bağlamda DTD kendisini kullanan XML belgesinin yapısını mantıksal ve pratik olarak kısıtlar veya tanımlar. DTD sayesinde XML belgesinden türetilmiş yeni belgelerin XML işleyicileri tarafından aykırı bir durumda algılanmaması sağlanmaktadır. DTD'ler XML belgelerine modülerlik kazandırır. XML belgelerinin amacı ve yapısı hakkında bilgiler DTD içinde bulunmaktadır. İyi hazırlanmış bir DTD ile çok büyük boyutlardaki XML belgeleri hakkında etkin bir görüşe sahip olunmaktadır. Bununla birlikte, XML belgesi ile ilişkili

olan tanımlama (definition) yine XML belgesi içinde yer alan bir bildiri (declaration) ile tanımlanmaktadır.

2.6 XML Parsing (Ayrıştırma) İşlemleri

XML belgesi aslında arka arkaya sıralanmış, gerçek bir anlamı olmayan karakterler dizisinden oluşan yalın bir metin belgesi olarak görülebilir. Bir metin düzenleyici, XML kütüğünü bir dizi karakterler sıralaması şeklinde görür ama bu karakterlerin içindeki mantıksal yapı hakkında hiçbir fikri yoktur. Dolayısı ile bu karakterler dizisinin anlamına erişmek için mantıksal bir yapıya çevrilmelidir. Bir XML ayırıcısı sayesinde bu mantıksal yapıya dönüştürme işlemi gerçekleştirilebilir. XML belgelerinin söz dizimi ve mantıksal içeriği XML ayrıştırıcıları tarafından denetlenir. Bütün ayrıştırıcılar XML belgesinin iyi oluşturulmuş (well formed) olup olmadığını kontrol eder. Bununla birlikte ayrıştırıcılar DTD ile belgenin uygun eşlenip eşlenmediğini doğrularlar. XML belgeleri iki kıstasa uygun olmalıdır. Bunlar iyi oluşturulmuş ve geçerli (valid) olmasıdır. Bir XML belgelerinin iyi oluşturulmuş olması, XML için belirlenen temel kurallara uygun olması ile geçerli olması ise kendi DTD'si veya şemasında tanımlanmış kurallara uygun olması ile belirlenir. Bir XML belgesi XML kurallarına göre oluşturulmuş olduğundan iyi oluşturulmuş bir yapıya sahip olmasına rağmen XML belgesindeki bazı elemanların tanımlarının DTD'de bulunmamasından dolayı geçerli olmayabilir. Bir uygulama, XML belgesinin anlamına dört farklı yöntem ile ulaşabilir. Bunlardan en çok kabul göreni SAX (the Simple API for XML)'tır. İkincisi DOM (the Document Object Model) isimli test edilmiş ara yüzdür. Üçüncü yöntem bir API (Application Programming Interface)'ye sahip XML ayrıştırıcısı kullanmaktır. Son yöntem ise kendi bilgisayar yazılımımızı üreterek kendi ara yüzümüzü kullanmamızdır. Temel olarak XML API'leri ağaç tabanlı (tree based) ve olay güdümlü (event-driven) olmak üzere iki çeşittir. Ağaç tabanlı API'ler XML belgesini bir ağaç yapısına dönüştürür ve XML belgesini kullanan uygulamanın bu ağacı dolaşmasını sağlar. DOM isimli API ağaç tabanlı API'lere bir örnektir. Veri merkezli olmayan belgeye dayalı bir uygulama yani belgenin içindeki verilerle değil, belgenin kendisi ile olan işlemlere daha çok ağırlık veriyor ise ağaç tabanlı API'lerin kullanılması daha kullanışlıdır. Ağaç tabanlı API'lerin kullanılması sistem kaynaklarında ağır bir yük oluşturabilmektedir. Bununla

birlikte olay güdümlü API'ler ayrıştırma sırasında elemanları doğrudan uygulama programına aktarırlar. Bu nedenle XML belgelerinin ağaç yapısına dönüştürülmesine ihtiyaç duyulmamaktadır. SAX olay güdümlü API'lere bir örnektir.

2.6.1 SAX (Simple API for XML)

Bazı XML standartları W3C tabanlı olmamaktadır. SAX da bu standartlardan biridir. W3C tabanlı bir standart olmayan SAX ile ağaç tabanlı API'lere göre XML belgesine daha alt düzeyde ve basit bir erişim sağlanmaktadır. Çok büyük ve hafızaya sığmayan belgelerin SAX ile ayrıştırılması mümkündür. SAX ile XML belgesi tarandığında her bir etiket ile bir olay ilişkilendirilir. İlişkilendirilen olayın uygulama tarafından fark edilmesi sağlanır. Bu sayede uygulamanın amacına göre her olay ayrı ayrı ele alınır. Kullanıcılar tarafından SAX'ın ürettiği olaylar için ayrı program parçacıkları yazılabilir. SAX tarafından üretilen olaylar ile XML belgesinin tutarlılığını sınamak, olayın kullanıldığı uygulama parçasına bırakılmıştır. Uygulamanın kullandığı olaylar için belgenin tümünün hafızada tutulmasına gerek yoktur. Aşağıdaki XML belgesi incelenecek olursa;

```
<?xml version="1.0"?>
<belge>
        <para>Örnek</para>
</belge>
```

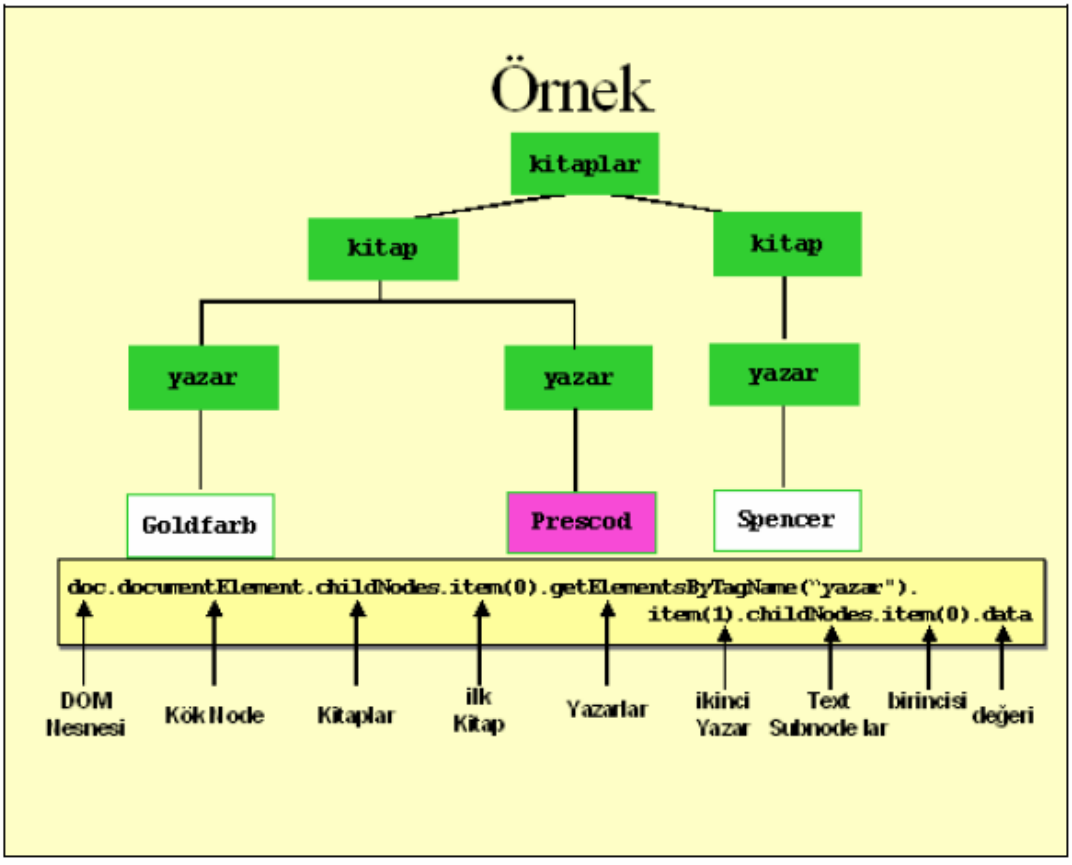
SAX kullanan bir ayrıştırıcı yukarıdaki belgeyi aşağıdaki şekilde bir dizi olay haline getirir.

```
startDocument
startElement: belge
startElement: para
characters: Örnek
endElement: para
endElement: belge
endDocument
```

İstemci durumunda bulunan uygulama yukarıdaki olayları ele alır ve gerekli olan faaliyetleri yerine getirir. Yukarıda da görüldüğü gibi olaylar SAX tarafından oluşturulduktan sonra bütün belgenin hafızaya alınmasına gerek kalmaz ve belge içinde ilgilenilmeyen bölümler uygulama ile ele alınmak zorunda kalınmaz.

2.6.2 DOM (Document Object Model)

DOM W3C tabanlı bir modeldir. DOM ile XML belgesi düğümlerden oluşan bir ağaca dönüştürülür ve bu ağaç yapısı hafızaya alınır. Ağaç yapısına dönüştürülen belge hafızada bir kerede tutulmak zorundadır. Çok büyük XML belgelerini DOM tabanlı yaklaşımla ele almak hafıza yetersizliklerine sebep olmaktadır. Ayrıca çok büyük belgelerde DOM tabanlı yaklaşım kullanmak işlemlerin yavaşlamasına neden olmaktadır. DOM tarafından çevrilen ağaç yapısının üzerinde bulunan düğümlere ulaşımı kolaylaştırmak için değişik ara yüzler (interface) bulunmaktadır. Bu ara yüzler düğüm tipine göre değişik özellik ve yöntemler içerir.



Şekil 2.1: Örnek XML Belgesi.

Şekil 2.1'de ulaşılmak istenilen eleman pembe renkli prescod elemanı olsun. DOM nesnesinden documentElement ile kök düğüme, kök düğümden childeNodes.item(0) ile ilk kitaba, buradan .getElementsByTagName ("yazar").item(1) ile birinci kitabın ikinci yazarına buradan da .childNodes.item(0).data ile ulaşılmak istenilen prescod elemanına ulaşılmaktadır. Yukarıdaki örnekte kök düğüme ulaşılmak için "doc.documentElement", kök düğümün çocuk düğümlere ulaşmak için "childeNodes", çocuk düğümlerin etiket adlarına göre sahip olduğu düğümlere ulaşmak için "getElementsByTagName", seçili olan düğümün değerine ulaşmak için"data" komutları kullanılmıştır.

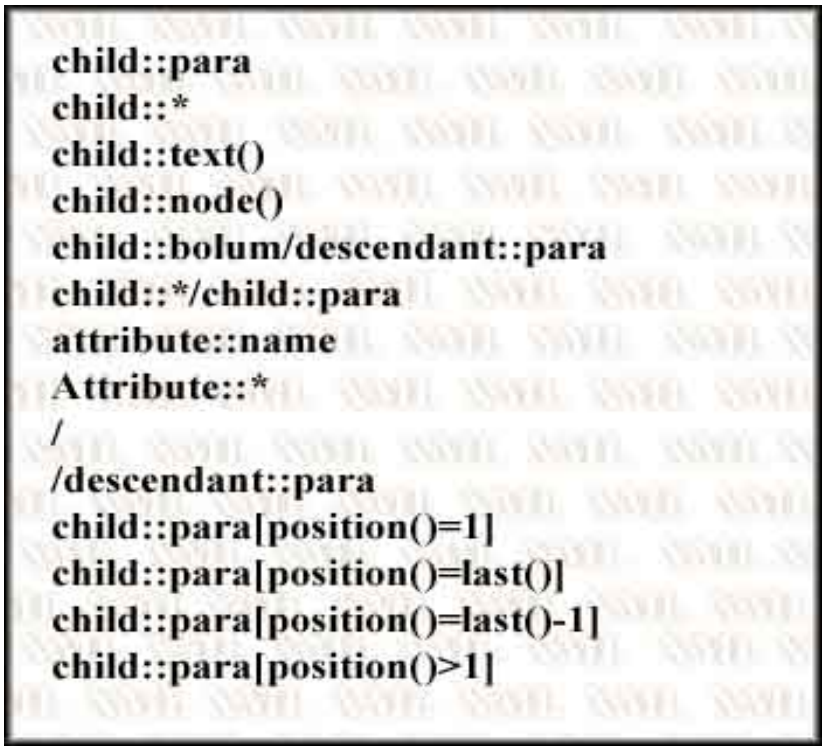
2.6.3 XPath

W3C'nin tanımına göre Xpath, "XML belgesinin ağaç yapısı içerisindeki elemanlarının adreslenmesi için geliştirilmiş bir dildir ve Xpointer tarafından da kullanılmaktadır." (www.w3.org/TR/xpath) Xpath sayesinde XML düğümleri ile belgenin içinde bulunan elemanlar arasında basit ve standart bir yapı sağlanır. Xpath'in kendine ait XML söz diziminden farklı bir söz dizimi bulunmaktadır. Bu söz dizimi sayesinde bir belgenin içindeki elemanların dizilişini gösteren şablonların oluşturulması mümkündür. XPath XML belgesini düğümlerden oluşan bir ağaç yapısında görür. XPath'in söz dizimi deyimlerden oluşmaktadır. Bu deyimler nesneleri işleyerek sonucunda bir kaç düğümün birleşimi (node-set), mantıksal (boolean) bir değer, sayı (number) veya dizi (string) türünden bir veri üretir. XPath'e göre ağaç yapısındaki XML belgesinin yedi türde düğümleri olabilir. Bunlar,

- Kök düğümler (root- nodes)
- Eleman düğümleri (element-nodes)
- Metin düğümleri (text-nodes)
- Özellik düğümleri (attribute-nodes)
- İsim Uzay Düğümleri (Namespace nodes)
- İşlem komutları düğümleri (processing instruction nodes).
- Yorum düğümleri (comment nodes).

Konum Yolu (Location path) sayesinde Xpath'te adresleme yapılır. Konum yolları ulaşılmak istenen yerin bulunulan noktaya (context node) göre yerini belirler. Xpath'te on üç adet eksen (axis) bulunur. Bunlar;

- “Child Axis”, bulunan düğümü çocuk düğümüdür.
- “Descendant Axis”, bulunan düğümün herhangi bir seviye altındaki alt düğümü.
- “Parent Axis”, bulunan düğümün bir seviye üstteki düğümü.
- “Uncestor Axis”, bulunan düğümün atası.
- “Following- sibling Axis”, bulunan düğümün devamındaki kardeş düğümler.
- “Preceding-Sibling Axis”, bulunan düğümden önce gelen kardeş düğümler.
- “Following Axis”, belgedeki sıralamaya göre bulunulan düğümden sonra yer alan alt düğümleri hariç bütün düğümler.
- “Preceding Axis”, belgedeki sıraya göre bulunulan düğümden önce yer alan ataları hariç bütün düğümler.
- “Attribute Axis”, bulunan düğümün özellikleri.
- “Namespace Axis”, bulunan düğümün ad uzay düğümleri.
- “Self Axis”, bulunan düğümün kendisi.
- “Descendant- Or- Self Axis”, bulunan düğümün kendisi ve ata düğümleri.
- “Uncestor-or-Self Axis”, bulunan düğümün kendisi ve alt düğümleri ifade eder.



Şekil 2.2: XPath Örneği

XML verileri üzerinde yapılan sorgulama işlemleri özellikle XPath eksenli çalışmalarda mevcuttur [2; 3].

2.6.4 XPointer

Herhangi bir URI (Uniform Resource Identifier) referansı için XML belgesi içinde kesimler yaratmak amacıyla XPointer kullanılmaktadır. Diğer bir deyişle XML belgesinin içyapısını ve belge içindeki XML kesimlerinin yerlerini adreslemek, Xpointer ile sağlanmaktadır. Sıra düzensel bir belgenin yapısının sınanması Xpointer ile mümkündür. Xpath'in içindeki bazı ifadelerin genişletilmesi amacıyla Xpath üzerine Xpointer kurulmuştur. Xpointer Xpath'in ifadelerini kullanır. Böylelikle Xpath kullanılarak XML belgesi içinde bulunan bir noktanın diğer bir nokta arasının adreslenmesi, dizi eşlemesi yapılan bilgilerin yerlerinin belirlenmesi ve URI referanslarındaki adresleme deyimlerinin kullanılması sağlanır. Xpointer, yerini belirlediğimiz bir nesneyi bağlamak (link) için bir taban olarak kullanılabilmektedir. Bu bağlamda Xpointer Xpath'e çok benzemekle birlikte çok daha fazla gelişmiş seçme özelliğine sahiptir.

```
<!DOCTYPE konusma [
<!ELEMENT konusma (#PCDATA|konusan|yon)*>
<!ATTLIST konusma
    ID   ID   #IMPLIED>
<!ELEMENT konusan (#PCDATA)>
<!ELEMENT yon (#PCDATA)>
]>
<konusma ID="a27">
 <konusan>Tolga</konusan>
 <yon> pencereden disari</yon>
 Acele et , ders basliyor !!
 <yon>Levent'e</yon>
 Mehmet birazdan sinifta olur
</konusma>
```

Şekil 2.3: XPointer Örneği

Yukarıdaki örnekte; id(a27).child(2,yon) İkinci “yon” elemanını seçer (içeriği Levent’e” olandır). id(a27).child(2,#element) İkinci çocuk elemanı seçer (bu ilk “yon”, içeriği “pencereden dışarı” olandır). id(a27).child(2,#text) İkinci metin bölümünü seçer, "Mehmet birazdan sınıfta olur" (“Acele et ders başlıyor” ilk bölgeyi göstermektedir.) Görüldüğü üzere #element ve #text yardımıyla içeriğini tam olarak bilmediğimiz bölümleri de seçebiliriz. Aşağıda bazı Xpointer deyimleri ile bu deyimlerin hangi bilgileri seçmekte kullanıldıkları bulunmaktadır:

- #element
  XML elemanlarını seçmekte kullanılır.
- #pi
  XML işleyici komutlarını seçmekte kullanılır.
- #comment
  XML açıklama satırlarını seçmekte kullanılır.
- #text
  Elemanlar arasındaki ve CDATA bölümlerindeki metin alanlarını seçmekte kullanılır.
- #cdata
  CDATA bölümleri içerisinde bulunan metin alanlarını seçmekte kullanılır.
- #all
  Yukarıdaki düğüm türlerinden hepsi için kullanılır.

2.6.5 XLink

Xlink söz dizimi, Xpath ve Xpointer’dan farklı olarak XML’e bağımlıdır. Temel bağlar ve daha karmaşık bağ yapıları için XLink temel bir yapı oluşturur. Bununla birlikte XLink internet bağları da (hyperlink) sağlamaktadır. W3C’ye göre, “XLink Xpointer ile koşut kullanılmalıdır.”(www.w3.org/TR/xlink) XML belgelerinde kaynaklar arası bağları tanımlayan ve yaratan elementlerin yerleştirilmesi XLink ile sağlanmaktadır. XLink kaynaklar arasındaki net ilişkidir. Bir kaynak (kütükler, programlar, resimler, sorgu sonuçları v.b.) ise bir bilginin ya da servisin herhangi bir adreslenebilir bileşenidir. XLink sayesinde XML belgelerinde ikiden fazla kaynak arasında bağ kurmamız, meta - veri’yi bağlarla birleştirmemiz bağlanmış kaynaklardan farklı yerlerdeki bağları ifade etmemiz mümkün olmaktadır.

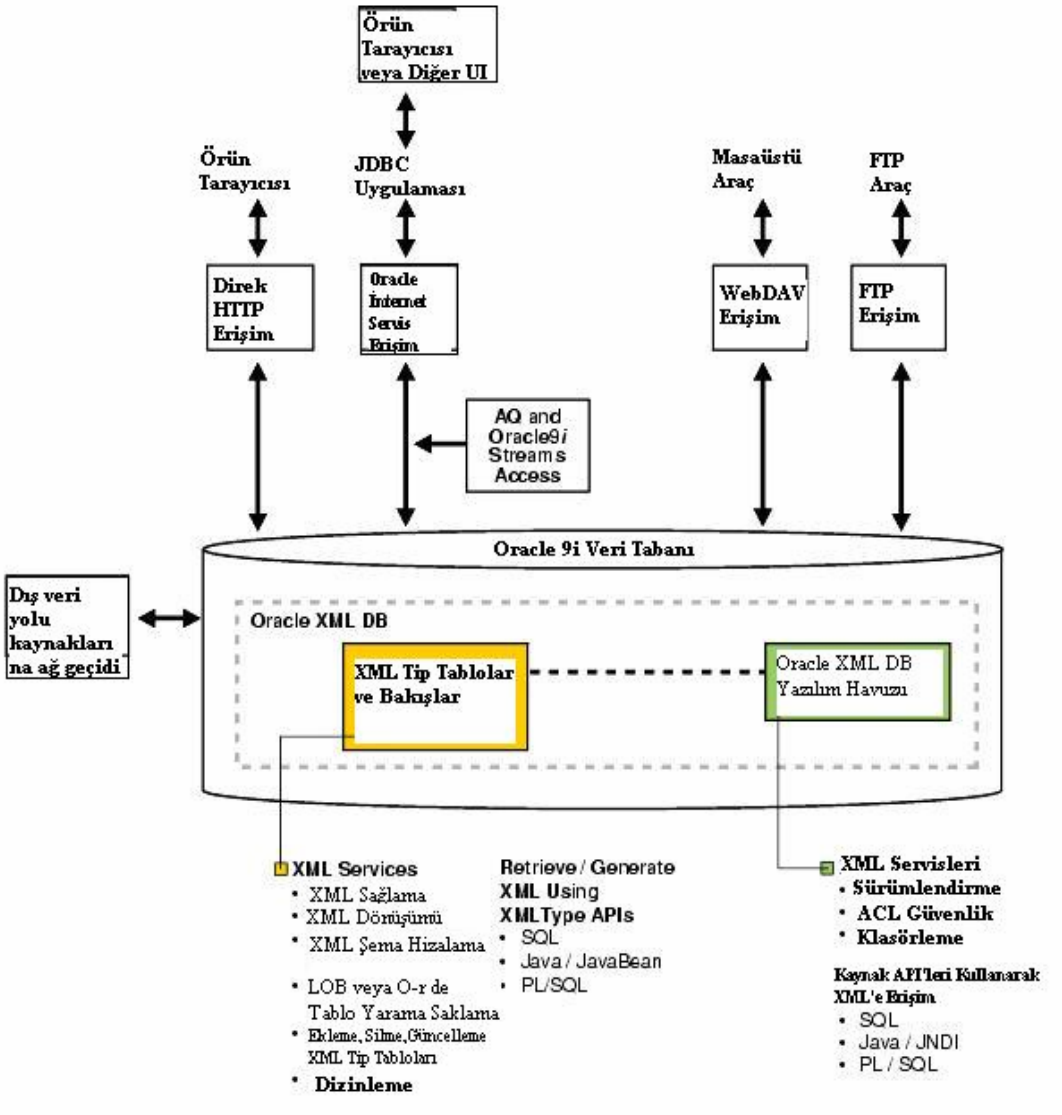
BÖLÜM 3

XML VERİTABANLARI

Bilindiği gibi XML hem dil hem de platformdan bağımsız bir işaretleme dilidir. Yukarıdaki bölümlerde, XML'in internet uygulamalarında iki bilgisayar arasındaki bilgi alışverişinde kullanıldığını gördük. Bu bölümde, belgelerin veritabanında saklanılarak gerektiğinde çeşitli sorgularla istenilen belgelerin bulunması ve gösterilmesinde XML'in kullanılışını inceleyeceğiz. Günümüzde XML için bir çok veritabanı bulunmaktadır. Tez kapsamında Oracle 9i XML DB incelenmiştir. XML veritabanlarında XML belgelerinin saklanması ve taşınması ile ilgili olarak çeşitli araştırmalar mevcuttur [4; 5].

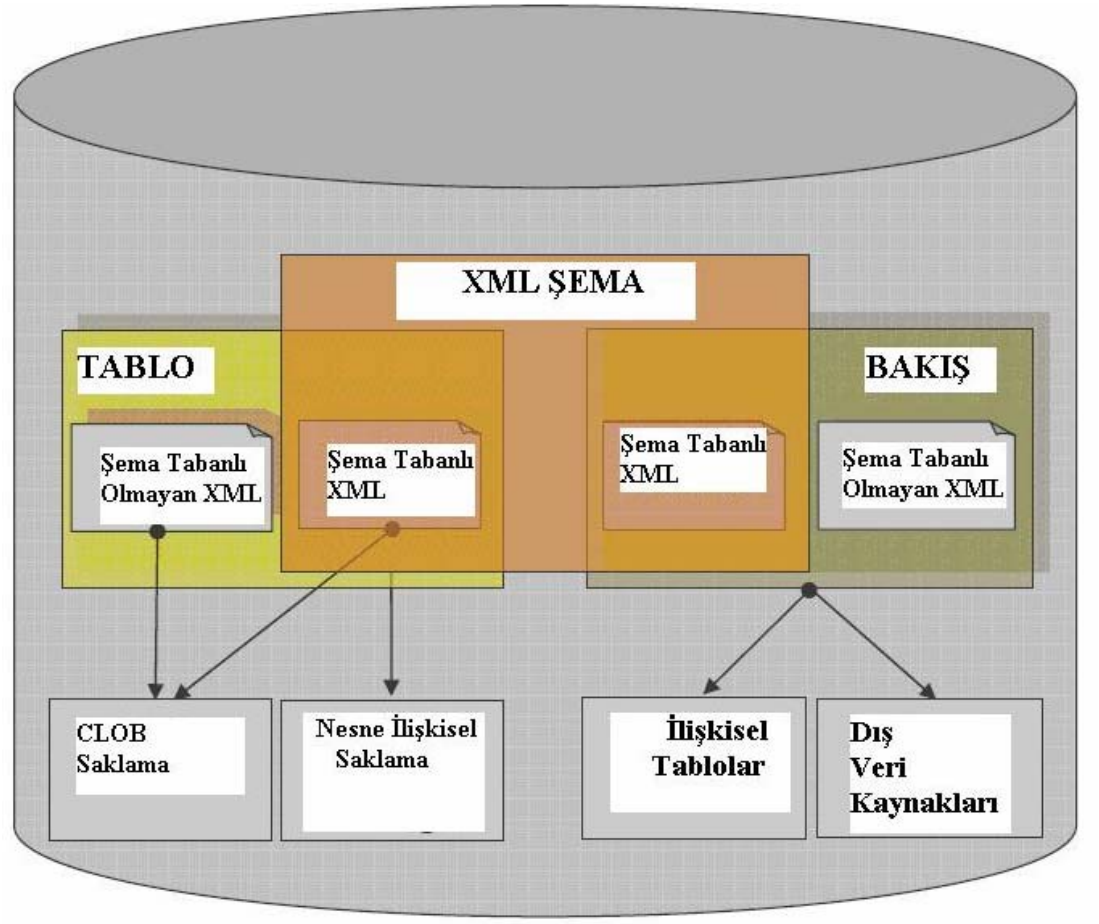
3.1 Oracle XML DB Mimarisi

Oracle XML DB, Oracle'ın ilişkisel veritabanından türetilmiş olup XML'e özgü fonksiyonların eklenmesi ile oluşmuş bir veritabanıdır. Oracle XML DB, XML verileri içerik ve kullanılan programlama dili bakımından bağımsız olarak saklayabilir ve yöntebilir. İçerik bakımından büyük olan belgeler ilişkisel veritabanlarında LOBs (Large Objects) olarak depolanmaktadır. XML'e özgü veritabanlarında XML belgeleri XMLType tablolar ve bakışlarla saklanarak yönetilmektedir. Şekil 4'de Oracle XML DB mimarisinde XMLType saklama ve bilgi havuzu görülmektedir. Şekil 3.1 incelendiğinde, veritabanına ulaşan dört farklı istek bulunmaktadır. Bunlardan ilki doğrudan HTTP erişimi yapan örün tarayıcısı (browser), ikincisi Oracle'ın servisleri ile erişim sağlayan JDBC uygulamaları kullanan örün tarayıcıları ve kullanıcı ara yüzleri (user interface), üçüncüsü Web DAV erişimini kullanan masaüstü araçları, sonuncusu ise FTP erişimini kullanan FTP araçlarıdır. Oracle XML DB XMLType tablolar ve bakışlarla Oracle XML DB bilgi havuzundan oluşmaktadır. XMLType tablolar ve bakışlar, XML doğrulama, çevirme, şema kayıtları, tabloların yaratılması, belgelerin LOB veya nesne ilişkisel olarak saklanması, tablolar üzerinde ekleme, silme, güncelleme ve dizinleme işlemlerinin yapıldığı XML servislerinden oluşmaktadır. Bütün bu işlemler SQL, Java/ Java Bean, PL/ SQL gibi XMLType API'ler kullanılmaktadır.



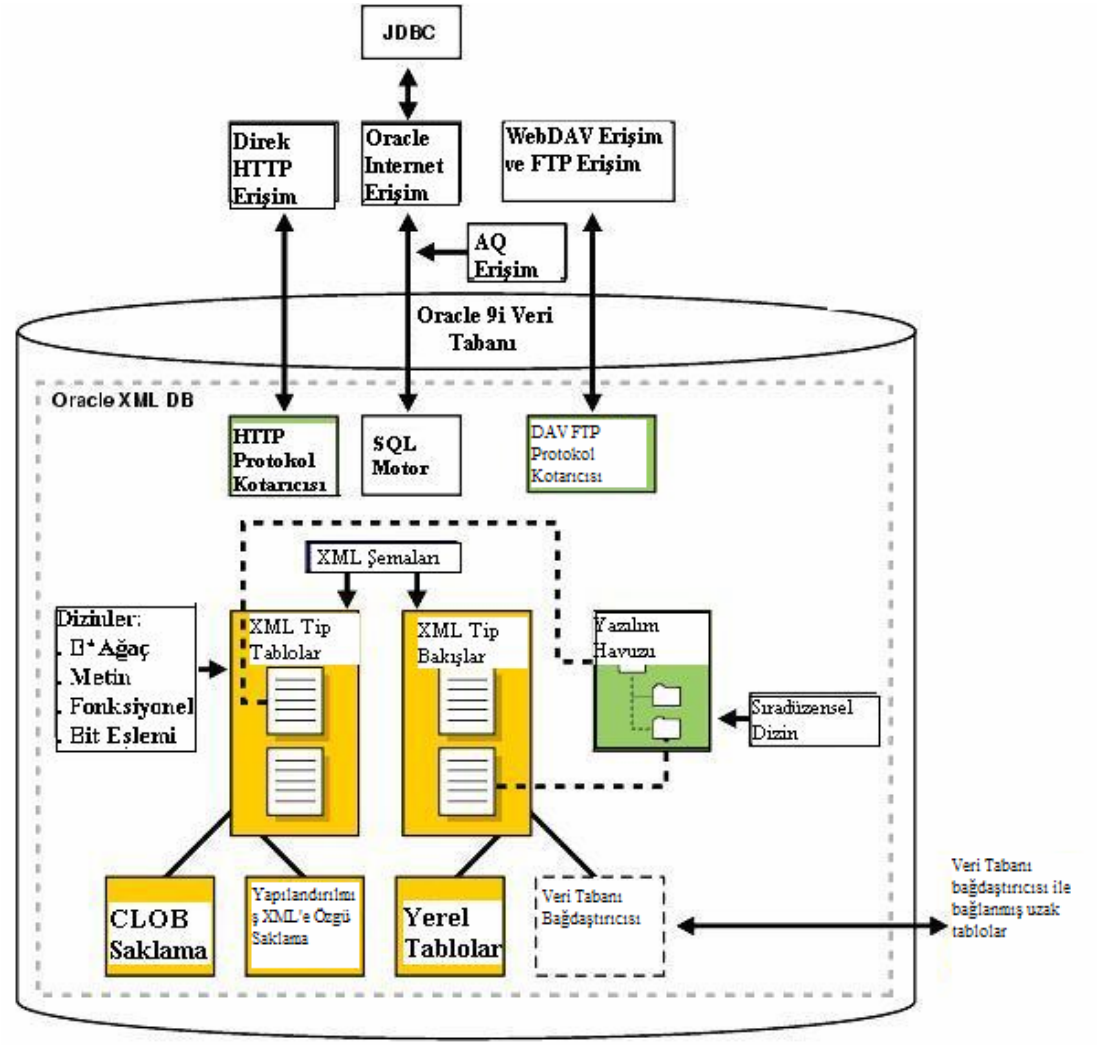
Şekil 3.1: XML Type Saklama ve Bilgi Havuzu Mimarisi [25]

Oracle XML DB bilgi havuzu sürümleri ise gizlilik ve dosyalama işlemlerinin kullanıldığı XML servislerinden oluşmaktadır. Bununla birlikte Oracle veritabanı dışsal kaynaklarla da bağlantı kurabilmektedir.



Şekil 3.2: XMLType Saklama Çeşitleri [25]

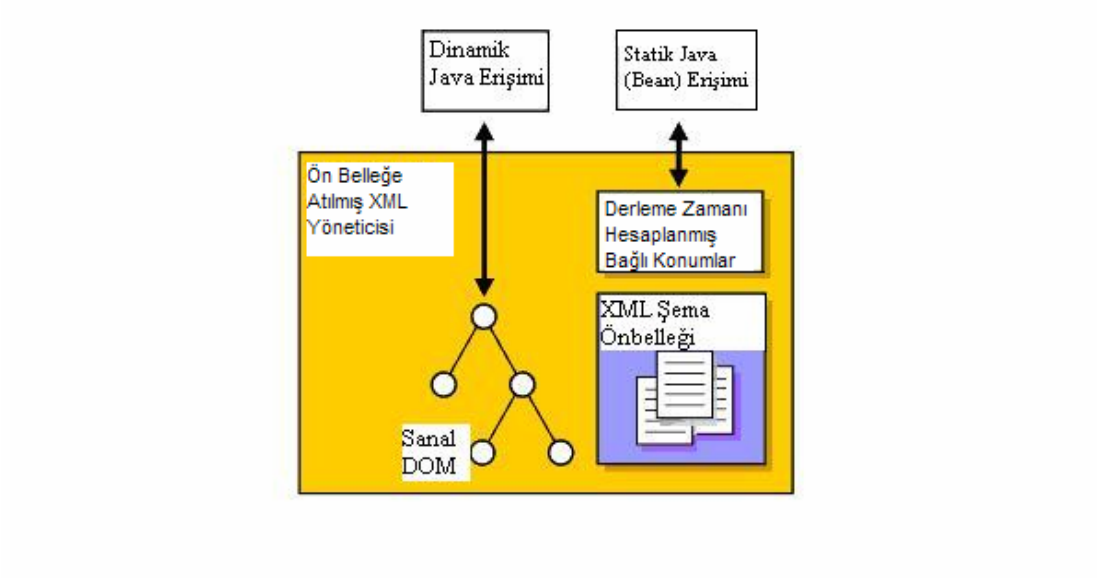
Şekil 3.2'de görüldüğü gibi XML belgeleri iki şekilde saklanabilir. Bunlardan ilki, ilişkisel tablolarda XMLType kolonlar şeklinde, diğeri ise XMLType tablolarda XML nesnesi şeklindedir. Şemalandırılmamış XML belgeleri daima CLOB (character large object) olarak saklanırlar. Şemalandırılmış XML belgeleri ise, CLOB alanlarda saklanabileceği gibi nesne kümesi olarak da saklanabilirler. İlişkisel ve dışsal veriler XML bakışları haline getirilebilirler. Bu bakışlar XML tipi kolonlardan ve başka XMLType bakışlardan oluşabilmektedir [6].



Şekil 3.3: XMLType Saklama ve Geri Erişim Mimarisi [25]

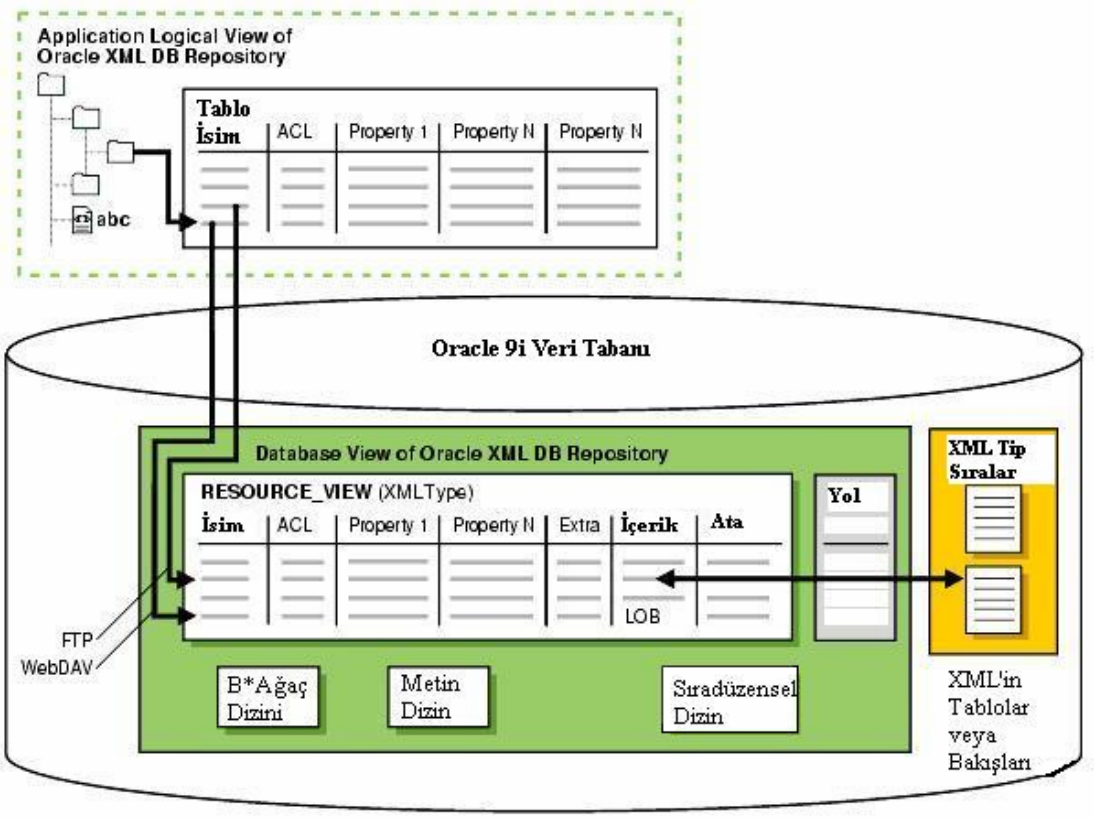
Şekil 3.3'te doğrudan HTTP erişimiyle, Oracle'ın net erişimini kullanan JDBC programı ile WebDAV ve FTP erişimi ile Oracle veritabanına bağlanan üç adet istemci bulunmaktadır. Doğrudan HTTP erişimi kullanan istemci Oracle XML DB'de bulunan HTTP protokol kotarıcıları (HTTP Protocol Handler) ile Oracle'ın net erişimini kullanan JDBC programı SQL motoru ile WebDAV ve FTP erişimini kullanan istemci DAV, FTP Protokol kotarıcıları (DAV, FTP Protocol Handlers) ile karşılanır. Veritabanında bulunan ve istemcilerin elde etmek istediği veriler, CLOB saklama ve şemalandırılmış XML'e özgü saklama metodlarını kullanan B*Tree, Text, Functional, Bitmap dizinlemesi yapabilen XMLType tabloları ile yerel tablolar ve DBLink ile uzak tablolara erişebilen XMLType bakışlar hiyerarşik bir dizinleme

yapan bilgi havuzundan geçerek elde edilmektedir. XML belgelerinin sorgulanması amacı ile veritabanı motorları hakkında çeşitli araştırmalar mevcuttur [7; 8; 9].



Şekil 3.4: Önbelleğe Atılmış XML Nesne Yönetimi Mimarisi

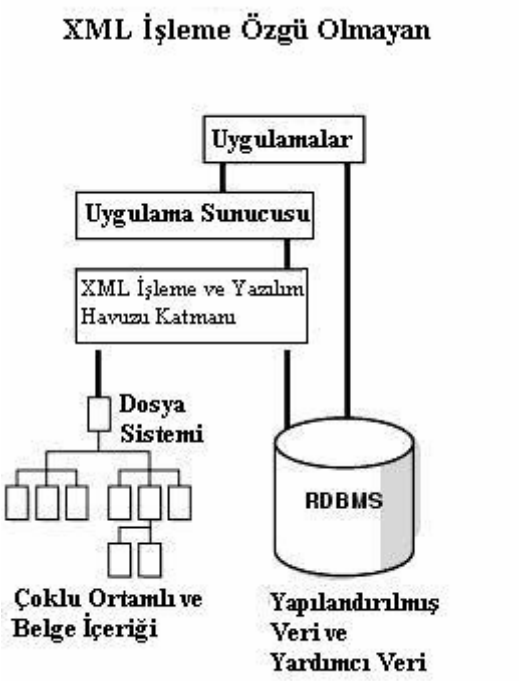
Şekil 3.4'te Oracle XML DB tarafından programatik olarak XMLType belgelere erişilmek istenildiğinde kullanılan önbelleğe atılmış XML nesnelerinin yönetim mimarisi görülmektedir. Daha önceden de belirttiğimiz gibi XML belgeleri DOM kullanılarak hafızaya alınmakta, dinamik olan Java erişimleri bu hafızada bulunan DOM'u kullanabilmektedirler. Statik erişimler de mümkün olabilmektedir. Fakat bu tip erişimlerde DOM'un tekrar derlenerek DOM içinde bulunan düğümlerin yerlerinin tekrar hesaplanması gerekmektedir.



Şekil 3.5: Oracle XML DB Bilgi Havuzu Mimarisi [25]

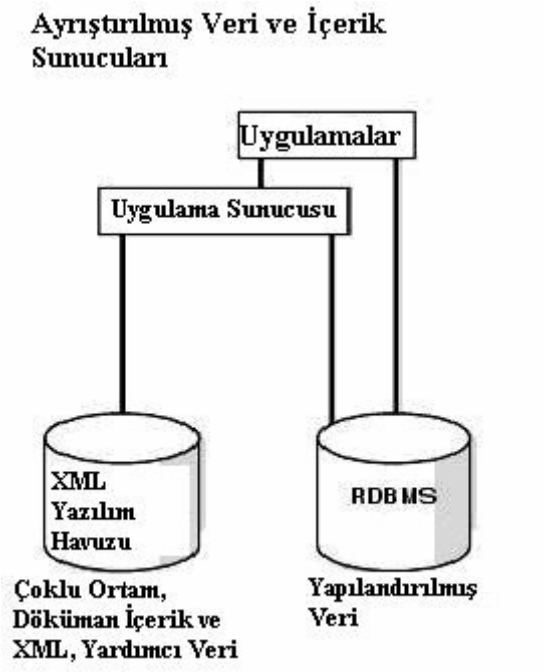
Şekil 3.5'te, XMLType olan kaynak bakış içinde bulunan içerik alanı, XML tabloları veya bakışlarından elde edilmiş olan XMLType satırlarının LOB olarak tutulmasından meydana gelmiştir. B*Tree, Metin, hiyerarşik dizinleme yapan bilgi havuzu uygulama tarafından istenilen bilgileri mantıksal olarak dosyalamaktadır.

XML belgelerine uygulamalar tarafından üç farklı mimaride erişmek mümkündür. Bunlar sırası ile XML'e Özgü Olmayan işleme, Ayrık Veri ve İçerik Sunucuları mimarisi ve Oracle XML DB mimarisidir.



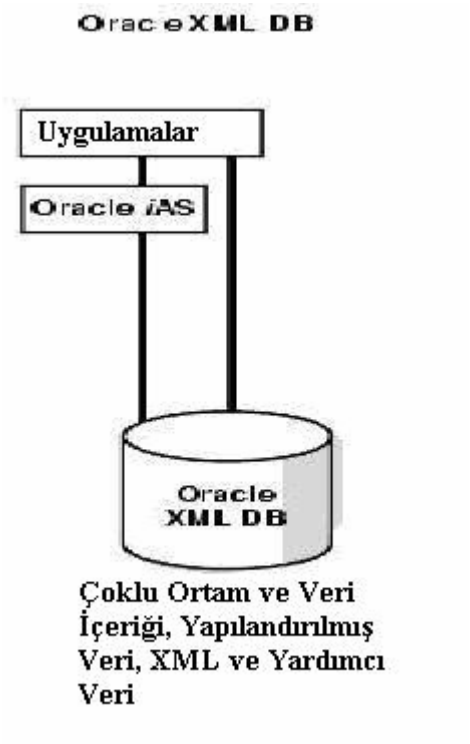
Şekil 3.6: XML'e özgü olmayan işleme [25]

Uygulamalar dosya sistemi içinde bulunan veya ilişkisel veritabanında tutulan XML belgelerine uygulama sunucusu vasıtası ile XML işleme ve bilgi havuzu katmanı ile ulaşabilmektedirler.



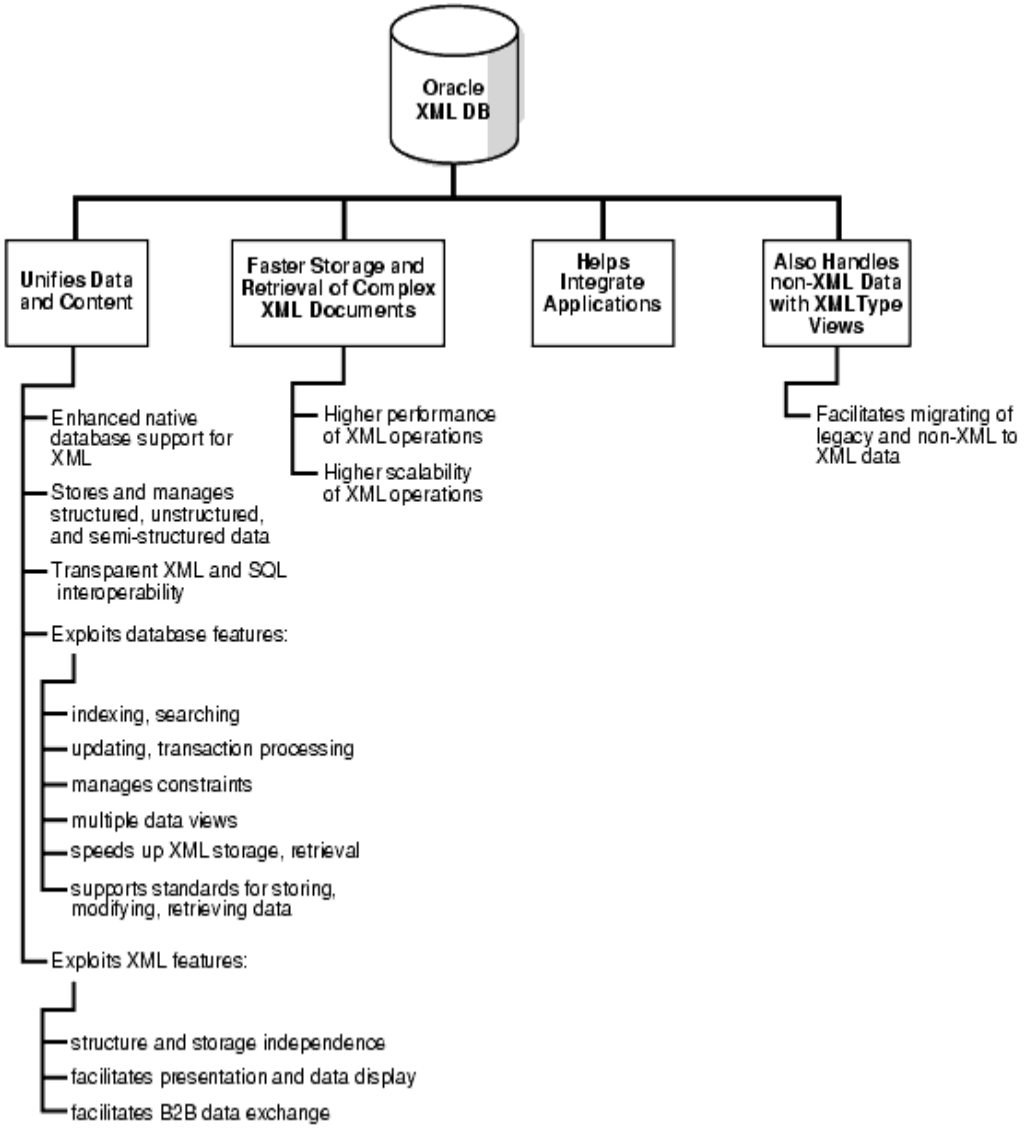
Şekil 3.7: Ayrık Veri ve İçerik Sunucuları [25]

Ayrık verilerde XML belgeleri ilişkisel veritabanından ayrı olarak XML bilgi havuzunda tutulmaktadırlar. Uygulamalar uygulama sunucuları vasıtası ile ayrı olarak tutulan XML belgelerine ulaşabilmektedirler. Aynı zamanda uygulamalar ilişkisel veritabanına da erişebilmektedirler. İçerik ve veri bakımından ilişkisel veritabanından ayrı tutulan XML belgeleri ile, ilişkisel veritabanında bulunan yapılandırılmış verilere uygulama sunucuları ayrı olarak erişebilmektedirler.



Şekil 3.8: Oracle XML DB [25]

Oracle XML veritabanı yine Oracle'ın ilişkilendirilmiş veritabanına XML'e özgü fonksiyonların eklenmesi ile türetildiğinden, XML belgeleri ile ilişkilendirilmiş veritabanında bulunan veriler ayrı olarak tutulmamaktadır. Dolayısı ile uygulamalar Oracle XML veritabanına erişmeleri halinde hem XML belgelerine hem de ilişkilendirilmiş veritabanında bulunan verilere ulaşabilmektedirler.

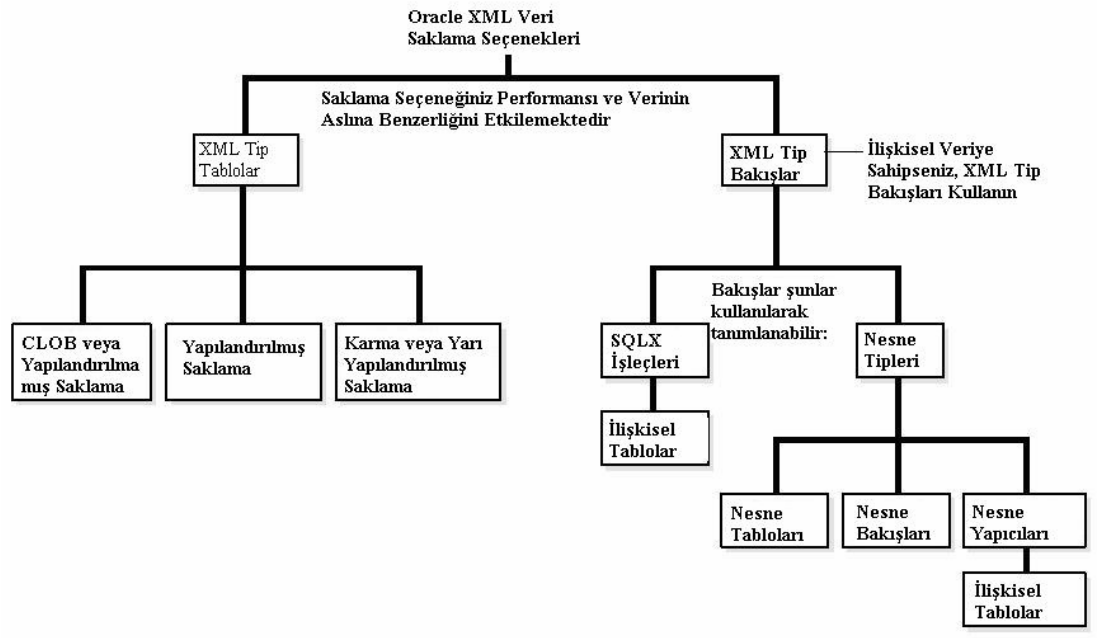


Şekil 3.9: Oracle XML Veritabanının Faydaları [25]

Şekil 3.9'da görüldüğü gibi, XML belgeleri ile ilişkisel veritabanında tutulan verilerin birarada tutulduğu Oracle XML veritabanının faydaları aşağıdaki şekilde sıralanabilmektedir.

- XML'e özgü fonksiyonlarla iyileştirilmiş bir veritabanı olması,
- Yapılandırılmış, yapılandırılmamş ve yarı yapılandırılmış XML'lerin saklanması ve yönetilmesi,
- XML ve ilişkisel veritabanı sorgulama dili olan SQL'in birlikte işlerliği,

- XML verileri üzerinde dizinlemenin, aramanın, güncellemenin, hareket işlemlerinin etkin olarak yönetilmesi, çoklu veri bakışlarının bulunması, XML saklama ve geri çağırımlardaki hız, sıralama, değiştirme ve geri çağrılan verilerin standart olarak desteklenmesi,
- Yapı ve saklamadaki bağımsızlığı,
- Sunum ve veri gösterim kolaylıkları,
- Veri değişimi olanakları,
- XML veriler üzerindeki işlemlerin yüksek performansta ve yüksek ölçeklenebilirlikte gerçekleştirilmesi,
- XMLType bakışların yerel ve uzak kaynaklardan oluşturulabilmesi,
- Diğer veritabanları ve dosyalara bağlanılabilirliği,
- Çoklu kaynaklardan elde edilen birleştirilmiş verilerin SQL ve XML sorgularında bir biçimli (uniform) olarak kullanılabilirliği,
- XML verilerle XML olmayan yani yeni bir teknoloji ile beraber eski bir teknolojiye ait olan belgelere ve bileşenler arasında geçişe imkan sağlaması Oracle XML veritabanının en büyük faydalarıdır.



Şekil 3.10: Yapılandırılmış Saklama Seçenekleri [25]

XMLType tablosu kullanılarak bir veri saklanmasında, yapılandırılmamış veriler için CLOB saklama, yapılandırılmış saklama, birleştirilmiş veya yarı yapılandırılmış saklama modelleri kullanılmaktadır. Varolan ilişkisel verilerle birlikte XMLType bakışları kullanılmaktadır. Hem ilişkisel tablolar hem de XML belgeleri kullanılarak yapılan işlemlerde SQLX fonksiyonları kullanılmakla birlikte, nesne tipi olarak da nesne tabloları, nesne bakışları ve ilişkisel tablolarda tutulan verilerin saklanması için nesne yapıları kullanılmaktadır.

3.2 Oracle XML Veritabanının Kullanımı

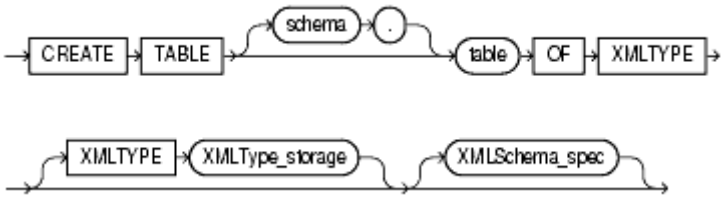
XML belgeleri saklanırken XMLType veri tipinin kullanılması ile iki farklı seçenek izlenebilmektedir. İlk olarak XML belgelerin XMLType kolonları şeklinde,

```
CREATE TABLE Example1
(
KEYVALUE varchar2(10) primary key,
XMLCOLUMN xmltype
);
```

ikincisi XML belgelerin XMLType tablolar şeklinde saklanmasıdır.

```
CREATE TABLE XMLTABLE OF XMLType;
```

XMLType tablolar oluşturulurken aşağıdaki söz diziminden de anlaşılacağı gibi,



XMLType tablo yaratılırken XML şeması da belirtilebilmektedir.

```
CREATE TABLE po_tab OF XMLTYPE
   XMLSCHEMA "http://www.oracle.com/PO.xsd" ELEMENT "PurchaseOrder";
```

Yukarıdaki örnekte XMLType bir tablo yaratılırken, XML şemasının da belirtildiği görülmektedir [10; 11].

XML belgelerinin gerek XMLType tablolara gerekse kolonlara atılmadan önce mutlaka XMLType nesnesine çevrilmesi gerekmektedir. Aşağıdaki kod getCLOBDocument() fonksiyonudur.

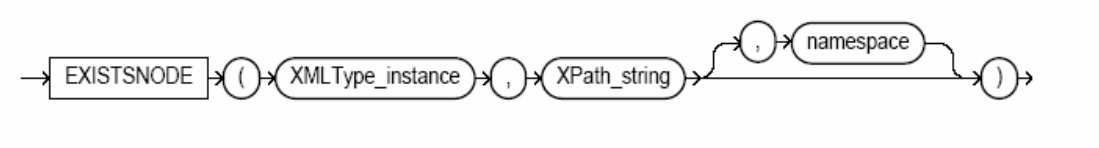
```
create or replace function getClobDocument(
        filename in varchar2,
        charset in varchar2 default NULL)
        return CLOB deterministic
        is
        file bfile := bfilename('DIR',filename);
        charContent CLOB := ' ';
        targetFile bfile;
        lang_ctx number := DBMS_LOB.default_lang_ctx;
        charset_id number := 0;
        src_offset number := 1 ;
        dst_offset number := 1 ;
        warning number;
begin
        if charset is not null then
                charset_id := NLS_CHARSET_ID(charset);
        end if;
                targetFile := file;
                DBMS_LOB.fileopen(targetFile, DBMS_LOB.file_readonly);
                DBMS_LOB.LOADCLOBFROMFILE(charContent, targetFile,
                DBMS_LOB.getLength(targetFile), src_offset, dst_offset,
                charset_id, lang_ctx,warning);
                DBMS_LOB.fileclose(targetFile);
        return charContent;
end;
```

Bu fonksiyonun kullanılması ile XML tablosunda istenilen XML belgeleri saklanabilmektedir.

```
INSERT INTO XMLTABLE
VALUES(XMLTYPE(getCLOBDocument('purchaseorder.xml')));
```

Komutu ile "purchaseorder.xml" belgesi veritabanına eklenebilmektedir. XMLType kolon ve tablolarda bulunan verilere ulaşmak için W3C XPath'i tavsiye etmektedir. Oracle XML veritabanı da XMLType kolon ve tablolardaki verilere XPath kullanarak erişmektedir. XPath gerekli olan XML belgesinin üzerinde gezinerek, belgede bulunan eleman ve özelliklerini araştırır. Bu dolaşım extract (), extractValue() ve existNode fonksiyonları ile sağlanmaktadır.

ExistNode fonksiyonunun söz dizimi;



şeklindedir. existNode metodunu incelemek için aşağıda “purchaseorder.xml” belgesi bulunmaktadır:

```
<PurchaseOrder
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://www.oracle.com/xdb/po.xsd">
<Reference>ADAMS-20011127121040988PST</Reference>
        <Actions>
                <Action>
                        <User>SCOTT</User>
                        <Date>2002-03-31</Date>
                </Action>
        </Actions>
<Reject/>
        <Requestor>Julie P. Adams</Requestor>
                <User>ADAMS</User>
        <CostCenter>R20</CostCenter>
<ShippingInstructions>
        <name>Julie P. Adams</name>
        <address>Redwood Shores, CA 94065</address>
        <telephone>650 506 7300</telephone>
</ShippingInstructions>
<SpecialInstructions>Ground</SpecialInstructions>
<LineItems>
        <LineItem ItemNumber="1">
                <Description>The Ruling Class</Description>
                <Part Id="715515012423" UnitPrice="39.95" Quantity="2"/>
        </LineItem>
        <LineItem ItemNumber="2">
                <Description>Diabolique</Description>
                <Part Id="037429135020" UnitPrice="29.95" Quantity="3"/>
        </LineItem>
        <LineItem ItemNumber="3">
                <Description>8 1/2</Description>
                <Part Id="037429135624" UnitPrice="39.95" Quantity="4"/>
        </LineItem>
</LineItems>
</PurchaseOrder>
```

yukarıdaki satın alma siparişi XML’inde aşağıdaki existNode fonksiyonları çalıştırıldıklarında fonksiyonun geri döndüğü değer true(1) olmaktadır.

```
SELECT existsNode(value(X),’/PurchaseOrder/Reference’)
        FROM XMLTABLE X;

SELECT existsNode(value(X),
                ’/PurchaseOrder[Reference="ADAMS-20011127121040988PST"]’)
        FROM XMLTABLE X;

SELECT existsNode(value(X),
                ’/PurchaseOrder/LineItems/LineItem[2]/Part[@Id="037429135020"]’)
        FROM XMLTABLE X;
SELECT existsNode(value(X),
                ’/PurchaseOrder/LineItems/LineItem[Description="8 1/2"]’)
        FROM XMLTABLE X;
```

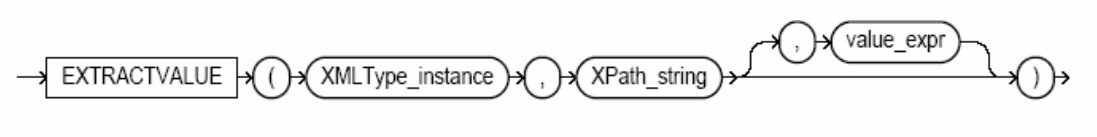
Aşağıdaki existNode fonksiyonları Xpath ifadesi ile herhangi bir düğüm bulamamış olup fonksiyonun geri döndürdüğü değer false(0) olmaktadır.

```
SELECT existsNode(value(X),'/PurchaseOrder/UserName')
        FROM XMLTABLE X;
SELECT existsNode(value(X),
                '/PurchaseOrder[Reference="ADAMS-XXXXXXXXXXXXXXX"]')
        FROM XMLTABLE X;

SELECT existsNode(value(X),
                '/PurchaseOrder/LineItems/LineItem[3]/Part[@Id="037429135020"]')
        FROM XMLTABLE X;
SELECT existsNode(value(X),
                '/PurchaseOrder/LineItems/LineItem[Description="Snow White"]')
        FROM XMLTABLE X;
```

existNode fonksiyonu SELECT, UPDATE, DELETE, SQL cümlelerinde WHERE içinde de kullanılabilmektedir.

```
SELECT count(*)
        FROM XMLTABLE x
        WHERE existsNode(value(x),'/PurchaseOrder[User="ADAMS"]') = 1;

DELETE FROM XMLTABLE x
        WHERE existsNode(value(x),'/PurchaseOrder[User="ADAMS"]') = 1;
```

XMLType olarak saklanan XML belgelerindeki diğer fonksiyon extractValue() fonksiyonudur. ExtractValue fonksiyonunun söz dizimi,



şeklindedir. Geçerli extractValue kullanımı:

```
SELECT extractValue(value(x),'/PurchaseOrder/Reference')
FROM XMLTABLE X;
```

Yukarıdaki satır çalıştırıldığında;

```
EXTRACTVALUE(VALUE(X),'/PURCHASEORDER/REFERENCE')
--------------------------------------------------------------------------
ADAMS-20011127121040988PST
```

bilgisi geri dönmektedir. Geçersiz bir extractValue fonksiyonu ise aşağıdaki gibi değer döndürmektedir.

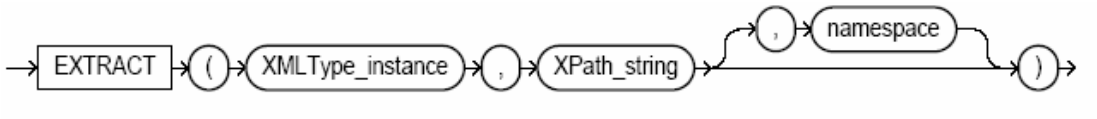
```
SELECT extractValue(value(X),'/PurchaseOrder/LineItems/LineItem/Description')
        FROM XMLTABLE X;

-- FROM XMLTABLE X;
-- *
-- ERROR at line 3:
```

-- ORA-19025: EXTRACTVALUE returns value of only one node

extractValue fonksiyonu da existNode fonksiyonu gibi WHERE'in altında da kullanılabilmektedir. Örnek olarak,

```
SELECT extractValue(value(x),'/PurchaseOrder/Reference')
        FROM XMLTABLE X, SCOTT.EMP
        WHERE extractValue(value(x),'/PurchaseOrder/User') = EMP.ENAME
                AND EMP.EMPNO = 7876;

-- This returns:
-- EXTRACTVALUE(VALUE(X),'/PURCHASEORDER/REFERENCE')
-- ------------------------------------------------
-- ADAMS-20011127121040988PST
```

extract() fonksiyonunun söz dizimi;



şeklindedir. Aşağıdaki extract fonksiyonu uygulamalarında XMLType içerikli XML belgeleri parçaları geri döndürülmektedir.

```
SELECT extract(value(X),'/PurchaseOrder/LineItems/LineItem/Description')
        FROM XMLTABLE X;
-- This returns:
--
EXTRACT(VALUE(X),'/PURCHASEORDER/LINEITEMS/LINEITEM/DESCRIPTIO
N')
-- ------------------------------------------------------------
-- <Description>The Ruling Class</Description>
-- <Description>Diabolique</Description>
-- <Description>8 1/2</Description>
```

XMLType tabloda veya kolonlarda saklanan XML belgeleri updateXML() fonksiyonu sayesinde güncellenebilmektedir. updateXML() fonksiyonunun söz dizimi,

şeklindedir. XML belgesi içerisindeki bir düğümün güncellenmesi aşağıdaki gibi yapılabilmektedir.

```
UPDATE XMLTABLE t
        SET value(t) = updateXML(value(t),'/PurchaseOrder/Reference/text()',
                'MILLER-20020331120000000PST')
WHERE existsNode(value(t),
        '/PurchaseOrder[Reference="ADAMS-20011127121040988PST"]') = 1;
```

Bunlarla birlikte, XML belgesine yeni çocuk düğümler eklemek için insertchildXML(), bir XML düğümüne herhangi bir çeşit çocuk düğümü eklenmesi için appendChildXML(), herhangi bir XML belgesinde bulunan şema tabanlı ve şema tabanlı olmayan bir düğümün silinmesi için deleteXML() fonksiyonları kullanılmaktadır.

Aranılan XML belgelerin daha hızlı bir şekilde bulunabilmesi için dizinleme yöntemleri kullnanılmaktadır. Bunlardan ilki, fonksiyon tabanlı dizinlemedir. Aşağıdaki şekilde fonksiyon tabanlı dizin yaratılması örneği bulunmaktadır.

```
CREATE INDEX city_index ON po_xml_tab
   (poDoc.extract('//PONO/text()').getNumberVal());
```

Diğer bir dizinleme yöntemi, XMLType kolonlar üstünde metin dizinlemesidir. Bu dizinleme yöntemi ile CLOB ve VARCHAR alanlar üzerinde metinsel dizinleme gerçekleştirilmiş olmaktadır. Örnek kullanım aşağıdaki gibidir.

```
CREATE INDEX po_text_index ON
    po_xml_tab(poDoc) indextype is ctxsys.context;
```

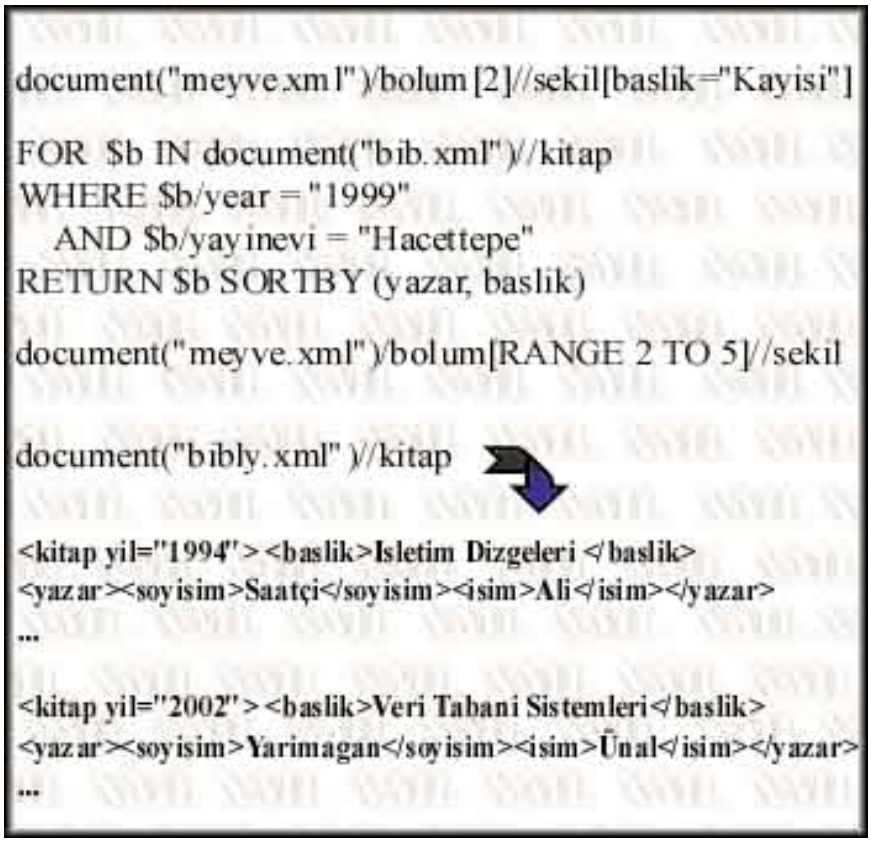
Bir diğer dizinleme yöntemi XMLType kolonlar üstünde XPath dizininin yaratılmasıdır. Bu dizinleme XPath aramalarının hızını artıran etkin bir tarz olarak karşımıza çıkmaktadır. Örnek kullanım,

```
CREATE INDEX po_text_index ON
    po_xml_tab(poDoc) indextype is ctxsys.ctxxpath;
```

şeklindedir. XML belgelerinin sorgulama performansının arttırılması için kullanılan dizinleme yöntemleri [12], ağaç yapısında kullanılan dizinleme ve yol dizinleme yöntemlerine ilişkin araştırmalar mevcuttur [13, 14; 15].

3.3 XQuery Sorgulama Dili

XQuery de XPath veri modelleri gibi XML belgeleri için ağaç tabanlı bir modeldir [16]. XQuery dili XPath dilinden türetilmiştir. XQuery sayesinde XML belgelerinin veri kaynaklarından erişilerek sorgulanması sağlanmaktadır. XQuery'nin tür dizgesi XML şemayı taban alır, aynı zamanda XML tabanlı bir söz dizimine sahiptir. XQuery'nin işlevsel bir dil olması başka deyimlerle birleştirilebilirliği sağlamaktadır. Sorgulama dillerinin özellikle XQuery dili ile ilgili olarak bilginin yeniden edinilmesi amacı ile yapılmış araştırmalar mevcuttur [17; 18].



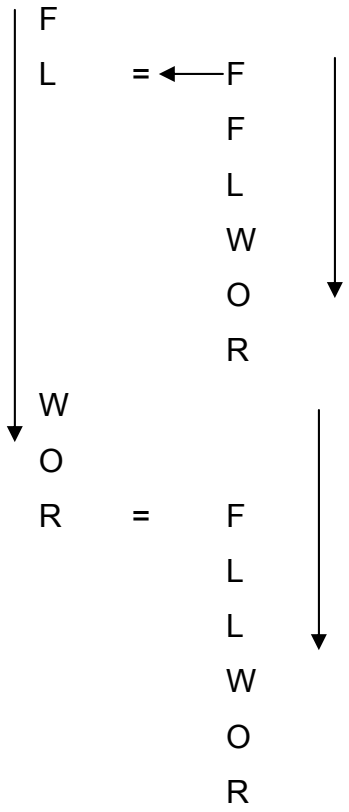
Şekil 3.11: XQuery Örneği.

Yukarıdaki XQuery örneğinden de anlaşılacağı gibi, söz dizimi FLWOR ifadesinden oluşmaktadır. Aşağıda FLWOR ifadesinde yer alan harflerin kullanım şekilleri belirtilmiştir.

- **F**; For $r in doc("…")/data/row,
- **L**; Let $d := doc("…")/data/row/dlay,
- **W;** Where $r/dat = 20000801000000
- **O;** Order by $r/dlay
- **R**; Return <row> { ($r/dat. $r/dlay)} <row>

Şekil 3.11'deki örnekte kitap ve özelliklerinden oluşturulmuş olan bib.xml belgesi içinde yayınevi Hacettepe ve yılı 1999 olan kitaplar yazar ve başlıklarına göre sıralı olarak getirilmesi için sorgulanmıştır. XQuery'deki RETURN kısmı, SQL'deki SELECT kısmı ile, FOR kısmı, SQL'deki FROM kısmı ile benzer özellikler göstermektedir.

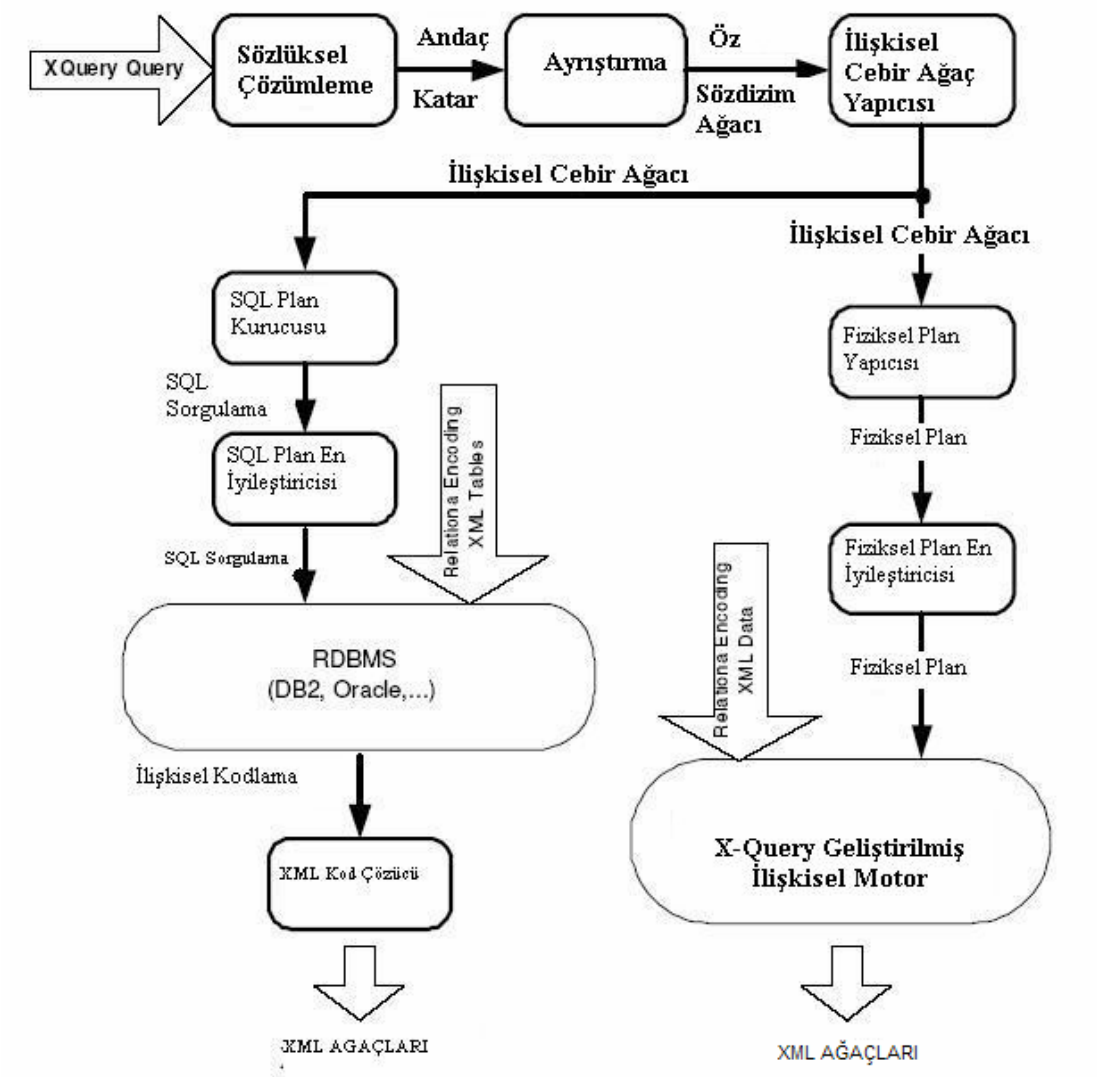
XQuery söz dizimi olan FLWOR, içiçe FLWOR söz dizimlerinden oluşabilmektedir.



FLWOR söz dizimi ile ilgili Dinamik Aralıklar yaklaşımına dayalı olarak FLWOR ifadesi, temel fonksiyonlar ve eleman yapılarını içeren çeşitli araştırmalar da mevcuttur [19].

3.3.1 XQuery'den SQL Sorgusu Oluşturulması

XQuery sorgusu ile XML ağaçlarına ulaşabilmek için iki farklı yöntem bulunmaktadır. Bunlardan birincisi, XQuery sorgu cümlesinin kullanılabildiği XQuery geliştirilmiş ilişkisel motor sayesinde XML ağaçlarına ulaşmaktır. Diğeri ise ilişkisel veritabanında saklanılan XML ağaçlarına ulaşmak için XQuery sorgu cümlesinin SQL cümlesine çevrilmesidir. Her iki durumda da XML ağaçlarına ulaşım mümkün olmaktadır.

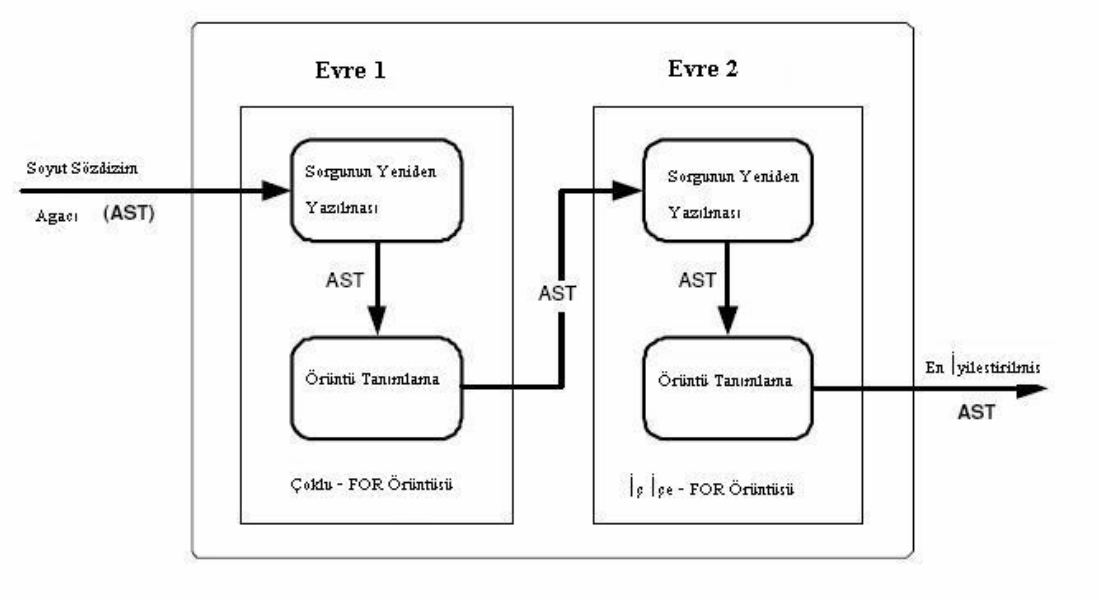


Şekil 3.12: XQuery'den SQL Sorgusunun Oluşturulması [19].

Şekil 3.12'de görüldüğü gibi XQuery sorgusu öncelikle sözcüksel analiz yapıldıktan sonra ayrıştırılarak, ilişkisel ağaç yapısına çevrilir. İlişkisel veritabanından kullanılacak bilgiler için bir SQL plan yapıcısından SQL sorgusu oluşturulur. Oluşturulan SQL sorgusu optimize edilerek ilişkisel veritabanından gerekli sorgulama yapılır, XML çözücüsü ile XML biçimine çevrilir. İlişkisel ağaç fiziksel bir plan yapıcısı ile fiziksel plana çevrilir, fiziksel plan optimizasyonu ile oluşturulmuş olan fiziksel plan iyileştirilmiş ilişkisel motor ile XML biçimine çevrilir.

3.3.2 XQuery Optimizasyonu

XQuery optimizasyonu, XQuery içinde bulunan temel operasyonların birleştirilmesi, FOR ifadelerinin birleştirilmesi, WHERE ifadesinin birleştirilmesi, birinci evrede çoklu FOR kalıbının yeniden yazılması, ikinci evrede ise tekrarlı FOR kalıbının tekrardan yazılması ile yapılmaktadır [20; 21].



Şekil 3.13: XQuery Optimizasyon Şeması

BÖLÜM 4

DURUM ÇALIŞMASI

4.1 Problemin Tanımı

Günümüzde, elektronik belgeler XML tabanlı CM (Content Manager) sayesinde veritabanlarında saklanılarak bu belgeler üzerinden işlem yapılabilmekte, var olan belgelere ek olarak yeni belgeler tanımlanabilmektedir. Böylelikle çok geniş çaplı ve boyutları büyük veritabanlarında çok sayıda XML belgeleri tutulabilmektedir. Bu XML belgelerin sorgulamaları veritabanları için ilave yükler getirebilmekte, aranılan kayıtlara ulaşılmak için fazla zaman harcanmaktadır. Bu harcanan zamanın asgari hale getirilmesi için veritabanlarında dizinleme metodları kullanılmaktadır [25]. Bu tez çalışmasında, farklı iki yöntem ile saklanan belgelerin sorgulama performanslarının karşılaştırılması amacı ile bir uygulama programı yazılmıştır.

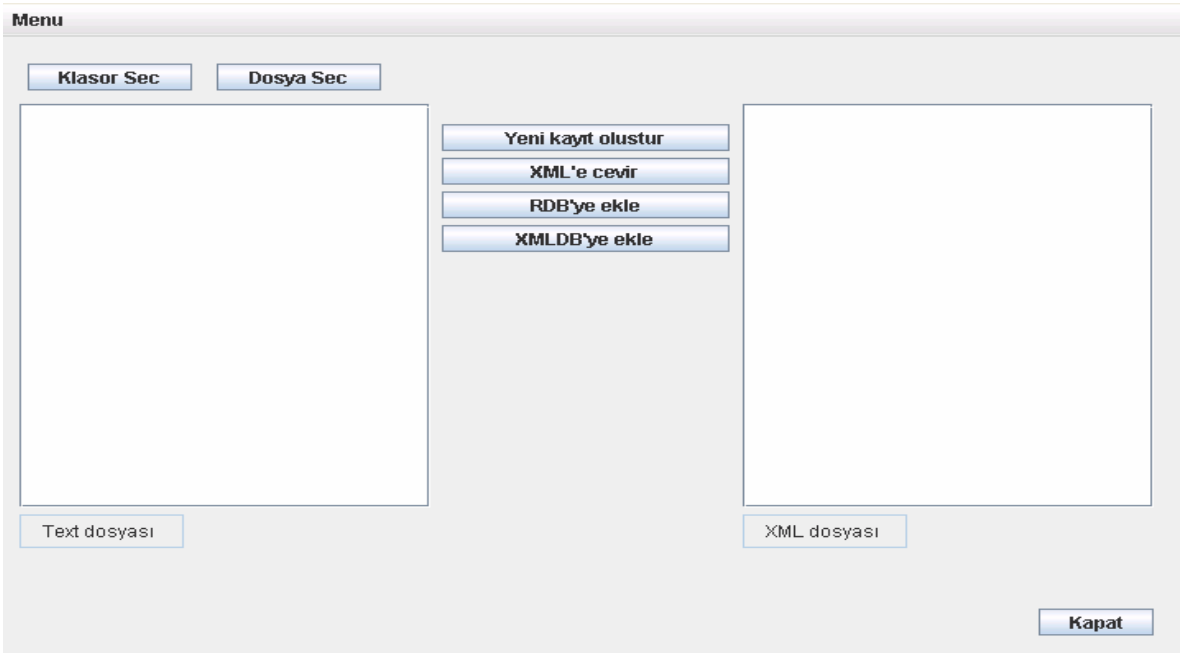
Yazılan uygulama programı, aynı içeriğe sahip olan belgeleri farklı iki biçimde veritabanında saklamaktadır. Sonraki aşamada, çeşitli dizinleme yöntemleri kullanılarak, kullanılan yöntemlerin belgelerin sorgulanmasındaki performansları değerlendirilmiştir.

4.2 Uygulama Yazılımının Kullanılışı

Geliştirilen yazılım, içerik olarak aynı olan bir belgeyi faklı iki biçimde veritabanında saklamaktadır. Bunlardan birincisinde, ilişkisel veritabanında CLOB bir alan içinde belge metin halinde saklanırken ikincisinde ise XML veritabanında belge XMLType alanda XML biçiminde saklanmaktadır. Bunun için öncelikle kullanıcıların giriş yapabildikleri bir arayüz yazılmıştır. Bu arayüz sayesinde belgeler kullanıcılar tarafından veritabanına kaydedilmektedir.

Şekil 4.1: Programın Açılış Menüsü

Program ilk çalıştığında yukarıdaki menü ekrana gelmekte ve program içerisindeki seçenekler sunulmaktadır. Menüde görünen “VT’ye ekle seçimi sayesinde kullanıcı, veritabanında saklamak istediği belgeyi hem XML olarak hem de metin olarak veritabanına ekleyebilmektedir. “Text bul” seçimi ile kullanıcı veritabanında bulunan belgeler arasından istediği kelimeyi arayabilmekte arama sonuçlarını ayrıntılı olarak izleyebilmektedir. “kapat” seçimi ile programdan çıkılmaktadır. “VT’ye ekle” seçimi yapıldığında aşağıdaki veritabanına ekleme ara yüzü ekrana gelmektedir.



Şekil 4.2: Veritabanına ekleme ara yüzü

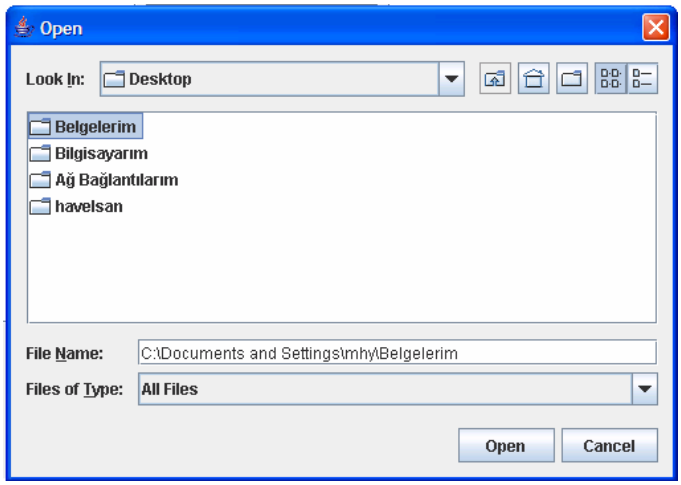
Yukarıda görülen ara yüz sayesinde kullanıcı, metin dosyası olan bir belgeyi XML dosyası haline çevirebilmektedir. Örnek olarak,

Ad : örnek
Soyad: örnek
No : örnek
Derskodu : örnek
Dersadı : örnek
Dersnotu : örnek

Yukarıdaki metin belgesi Şekil 4.2'de metin dosyası bölümünde görüntülenmektedir. XML'e çevir fonksiyonu ile metin belgesi aşağıdaki XML biçimine dönüştürülüp Şekil 4.2'de XML dosyası alanında görüntülenmektedir.

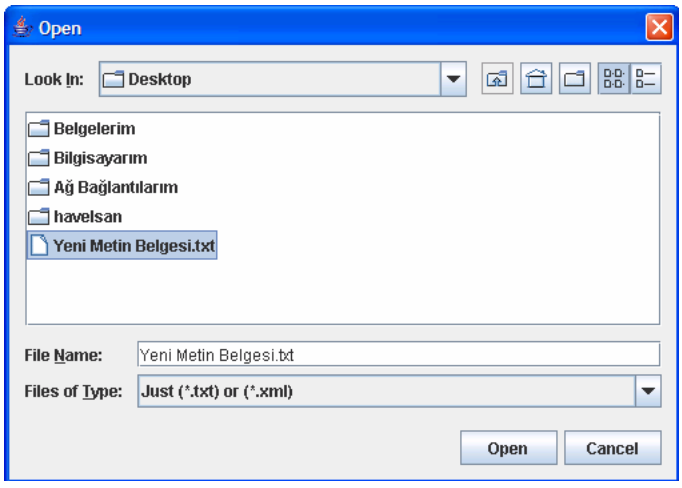
```
<?xml version="1.0"?>
<KISI>
    <ad> örnek</ad>
    <soyad> örnek</soyad>
    <no> örnek</no>
    <derskodu> örnek</derskodu>
    <dersnotu> örnek</dersnotu>
    <dersadi> örnek</dersadi>
</KISI>
```

Kullanıcı “klasör seç” fonksiyonu ile bir klasör içindeki bütün metin ya da XML dosyalarını veritabanına çoklu olarak ekleyebilmektedir. Burada dikkat edilmesi gereken husus XML dosyaları için XML şemalarının aynı olması gerekliliğidir. XML şemalarının farklı olması durumu, XML belgelerinde kullanılan etiketlerin aynı olmaması anlamına gelmektedir. Bu durumda veritabanında yapılan sorgulamalarda hata alınması ihtimali bulunmaktadır. Gerek metin belgelerinin gerekse XML belgelerinin aynı içeriğe sahip olması sorgulama performansının incelenmesinde önemlidir.



Şekil 4.3: Klasör Seç

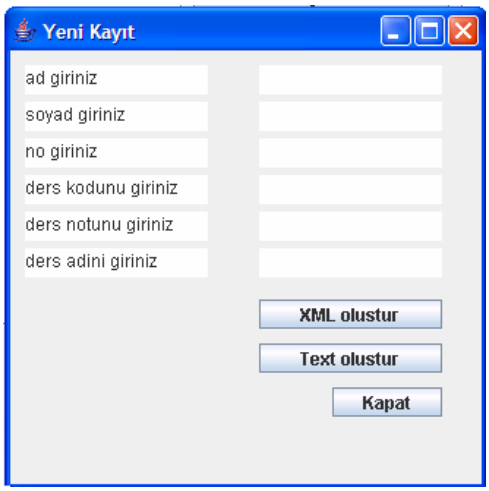
“Dosya seç” fonksiyonu ile seçilen metin belgesi veya XML belgesi veritabanına kayıt edilebilmektedir.



Şekil 4.4: Dosya Seç

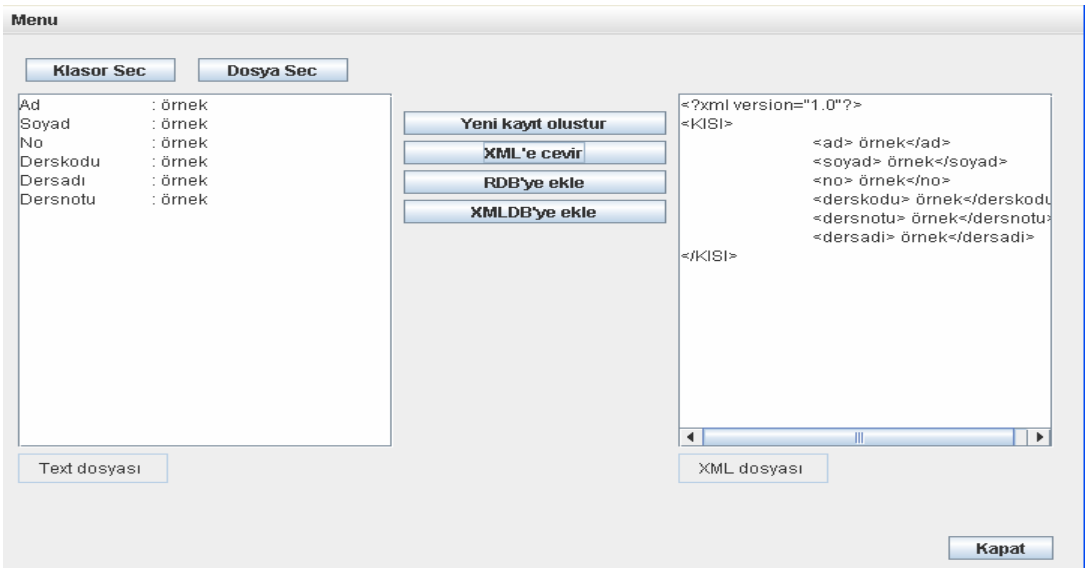
Bununla birlikte kullanıcı tarafından belirlenen şemaya uygun yeni kayıt imkanı da kullanıcıya sunulmuştur. Programın kullandığı örnek şema aşağıdaki gibidir. Dolayısı ile yeni kayıt ekranı Şekil 4.5'de gösterilmiştir.

```
Ad      :
Soyad:
No      :
Derskodu    :
Dersadı     :
Dersnotu    :
```



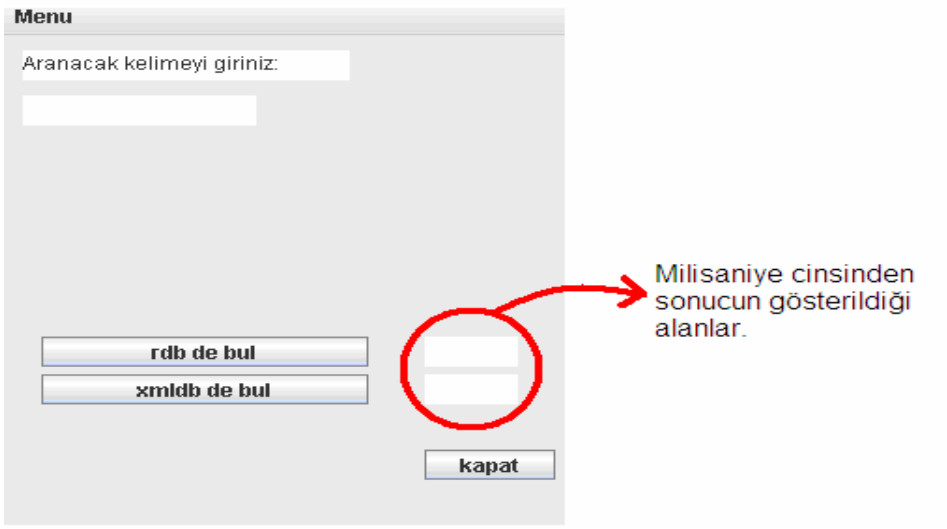
Şekil 4.5: Yeni Kayıt

Metin dosyası ve XML dosyası alanları doldurulduktan sonra, belgeler veritabanına atılabilecek duruma gelmektedirler.



Şekil 4.6: XML'e Çevir fonksiyonu

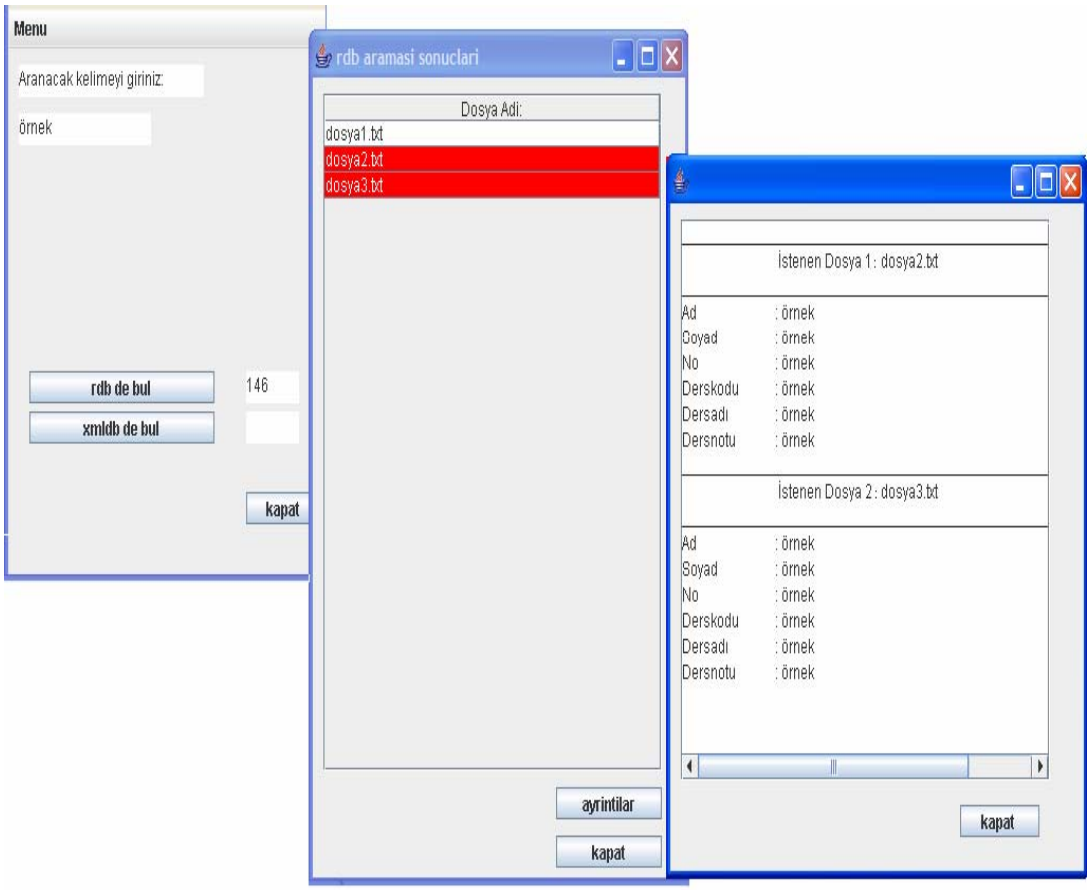
Şekil 4.6'da "RDB'ye ekle" fonksiyonu ile metin dosyası ilişkilendirilmiş veritabanında CLOB alana metin olarak aktarılmaktadır. "XML DB'ye ekle" fonksiyonu ile XML dosyası Oracle XML DB'de XMLType alanına XML olarak aktarılmaktadır. Şekil 4.7'deki Menüden "text bul" fonksiyonunun seçilmesi ile kullanıcıya aşağıdaki ara yüz sunulmaktadır.



Şekil 4.7: Text Bul

Kullanıcı tarafından veritabanında kayıtlı bulunan belgeler arasında yapılacak arama için kelimenin girilmesinden sonra iki sorgulama yöntemi mevcuttur. Bunlardan ilki veritabanında CLOB alanda metin olarak tutulmuş olan belgeler için "rdb de bul" seçimi ile yapılan aramadır. Diğer bir seçenek ise veritabanında

XMLType alanda XML olarak tutulan XML belgeler için “xmldb de bul” ile arama yapmaktır. Her iki sorgulamanın süresi yanlarındaki kutucuklarda milisaniye cinsinden gösterilmektedir. Sorgulamanıın sonuçları Şekli 4.8’de görüldüğü gibi kullanıcıya gösterilmektedir.



Şekil 4.8: Arama Sonuçlarının Gösterilmesi

Yukarıdaki şekilde belirtildiği gibi aranacak kelime “örnek” olarak girilmiş ve RDB’de bul fonksiyonu çalıştırılmıştır. CLOB alanlardaki metin belgeleri içinde yapılan sorgulama sonucu üç adet dosya 146 milisaniyede bulunmuş, kullanıcı dosya2 ve dosya3 adlı dosyaların içeriğini görüntülemek için ayrıntılar fonksiyonunu çalıştırmıştır.

4.3 Performans Analizi

Performans analizinin amacı, veritabanında CLOB alanda metin olarak tutulan belgeler arasında yapılan sorgulamanın hızı ile XMLType alanında XML olarak tutulan belgeler arasında yapılan sorgulamanın hızının, dizinleme kullanımına ve dizinleme çeşitlerine göre karşılaştırılmasıdır.

Performans analizleri, IBM marka Pentium 4 3.0Ghz 1.00GB RAM'e sahip bir masaüstü bilgisayarı, Oracle 9i veritabanı (XMLDB destekli) ve veritabanında 100 ila 1,000,000 arasında değişen sayılarda belgenin kullanılması ile gerçekleştirilmiştir. Performans analizinde kullanılacak belgelerin oluşturulması için veritabanında çişitli tablolar yaratılmış ve yazılan fonksiyonlar ile istenilen sayısıda belgelerin vertabanında tutulması sağlanmıştır. Yapılan işlemlere aşağıda belirtilmiştir.

- Veritabanında "kisiler" tablosu aşağıdaki komut ile oluşturulmuştur.

```
create table kisiler
(
  adi           varchar2(50),
  soyadi        varchar2(50),
  numarasi      number(4)
);
```

- Insert cümleleri ile "kisiler" tablosu doldurulmuştur.

```
insert into kisiler values('ALİ','AYDIN',1);
insert into kisiler values('VELİ','BALLI',2);
insert into kisiler values('AHMET','CAN',3);
insert into kisiler values('MEHMET','DUDEK',4);
insert into kisiler values('HASAN','ERDEM',5);
insert into kisiler values('HÜSEYİN','KAHRAMAN',6);
insert into kisiler values('AYŞE','GÜNDÜZ',7);
insert into kisiler values('NECLA','HAZIR',8);
insert into kisiler values('NEBAHAT','ÇOŞKUN',9);
insert into kisiler values('NEZAHAT','KARA',10);
```

- Veritabanında "ogrenciler" tablosu aşağıdaki komut ile oluşturulmuştur.

```
create table ogrenciler
(
  adi           varchar2(50),
  soyadi        varchar2(50),
```

```
  numarasi    number(4),
  ders_kodu   varchar2(50),
  ders_adi    varchar2(50),
  notu        number(3)
);
```

- Yazılan fonksiyon ile “kisiler” tablosunda bulunan kayıtlar için istenilen sayıda kayıt “ogrenciler” tablosu için otomatik olarak oluşturulmuştur. Ders kodu istenilen şekilde parametrik olarak oluşturulmuştur. Ders notu ise gelişi güzel olarak fonksiyon tarafından verilmiştir.

```
create or replace procedure registerStudent (p_sourceTab varchar2,
p_destTab varchar2, p_stdCount number, p_commitAt number,
p_coursePrefix varchar2) as

/*********************
 Parametrelerin Açıklaması:
 ---------------------------
      1) p_sourceTab: kişi bilgilerinin okunacağı kaynak tablonun adı.
                  Bu tablonun yapısı şu şekilde olmalıdır:

                  create table kisiler
                  (
                    adi           varchar2(50),
                    soyadi        varchar2(50),
                    numarasi      number(4)
                  );

      2) p_destTab: kişi bilgilerinden döngü ile oluşturulacak olan öğrenci
bilgilerinin tutulacağı hedef tablonun adı.
                  Bu tablonun yapısı şu şekilde olmalıdır:

                  create table ogrenciler
                  (
                    adi           varchar2(50),
                    soyadi        varchar2(50),
                    numarasi      number(4),
                    ders_kodu     varchar2(50),
                    ders_adi      varchar2(50),
                    notu          number(3)
                  );

      3) p_stdCount: Oluşturulmak istenen toplam öğrenci sayısı

      4) p_commitAt: Kaç satırda bir COMMIT edileceği
```

5) p_coursePrefix: “ogrenciler” tablosuna atanacak olan ders kodunun başlangıcı. Aşağıdaki fonksiyon, verilen bu koda sıradan sayılar ekleyerek ders kodunu olusturur.

**********/

```
  dyncursor number;
  dynstring varchar2(4000);
  dummy number := -1;

  v_adi varchar2(50);
  v_soyadi varchar2(50);
  v_numarasi number(4);

  v_ders varchar2(50);
  v_randGrade number(3);

  progressCount number(10) := 1;
  dersKodCount number(10) := 1;

begin

  DBMS_RANDOM.INITIALIZE (309666789);

  while progressCount<p_stdCount loop

     dynstring := 'select * from '||p_sourceTab;

     dyncursor := dbms_sql.open_cursor;
     dbms_sql.parse(dyncursor,dynstring,dbms_sql.native);
     dbms_sql.define_column(dyncursor,1,v_adi,50);
     dbms_sql.define_column(dyncursor,2,v_soyadi,50);
     dbms_sql.define_column(dyncursor,3,v_numarasi);

     dummy := dbms_sql.execute(dyncursor);
     loop

        exit when dbms_sql.fetch_rows(dyncursor) = 0;

        dbms_sql.column_value(dyncursor, 1, v_adi);
        dbms_sql.column_value(dyncursor, 2, v_soyadi);
        dbms_sql.column_value(dyncursor, 3, v_numarasi);

        v_ders := p_CoursePrefix||to_char(dersKodCount);

        dbms_random.seed( mod(progressCount*100000,999999)  );

        v_randGrade := dbms_random.value(1,100);
```

```
            insert into OGRENCILER values
            (v_adi,v_soyadi,v_numarasi,v_ders,v_ders,v_randGrade);

             progressCount := progressCount + 1;

             if (mod(progressCount,p_commitAt)=0) then
                 commit;
             end if;

          end loop;

          dersKodCount := dersKodCount + 1;

       end loop;

       DBMS_RANDOM.TERMINATE;

     end;
     /
```

- Aşağıdaki örnek kullanım ile 1.000.000 öğrenci, 10.000 satırda bir işlenilerek 'BİL' kodlu dersler için "kisiler" tablosundan "ogrenciler" tablosuna kayıt oluşturulmuştur.

```
exec registerStudent ('KISILER', 'OGRENCILER', 1000000,10000,'BİL');
```

"ogrenciler" tablosunda istenilen sayıda kayıdın oluşturulmasından sonra aşağıdaki kodlar ile metin ve XML belgeleri hazrlanmış ve veritabanında saklanmıştır. XML belgesi oluşturmak için,

```
CREATE OR REPLACE procedure ORNEK_XML_OLUSTUR as
cursor C1 is select * from ogrenciler ;
c1data c1%rowtype;
cursor_name  INTEGER;
rows_processed  INTEGER;
a utl_file.file_type;
sql_sentence varchar2(3000);
i number;
begin
i:=1;
FOR C1data IN C1 LOOP
```

```
a := utl_file.fopen('C:\client_logs\xml',i||'.xml', 'w');
SQL_SENTENCE := '<KISI>
  <?xml version="1.0"?>
              <ADI>'||C1DATA.adi||'</ADI>
              <SOYADI>'||C1DATA.soyadi||'</SOYADI>
              <OGRENCI_NO>'||C1DATA.numarasi||'</OGRENCI_NO>
              <DERS_KODU>'||C1DATA.ders_kodu||'</DERS_KODU>
             <DERS_ADI>'||C1DATA.ders_adi||'</DERS_ADI>
              <DERS_NOTU>'||C1DATA.notu||'</DERS_NOTU>
</KISI>';
utl_file.put_line(a,sql_sentence);
i:=i+1;
utl_file.fclose(a);
end loop;
COMMIT;
end;
/
```

Metin belgesini oluşturmak için,

```
CREATE OR REPLACE procedure ORNEK_TXT_OLUSTUR as
cursor C1 is select * from ogrenciler ;
c1data c1%rowtype;
cursor_name  INTEGER;
rows_processed  INTEGER;
a utl_file.file_type;
sql_sentence varchar2(3000);
i number;
begin
i:=1;
FOR C1data IN C1 LOOP
a := utl_file.fopen('C:\client_logs\txt',i||'.txt', 'w');
SQL_SENTENCE := 'Adi:'||c1data.adi||chr(10)||
'Soyadi:'||c1data.soyadi||chr(10)||
'Ögrenci_no:'||c1data.numarasi||chr(10)||
'Ders Kodu:'||c1data.ders_kodu||chr(10)||
'Ders Adi:'||c1data.ders_adi||chr(10)||
'Ders Notu:'||c1data.notu;
utl_file.put_line(a,sql_sentence);
i:=i+1;
utl_file.fclose(a);
end loop;
COMMIT;
end;
/
```

Performans analizlerinde metin ve XML belgeleri için veritabanında iki farklı dizinleme kullanılmıştır. Metin belgeleri için “Kelime Dizinleme” kullanılmıştır. Kelime Dizinleme ilişkisel veritabanlarında CLOB ve VARCHAR alanlarda sorgulama performansını arttırmak için kullanılmaktadır. Aşağıdaki komut ile ilgili tablo üzerinde bu dizinleme oluşturulmuştur.

```
CREATE INDEX po_text_index ON
    po_xml_tab(poDoc) indextype is ctxsys.context
```

Kesme dizinleme yönetimi ise XML belgelerinin saklandığı XMLType alanlarda sorgulama performansını arrtırmak amacı ile kullanılmaktadır. Kelime dizinlemeden farklı olarak, alan içindeki kelimenin tamamı değil sadece bir kısmı dizinlenmektedir. Aşağıdaki komut ile ilgili tablo üzerinde kesme dizinleme oluşturulmuştur.

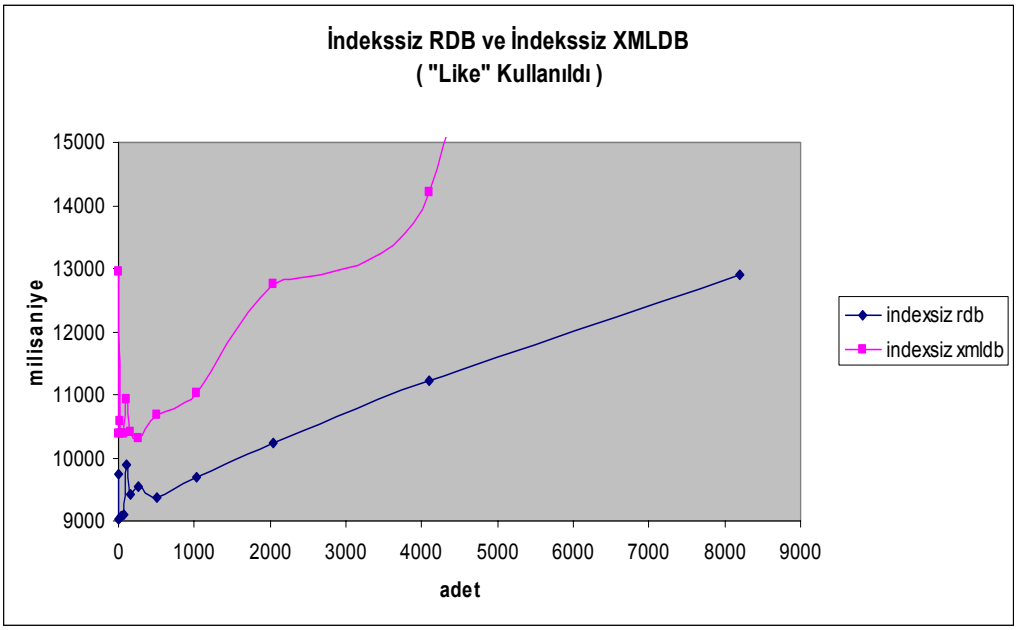
```
CREATE INDEX xmldb_index ON t_kisi_xml t
      (SUBSTR
            (XMType.getStringVal
                  (XMLType.extract
                        (t.KISI, ‘/tag/text()’)),1,5));
```

Performans analizi sonucunda,

- Metin belgesi ve XML belgeleri için dizinleme kullanılmadan yapılan sorgulama sonucu aşağıdaki performans değerleri elde edilmiştir.

Dosya Sayısı	CLOB Alanda Arama Süresi (ms.)	XMLType Alanda Arama Süresi(ms.)
1	9732	10389
6	9015	12951
16	8986	10580
36	9063	10391
66	9094	10844
106	9890	10922
156	9422	10407
256	9547	10312
512	9360	10671
1024	9703	11031
2048	10235	12750
4096	11234	14203
8192	12906	34718

Tablo 4.1: Metin ve XML Belgeleri için Dizinleme Kullanılmadan Yapılan Sorgulama



Şekil 4.9: Metin ve XML Belgeleri için Dizinleme Kullanılmadan Yapılan Sorgulama

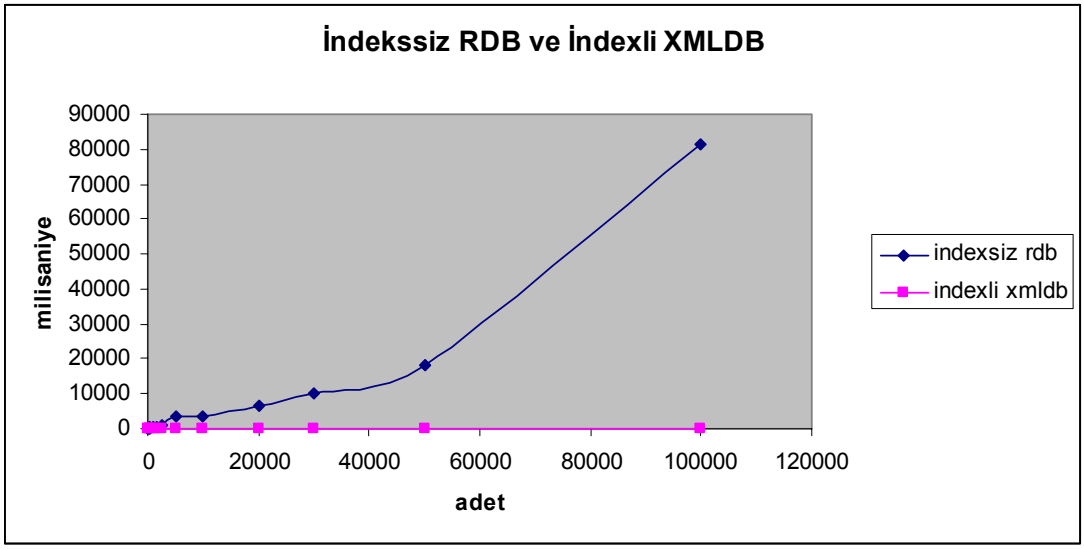
Yukarıdaki şekilde yatay eksende dosya sayısı, dikey eksende ise sorgulama sonucu milisaniye olarak gösterilmiştir. Bununla birlikte mavi eğri dizinleme

kullanılmayan metin belgelerinin, pembe eğri ise dizinleme kullanılmayan XML belgelerinin sorgulama performansını göstermektedir. Şekil 4.9'de dikkat çeken, aramada like kullanılması sonucu herhangi bir dizinleme kullanılmadan XML belgelerinin sorgulanma sürelerinin metin belgelerinden daha fazla olmasıdır. Metin belgeleri için dosya sayısının artması ile sorgulama performansı arasında doğrusal bir ilişki bulunurken, bu ilişkinin XML belgeleri için üstsel olduğu gözlemlenmiştir.

- Metin belgelerinde herhangi bir dizinleme olmaksızın XML belgeleri için kesme dizinleme kullanılarak yapılan sorgulama sonucu aşağıdadır.

Dosya Sayısı	**CLOB Alanda Arama Süresi (ms.)**	**XMLType Alanda Arama Süresi(ms.)**
100	219	16
200	250	16
400	344	15
1000	500	16
1600	672	16
2500	985	1
5000	3672	16
10000	3391	16
20000	6641	16
30000	10344	15
50000	17954	16
100000	81266	125

Tablo 4.2: Metin Belgeleri için Dizinleme Kullanılmamış XML belgeleri için Kesme Dizinleme Kullanılmış Sorgu



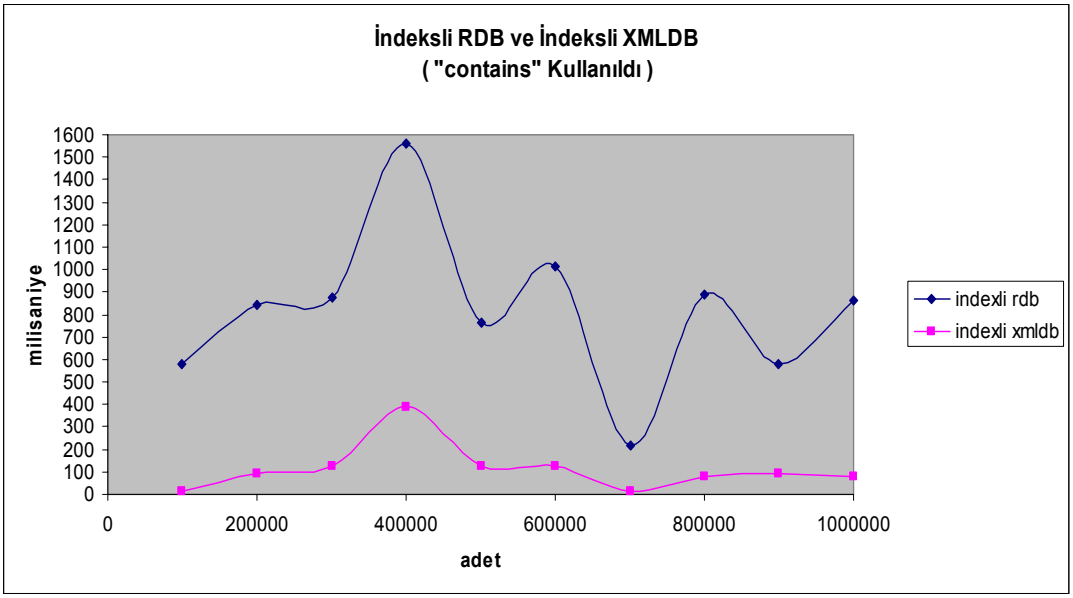
Şekil 4.10: Metin Belgeleri için Dizinleme Kullanılmamış XML belgeleri için Kesme Dizinleme Kullanılmış Sorgu

XML belgeleri için kullanılan kesme dizilemenin sorgulama hızını özellikle yüksek dosya sayısında etkin bir şekilde artırdığı görülmektedir. Dizinleme kullanılmayan metin belgeleri için dosya sayısının artması sonucunda sorgulama performansının olumsuz olarak etkilenmesi normaldir. Dizinleme kullanılan XML belgeleri için dosya sayısının artması herhangi bir şekilde sorgulama performansını etkilememiştir. Bu sonuç, dizinleme yöntemlerinin kullanılmasının sorgulama performansı üzerinde çok büyük bir etkisinin olduğunu göstermektedir.

- Metin belgeleri için kelime dizinleme yapılırken, XML belgeleri için kesme dizinleme kullanılmış ve aşağıdaki performans sonuçları elde edilmiştir.

Dosya Sayısı	CLOB Alanda Arama Süresi (ms.)	XMLType Alanda Arama Süresi(ms.)
100000	578	16
200000	843	93
300000	875	125
400000	1563	391
500000	766	125
600000	1016	125
700000	218	16
800000	891	78
900000	578	94
1000000	860	78

Tablo 4.3: Metin Belgeleri için Kelime Dizinleme, XML Belgeleri için Kesme Dizinleme Kullanılmış Sorgu



Şekil 4.11: Metin Belgeleri için Kelime Dizinleme, XML Belgeleri için Kesme Dizinleme Kullanılmış Sorgu

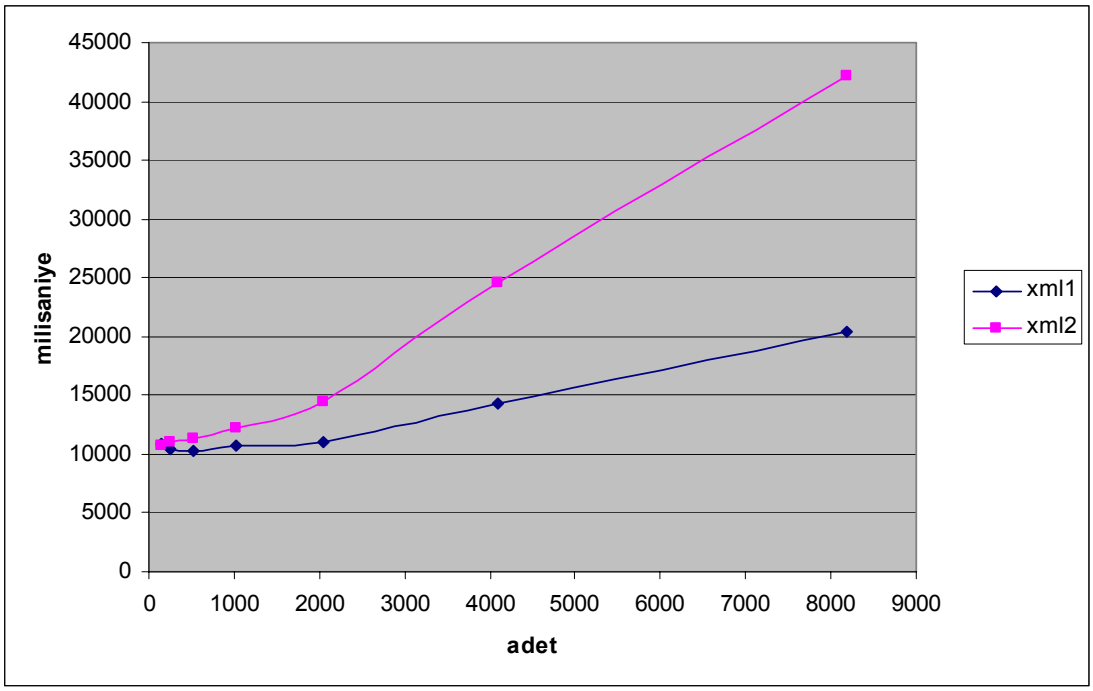
Yukarıdaki tablolarda da gözlemlendiği gibi kelime dizinleme metodu kullanılan metin belgeleri için sorgulama performansı dosya adedine göre farklılık göstermesine rağmen kesme dizinleme kullanılan XML begeleri için sorgulama

performansı çok yüksek adetlerde dosya olmasına rağmen çok farklılaşmamış metin belgelerindeki arama performansına göre çok daha hızlı ve etkin olduğu görülmüştür.

- XML belgeleri içinde bulunan etiketlerin karmaşıklığının artırılması ve iç içe olarak kullanılmasının sorgulama performansını ne şekilde değiştireceği araştırılmış, sonuçları aşağıda belirtilmiştir.

Dosya Sayısı	xml1 (ms.)	xml2 (ms.)
250	10240	10423
500	10116	11704
1000	10780	12765
2000	11012	14886
4000	14790	24939
8000	20214	42568

Tablo 4.4: Basit Etiket içeren XML belgesi ile Karmaşık Etiketler içeren XML Belgelerinin Sorgu Performansları



Şekil 4.12: Basit Etiket içeren XML belgesi ile Karmaşık Etiketler içeren XML Belgelerinin Sorgu Performansları

Şekil 4.12'de xml1 olarak gösterilen eğri XML belgeleri basit etiketler içermekte olup, xml2 olarak gösterilen eğri ise karmaşık etiket yapısına sahip ve iç içe etiketler kullanılmış XML belgelerinin sorgu performanslarıdır. Yukarıdaki tabloda görülmektedir ki, XML belgeleri içindeki etiket karmaşıklığı ve iç içe kullanım arttıkça dosya adedinin de artması ile sorgulama süreleri de artmaktadır.

Performans analizlerinin sonunda iki önemli sonuç ortaya çıkmıştır. Birincisi, dizinlemenin kullanılması, sorgulamanın performansında çok olumlu yönde etkisi bulunmaktadır. XML biçiminde tutulan belgeler için kesme, metin biçiminde tutulan belgeler için kelime dizinleme yöntemleri uygulanmıştır. Diğer bir sonuç ise kesme dizinlemenin kelime dizinlemeye oranla daha etkin bir performans sergilediğidir.

BÖLÜM 5

SONUÇ VE GELECEKTE YAPILACAKLAR

Bu tez çalışması ile günümüzün popüler işaretleme dili olan XML hakkında geniş ve kapsamlı bir çalışma yapılmıştır. Bu işaretleme dili ile klasik ilişkisel veritabanları ve XML'e uygun yeni nesil veritabanları karşılaştırmaları, mevcut yeteneklere ilave olarak getirilen yenilikler, uygulama geliştirilmesinde kılavuz olarak kullanılmıştır. Günümüzde her alanda kullanılan XML belgelerinin veritabanında saklanma şekilleri ve üzerlerinde sorgulama performansını etkileyecek dizinleme yöntemleri ortaya konulmuştur.

Tez çalışması kapsamında dizinleme yöntemlerinin sorgulama performansını doğrudan ve olumlu yönde etkilediği görülmüştür. XML verileri için farklı dizinleme yöntemleri ortaya çıkmaktadır. Bunlar;

- Birincil Dizinleme
- İkincil Dizinleme
    - Yol (Path) Dizinleme
    - Özellik (Property) Dizinleme
    - Değer (Value) Dizinleme
    - Kelime Dizinleme
    - Kesme Dizinlemedir

XML içerikli sütunların dizinlenebilmesi için gerekli ön koşullar mevcuttur. İlki, XML içerikli sütunun bulunduğu tabloda birincil anahtar (Primary Key) olarak tanımlanmış bir sütunun bulunması gerekmektedir.

```
CREATE TABLE tbKisiDetay
        (kisiId INT PRIMARY KEY,
        xKisi XML)
```

kodu ile tablo oluşturulur. Tablo incelendiğinde birincil anahtar olarak kisiId bulunmaktadır. Bu anahtar kişinin diğer bilgileri ile bir ilişki kurmak amacı ile oluşturulmuştur. xKisi sütununda XML verisi olarak her bir satıda farklı kişilere ait bilgiler saklanmaktadır. Tablo tanımından sonra birincil XML dizinlemesi tanımlanabilir. Bu tanımlanan birincil XML dizininin görevi birincil anahtar ile XML'deki düğümler arasında bir bağlantı kurmaktır. Bu nedenle bu tür dizinleme için "düğüm tablosu" deyimi kullanılmaktadır. Dizinleme özellikle doğaçlama sorgular için hızlandırıcı bir özelliğe sahiptir.

```
CREATE PRIMARY XML INDEX indXMLPRM ON tbKisiDetay(xKisi)
```

kodu ile dizin tanımlanır.

XML sütunu üzerinde birincil dizinleme yapıldıktan sonra ikincil dizin tanımlanabilmektedir. Bunlardan yol (path) dizinlemesi; FOR PATH ifadesinin kullanılması ile tanımlanır. Bu dizinleme, XQuery ifadelerinin yol tarama işlemlerinin daha hızlı bir şekilde çalışmasını sağlamaktadır. Yukarıda oluşturulan örnek tablo üzerinde yol dizinlemesi yapılacak olursa;

```
CREATE XML INDEX indXpath ON tbKisiDetay (xKisi)
    USING XML INDEX indXMLPRM FOR PATH
```

kodu ile tanımlanır. Bu dizin için ve takip eden diğer ikincil dizinler için daha öncesinde tanımladığımız birincil dizin kullanılmaktadır.

```
SELECT xkisi.query ( 'kisi//ad' )
    FROM tbKisiDetay
```

Sorgusunda tanımlanan dizin etkin olarak kullanılır. Oracle da ise benzer özelliklere sahip olan yol dizinlemesi;

```
CREATE INDEX xml_idx ON xml_tab(col_xml) indextype is
ctxsys.CTXXPATH;
```

veya

```
CREATE INDEX xml_idx ON xml_tab(col_xml) indextype is
ctxsys.CTXXPATH
PARAMETERS('storage my_storage memory 40M');
```

şeklinde çeşitli parametreler verilerek tanımlanmaktadır. Bu dizinleme existsNode() fonksiyonunun kullanıldığı sorgulamalarda performansı büyük ölçüde arttırmaktadır.

```
SELECT xml_id FROM xml_tab x WHERE
        x.col_xml.existsnode('/book/chapter[@title="XML"]')>0;
```

```
exec ctx_ddl.sync_index('xml_idx');
```

kodu ile tanımlanan dizin senkronize edilir.

```
exec ctx_ddl.optimize_index('xml_idx', 'FAST');
        veya
exec ctx_ddl.optmize_index('xml_idx', 'FULL');
```

kodu ile en iyileştirilmektedir.

```
ANALYZE INDEX myPathIndex COMPUTE STATISTICS;
```

yukarıdaki komut ile tanımlanan dizin analiz edilmektedir.

Değer dizinleme; FOR VALUE ile tanımlanır. XQuery hızlandırmak için kullanılır, özellikle xml.exist() ile bir değerin varlığı kontrol edilirken bu tür bir dizinleme performansı olumlu yönde etkilemektedir.

```
CREATE XML INDEX indXvalue ON tbKisiDetay(xKisi)
USING XML INDEX indXMLPRM FOR VALUE
```

şeklinde tanımlanır. Özellik dizinleme; FOR PROPERTY ifadesi ile tanımlanır. Özellikle [@xxx = “yyy”] şeklinde öznitelik sorgulayan XQuery sorgularını hızlandırmak için kullanılırlar.

```
CREATE XML INDEX indXprop ON tbKisiDetay(xKisi)
USING XML INDEX indXMLPRM FOR PROPERTY
```

şeklinde tanımlanır.

```
SELECT xkisi.query ( '
        for $s in
        kisi[@AD = "AHMET"]//adi
        where $s/@yas >= 21
        return <adres>{data($s/adres)}</adres>
' )
FROM tbKisiDetay
```

Yukarıdaki sorgu örneğinde hem değer hem de özellik dizinlemeleri etkin olarak kullanılmaktadır.

Yukarıdaki dizinleme metodlarının yanı sıra XML sütunları üzerinde tam metin dizinlemesi de yapılabilmektedir. Metin aramalarında etkin bir yol olan tam metin (full_text) aramalarının performanslarının arttırılmasında tam metin dizinleme kullanılmaktadır. Birincil anahtarın gerçekleşmesi için otomatik olarak oluşturulan PK__tbKisiDetay__1A14E395 tam metin dizininin oluşturulmasında kullanılmaktadır.

```
CREATE FULLTEXT CATALOG ft AS DEFAULT
CREATE FULLTEXT INDEX ON dbo.tbKisiDetay ( xkisi )
    KEY INDEX PK__tbKisiDetay__1A14E395
```

Yukarıdaki komut ile dizin tanımlandıktan sonra tanımlanan dizin üzerinden CONTAINS sorgusu aşağıdaki şekilde gerçekleştirilmektedir.

```
SELECT R.xKisi.query('//adres')
FROM ( SELECT kitapId , xKisi
         FROM tbKisiDetay
         WHERE contains ( xKisi,'AHMET' ))
       R ( kisiId , xKisi )
```

Yine Oracle da benzer özellikte bir metin sorgulaması;

```
select * from xmltab x where
  existsNode(value(x), '/book/chapter[ora:contains(body,"dog OR cat")>0]',
  'xmlns:ora="http://xmlns.oracle.com/xdb"') = 1;
```

şeklinde gerçekleşmektedir. Tam metin arama için LOB alanlar üstünde oluşturulan tam metin dizinleri aynı zamanda XML sütunlar içinde oluşturulabilmektedir.

Kelime dizinleme yöntemi ile metin alanlarda yapılan sorgulamanın performansının olumlu yönde etkilendiği görülmektedir. Bununla birlikte XML olarak saklanan belgeler için kullanılan kesme dizinleme yönteminin, dizinleme yöntemi kullanılmamasına oranla sorgulama performansını çok hızlandırdığı ortaya konulmuştur. Kesme dizinleme yöntemi kelime dizinleme yöntemine oranla daha etkin bir sorgulama performansı sergilemiştir.

Yukarıda da belirtilen farklı dizinleme yöntemleri üzerinde detaylı bir çalışma yapılmıştır. Uygulama programında farklı dizinleme yöntemlerinin XML belgelerinin sorgulanmasında çeşitli performans değerleri aldığı gözlemlenmiştir. İlişkisel veritabanlarında bulunmayan ve XML'e özgü çeşitli fonksiyonlar eklenmesi ile alışılagelmiş veritabanlarından türetilmiş olan XML veritabanlarında tutulan, XML belgelerinin sayısının artması ve yapısının daha da karmaşıklaşması sonucunda sorgulama performanslarının olumsuz yönde etkilendiği anlaşılmıştır.

Bu çalışmada aynı içeriği sahip farklı yöntemlerle veritabanında tutulan belgelerin sorgulanmasındaki dizinleme performansları ortaya konulmuştur. Aynı zamanda çalışma, XML veritabanları ile uygulama geliştirilirken bir kılavuz niteliği taşımaktadır. Gelecekte, gerek sayıca gerekse yapısal olarak meydana gelebilecek teknolojik gelişmeler sonucunda, sorgu performanslarını en üst düzeye çıkartabilecek yeni dizinleme yöntemleri de eklenerek daha ayrıntılı performans ölçümlerinin uygulanabilirliği hedeflenmiştir.

KAYNAKLAR

[1]. SGML Users' Group. (1990). A Brief History of the Development of SGML,http://www.sgmlsource.com/history/sgmlhist.htm (15 Ekim 2006,WEB)

[2]. Cho SungRan, Nick Koudas, Divesh Srivastava.(2006). International Conference on Management of Data Proceedings of the 2006 ACM SIGMOD international conference on Management of data Chicago, IL, USA SESSION: Query processing for XML data Pages: 455 – 466.

[3]. Chung Chin-Wan, Jun-Ki Min, Kyuseook Shim.(2002). International Conference on Management of Data Proceedings of the 2002 ACM SIGMOD international conference on Management of data Madison, Wisconsin SESSION: Research sessions: path indexing Pages:121 – 132.

[4]. Du, Fengdong.(2004). "Moving From XML Documents to XML Databases", The Univeristy of Colombia.

[5]. Mathis, Christian, Theo Härder.(2005)."A Query Processing Approach for XML Database Systems",University of Kaiserslautern.

[6]. Altıntaş Kemal, Gökçen Tuğsuz, "Query Mechanisms for a System to Store XML Documents in Relational Databases"

[7]. Malhotra, Rakesh.(August 2001). "XML Database Engines", University of Georgia Master of Science, Athens, Georgia.

[8]. Norbert Fuhr, Kai Großjohann.(April 2004). "An XML Query Language Based on Information Retrieval Concepts", University of Duisburg-Essen, USA.

[9]. Xinc Guangming, Java Esanakula, Suresh Jayanty.(2005). ACM Southeast Regional Conference Proceedings of the 43rd annual southeast regional conference - Volume 1 Kennesaw, Georgia SESSION: Database Pages: 218 – 219.

[10]. Murthy, Ravi, Sandeepan Banarjee.(2003)."XML Schemas in OracleXML DB", Oracle Corporation, USA.

[11]. XML Technology Center.(2006). http://www.oracle.com/technology/tech/xml/index.html, (update 9 November 2006).

[12]. Wang Haixun, Sanghyun Park and et al.(2003). International Conference on Management of Data Proceedings of the 2003 ACM SIGMOD international conference on Management of data San Diego, California SESSION: XML indexing and compression Pages: 110 – 121.

[13]. Zou Qinghua, Shaorong Liu, Wesley W.Chu.(2004). Workshop On Web Information And Data Management Proceedings of the 6th annual ACM international workshop on Web information and data management Washington DC, USA SESSION: XML processing Pages: 39 – 46.

[14]. Hammerschmidt, B.C. Kempa, M. Linnemann, V.(05-08 April2005). Data Engineering Workshops, 21st International Conference on, Autonomous Index Optimization in XML Databases, On page(s):1217, Germany.

[15]. Min, Jun-Ki, Jongik Kim, Miyoung Lee.(21-23 Feb. 2005). Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on, Effective path indexes for XML data on relational databases, Volume: 2 On page(s): 1355 – 1359, South Korea.

[16]. XQuery 1.0 and XPath 2.0 Data Model (W3C working Draft).(12/11/2003). Available at http://www.w3.org/TR/Xpath-datamodel. (25 Ağustos 2006).

[17]. Norbert Fuhr, Kai Großjohann, Sascha Kriewel.(Springer 2003). “A Query Language and User Interface for XML Information Retrieval”, University of Duisburg- Essen, USA.

[18]. Norbert Fuhr.(2001). XML Information Retrieval and Information Extraction, University of Dortmund, Germany.

[19]. Chen, Yiengwen.(2004). XQuery Query Processing in Relational Systems, University of Waterloo, Master of Mathematics in Computer Science, Canada.

[20]. Muralidhar Krishnaprasad, Zhen Hua Liu, and et al. (2004). “Query Rewrite for XML in Oracle XML DB”, Oracle Corporation, USA.

[21]. Che Dunren, Karl Aberer, Tamer Ozsu.(September 2006). "Query Optimization in XML Structured- Document Databases", The VLDB Journal -The International Journal on Very Large Data Bases Volume 15, Issue 3 Pages: 263– 289.

[22]. XQuery 1.0 : An XML Query Language. Available at http://www.w3.org/TR/Xquery. (24 Haziran 2006).

[23]. Borland Software Corporation.(2003). Developing Database Applications, Version 9, USA.

[24]. Banerjee, Sandeepan. (February 2003). Oracle XML DB vs.IBM DB2 XML Extender, "An Oracle Technical White Paper", USA.

[25]. Oracle9i XML Database Developer's Guide - Oracle XML DB Release 2 (9.2) Part Number A96620-01, http://www.lc.leidenuniv.nl/awcourse/oracle/index.htm. (08 Kasım 2006).

[26]. Drake, Mark.(May 2005). "Oracle Database 10G Release 2 XML DB", An Oracle Technical White Paper, USA.

[27]. Vakali Athena, Barbara Catania and Anna Maddalena.(March-April 2005). "XML Data Stores: Emerging Practices", Published by IEEE Computer Society, 1089-7801/05/2005 IEEE.

[28]. Chi-Hsing Chu, Chien-Hsuan Huang, Michael Lee.(2000). "Building a XML-Based Unified User Interface System under J2EE Architecture", 0-7695-0933-9/00/2000 IEEE, Taiwan.

[29]. Damien K. Fisher Franky Lam William M. Shui Raymond K. Wong.(2003). "Efficient Ordering for XML Data", University of New South Wales Sydney, Australia.

[30]. Shurug Al-Khalifa, Cong Yu, H. V. Jagadish.(2003). "Querying Structured Text in an XML Database", SIGMOD 2003, June 9-12, Copyright 2003 ACM 1-58113-634-X/03/06, San Diego, CA.

[31]. H. V. Jagadish, Shurug Al-Khalifa and et al.(19 December 2002). "A Native XML Database", The VLDB Journal (2002) 11: 274–291 / Digital Object Identifier (DOI) 10.1007/s00778-002-0081-x, Springer- Verlag, USA.

[32]. Cole, Timothy W.,Thomas G. Habing.(23/24 October 2002). XML Tutorial, Colorado Alliance of Research Libraries, University of Illinois at UC.

KAYNAK OLARAK KULLANILAN WEB SİTELERİ

[33]. http://xml.apache.org

[34]. http://technet.oracle.com/tech/xml/

[35]. http://msdn.microsoft.com/xml/

[36]. http://java.sun.com/xml/

[37]. http://www.w3.org/XML/

[38]. http://www.ibm.com/developer/xml/

[39]. http://www.oracle.com/xml/

[40]. http://www.xml.org/

[41]. http://www.xml101.com

[42]. http://www.sun.com/software/xml/

[43]. http://www.xml.com/

[44]. http://www.xml.org/

[45]. http://www.xmlinfo.com/

EKLER

EK A

A.1 Performans Analizinde Kullanılan Metin Belgesi

Önceki dokümanda TCP/IP'nin ne olduğunu ve adreslemenin nasıl yapıldığı anlatılmıştı, paketlerin internet üzerinde yollarını nasıl bulduklarını ise bu dokümanda anlatacağız.Buradaki IP numarası ve subnet maskesi hakkında detaylı bilgi önceki dokümanlarda verilmştir. Öncelikle bir TCP/IP networküne bağlanmak isteyen bir bilgisayarda yapılması gereken ayarları ele alalım. TCP/IP protokolü tanıtılırken bir takım bilgilere ihtiyaç duyulmaktadır, bunlarDNS sunucuları ise daha sonra açıklanacaktır. Bu kısımda özellikle üzerinde durulacak olan "ağ geçidi" kavramı ve internet üzerinde yönlendirmedir. Buradan hareketle yukarıdaki bilgilere sahip bir bilgisayarın yapacaklarını özetleyelim, ilk olarak daha önce anlatıldığı şekilde IP adresini ve subnet maskesini kullanarak ait olduğu networkü bulacaktır. Daha sonra kendisi için bir IP yönlendirme tablosu oluşturacaktır. Yönlendirme tablosu oluştururken izleyeceği yol şudur: KendiIP numarasına giden tüm paketleri kendisine yönlendirecektir. Aynı şekilde kendisi ile aynı networkte olan adreslere bu IP numarasının ait olduğu arayüz üzerinden ulaşmaya çalışacaktır. Bu network dışında kalan adreslere yönelen tüm paketleri...

A.2 Performans Analizinde Kullanılan Basit Etiketli XML Belgesi

<?xml version="1.0"?>
<tag>
Önceki dokümanda TCP/IP'nin ne olduğunu ve adreslemenin nasıl yapıldığı anlatılmıştı, paketlerin internet üzerinde yollarını nasıl bulduklarını ise bu dokümanda anlatacağız.Buradaki IP numarası ve subnet maskesi hakkında detaylı bilgi önceki dokümanlarda verilmştir. Öncelikle bir TCP/IP networküne bağlanmak isteyen bir bilgisayarda yapılması gereken ayarları ele alalım. TCP/IP protokolü tanıtılırken bir takım bilgilere ihtiyaç duyulmaktadır, bunlarDNS sunucuları ise daha sonra açıklanacaktır. Bu kısımda özellikle üzerinde durulacak olan "ağ geçidi" kavramı ve internet üzerinde yönlendirmedir. Buradan hareketle yukarıdaki bilgilere sahip bir bilgisayarın yapacaklarını özetleyelim, ilk olarak daha önce anlatıldığı şekilde IP adresini ve subnet maskesini kullanarak ait olduğu networkü bulacaktır. Daha sonra kendisi için bir IP yönlendirme tablosu oluşturacaktır. Yönlendirme tablosu oluştururken izleyeceği yol şudur: KendiIP numarasına giden tüm paketleri kendisine yönlendirecektir. Aynı şekilde kendisi ile aynı networkte olan adreslere bu IP numarasının ait olduğu arayüz üzerinden ulaşmaya çalışacaktır. Bu network dışında kalan adreslere yönelen tüm paketleri...

</tag>

A.3 Performans Analizinde Kullanılan Karmaşık ve İç içe Etiketlere Sahip XML Belgesi

```
<?xml version="1.0"?>
<tag>
	<tag1>
		<tag6>
			<tag7>
				<tag8>
					<tag9>
Önceki dokümanda TCP/IP'nin ne olduğunu ve adreslemenin nasıl yapıldığı anlatılmıştı, paketlerin internet üzerinde yollarını nasıl bulduklarını ise bu dokümanda anlatacağız.
					</tag9>
				</tag8>

			</tag7>
			<tag13>
Buradaki IP numarası ve subnet maskesi hakkında detaylı bilgi önceki dokümanlarda verilmiştir,
			</tag13>
			<tag14>
				<tag15>
Öncelikle bir TCP/IP networküne bağlanmak isteyen bir bilgisayarda yapılması gereken ayarları ele alalım. TCP/IP protokolü tanıtılırken bir takım bilgilere ihtiyaç duyulmaktadır, bunlar
				</tag15>

			</tag14>
		</tag6>
		<tag10>
DNS sunucuları ise daha sonra açıklanacaktır. Bu kısımda özellikle üzerinde durulacak olan "ağ geçidi" kavramı ve internet
		</tag10>
		<tag11>
üzerinde yönlendirmedir. Buradan hareketle yukarıdaki bilgilere sahip bir bilgisayarin yapacaklarını özetleyelim
		</tag11>
		<tag12>
özetleyelim, ilk olarak daha önce anlatıldığı şekilde IP adresini ve subnet maskesini kullanarak ait olduğu networkü bulacaktır.
		</tag12>
	</tag1>
	<tag2>
Daha sonra kendisi için bir IP yönlendirme tablosu oluşturacaktır. Yönlendirme tablosu oluştururken izleyeceği yol şudur. Kendi
```

</tag2>
<tag3>
IP numarasına giden tüm paketleri kendisine yönlendirecektir. Aynı şekilde kendisi
</tag3>
<tag4>
ile aynı networkte olan adreslere bu IP numarasının ait olduğu arayüz üzerinden
</tag4>
<tag5>
ulaşmaya çalışacaktır. Bu network dışında kalan adreslere yönelen tüm paketleri
</tag5>
</tag>

A.4 Performans analizi için yazılan uygulama yazılımının istatistiki verileri

Yordam No	Alt sistem ya da Yordam Adı	Güçlük Kodu	Satır Sayısı	Tahmini Süre(saat)
1	Arayüz	1.5	81	10
2	Oku	1.5	136	20
3	Yaz	1.8	64	10
4	tablo_olustur	1.8	126	20
5	performans_degerlendirme	2.0	469	72
6	RDBde_bul	1.8	73	10
7	RDBye_ekle	1.5	36	5
8	Metin_Bul	1.8	93	12
9	Veritabanina_ekle	1.5	275	33
10	XMLe_cevir	1.5	41	5
11	XMLDBde_bul	1.8	98	10
12	XMLDBye_ekle	1.5	38	4
13	yeni_kayit	1.5	277	36

EK B TANIMLAMALAR

Markup :İşaretleme

Markup Language :İşaretleme Dili

Tag :Etiket

Plain Text :Düz Metin

Client :İstemci

Data Mining :Veri Madenciliği

Unicode :Evrensel Kod

Root :Kök

Sibling :Kardeş

Child :Çocuk

Parent :Ata

Case Sensitive :Büyük / Küçük Harf Duyarlı

Well Formed :İyi Şekillenmiş

Valid :Geçerli

Tree Based :Ağaç Tabanlı

Event Driven :Olay Güdümlü

Interface :Arayüz

Node Set :Düğüm Kümesi

Boolean :Mantıksal

Number :Sayı

String :Katar (Dizi)

Handler :Kotarıcı

Root Nodes : Kök Düğümleri

Element Nodes : Eleman Düğümleri

Text Nodes :Metin Düğümleri

Attiribute Nodes :Özellik Düğümleri

Name Space Nodes : İsim Uzay Düğümleri

Comment Nodes :Yorum Düğümleri

Location Path : Konum Yolu

Context Node : Bulunulan Düğüm

Axis :Eksen

Uncestor Axis : Bulunulan Düğümün Atası

Link :Bağlantı

Browser :Örün Tarayıcısı

User Interface : Kullanıcı Arayüzü

Uniform :Evrensel Biçimli

ÖZGEÇMİŞ

Adı : Yüce
Soyadı : Pişkinsüt
Uyruğu : T.C
Doğum tarihi : 17/04/1978
Doğum Yeri : Ankara
Askerlik Durumu : Askerlik hizmeti Aralık 2002- Aralık 2003 tarihleri arasında Ağrı ili Patnos ilçesinde Muh. Atğm. olarak tamamlanmıştır.

Şimdiki
Görev Yeri : Havelsan A.Ş.
Görev Unvanı : Uzman Yazılım Mühendisi
İş Adresi : Eskişehir Yolu 4.Km. Mustafa Kemal Mahallesi
İş Tel. No : 0 312 287 35 65 – 1965
e-posta : ypiskinsut@sbd.havelsan.com.tr

İLK VE ORTA ÖĞRENİM DURUMU

Okul	**İl /İlçe**	**Giriş**	**Çıkış**
Yedi Eylül İlkokulu	Aydın/Merkez	1983	1988
Aydın Lisesi	Aydın/Merkez	1988	1994

YÜKSEK ÖĞRENİM DURUMU

Üniversite	**Giriş**	**Çıkış**	**Unvan**	**Derece**
Başkent Üniversitesi	1997	2002	Lisans	3.10
Başkent Üniversitesi	2004	2007	Y.L	3.87

ÇALIŞTIĞI KURUMLAR

Kurum	**İl/İlçe**	**Giriş**	**Çıkış**	**Görevi**
Aselsan A.Ş	Ankara	2000	2000	Stajyer Mühendis
Turkcell İletişim Hizmetleri A.Ş	Ankara	2001	2001	Stajyer Mühendis