HİYERARŞİK VERİLERİN XML VERİTABANI OLARAK MODELLENMESİ VE ARALARINDAKİ BENZERLİĞİN BULUNMASI

Özgür YÜREKTEN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2007
ANKARA

Özgür YÜREKTEN tarafından hazırlanan HİYERARŞİK VERİLERİN XML VERİTABANI OLARAK MODELLENMESİ VE ARALARINDAKİ BENZERLİĞİN BULUNMASI adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd.Doç.Dr. Hasan Şakir BİLGE
Tez Yönetici

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan: : Prof.Dr. M.Sezai DİNÇER

Üye : Yrd.Doç.Dr. Hasan Şakir BİLGE

Üye : Öğr.Gör.Dr. Kıvanç DİNÇER

Tarih : 26/06/2007

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Özgür YÜREKTEN

HİYERARŞİK VERİLERİN XML VERİTABANI OLARAK MODELLENMESİ VE ARALARINDAKİ BENZERLİĞİN BULUNMASI
(Yüksek Lisans Tezi)

Özgür YÜREKTEN

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Haziran 2007

ÖZET

Elektronik Harp Birleştirilmiş Yeniden Programlanabilir Veritabanı (EWIRDB), elektronik harp alanındaki verilerin saklandığı en büyük veri kaynağıdır. Bu veri kaynağındaki veriler dinamik olarak genişleyebilen, şablona göre eksik bilgilere sahip olabilen, 2400'ün üzerinde veriyi saklayabilen ve hiyerarşik olarak modellenmiş yapıya sahiptirler. Bu veri yapısının ilişkisel ve nesne-yönelimli veritabanları ile modellenebilmesi zor olduğu için, EWRIDB miras bir sistem olarak durmakta ve verileri hala düz dosyalarda saklanmaktadır. İlişkisel ve nesne yönelimli veritabanları ile bu kadar büyük bir yapı modellendiğinde, elde edilen veritabanını kullanan yazılımların geliştirilmesi karmaşıklaşmakta ve ortaya çıkan ürün kullanışsız olmaktadır. Bu çalışmada, EWIRDB veri kaynağının daha etkin yönetimi ve kullanımının sağlanması hedeflenmiştir.

Verilerin etkin yönetimi için, ilişkisel veya nesne-yönelimli veritabanlarına alternatif olarak, hiyerarşik yapıların yönetimi konusunda üstün özellikleri olan XML veritabanına taşınmıştır. Etkin kullanım için, veritabanından sorgulama yapılabilmesi ve veritabanında bulunan verilerin benzerlerinin hızlı şekilde ve doğru olarak bulunabilmesi konusunda çalışılmıştır. Bu kapsamda, hiyerarşik verilerin hem yapısal hem de içerik bilgilerini dikkate alan bir benzerlik bulma metodu sunulmuştur. Hiyerarşik verilerin yönetiminde, XML veritabanları etkin bir çözüm sunmasına karşın ilişkisel ve nesne yönelimli veritabanlarına

göre performansları daha düşüktür. XML veritabanındaki benzer verilerin daha hızlı bulunması için bir gruplama metodu sunulmuş ve kullanılmıştır.

Çalışmanın sonucunda, bu alanda kullanılan benzer metotlarla aynı performansa sahip olan sunulan benzerlik bulma metodu ile daha başarılı sonuçlar elde edilmiştir. Bu metot kullanılarak, veritabanında bulunan benzer veriler %97'nin üzerinde bir başarı ile bulunmuştur. Sunulan gruplama metodu kullanılarak, veritabanına erişim %16'ya indirilmiş ve yine başarı ortalaması %97'nin üstünde olmuştur.

Bilim Kodu : **902.1.014**
Anahtar Kelimeler : **Hiyerarşik Veri, Benzerlik, XML Veritabanı, Gruplama**
Sayfa Adedi : **118**
Tez Yöneticisi : **Yrd. Doç. Dr. Hasan Şakir BİLGE**

MODELING HIERARCHICAL DATA FOR XML DATABASE AND FINDING THEIR SIMILARITY

(M. Sc. Thesis)

Özgür YÜREKTEN

GAZI UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
June 2007

ABSTRACT

Electronic Warfare Integrated Reprogramming Database (EWIRDB) is the largest data source in the area of electronic warfare. The structure of this data source has the following characteristics; dynamic data expandability, ability to store over 2400 parameters, ability to accommodate incomplete data, supporting hierarchically modeled structure. It is difficult to design hierarchical data structure using the relational or object oriented database system; therefore EWIRDB remains as legacy system and its data is still stored in plain text files. When this large structure is modeled using the relational or object oriented databases, the complexity of development increases and the out-coming product is hard to use. The goal of this study is to enable EWIRDB data source to be managed and used more effectively.

As an alternative of relational and object oriented database, XML database, which has superior capability of managing hierarchical data structures, is used for better management of data. For effective usage, the study has been carried out to provide the ability to query and to find similar data in database rapidly and accurately. A similarity finding method is proposed based on both structure and content of hierarchical data. Although XML database has superior capability of managing hierarchical data, its performance is not as good as the performance of relational and object oriented databases. To find similar data quickly in XML database, a clustering method is proposed and used.

As a result of this study, we obtained better results with same performance with regard to existing methods in this domain. Using the proposed method, similar data has been found with success rate over %97. Using proposed clustering method, the access rate to database is lowered to %16; still keeping success rate over %97.

Science Code : 902.1.014
Key Words : Hierarchical Data, Similarity, XML Database, Clustering
Page Number: 118
Adviser : Assist. Prof. Dr. Hasan Şakir BİLGE

TEŞEKKÜR

Yüksek lisans yapmam konusunda bana teşvikte bulunan işyerim TÜBİTAK-UEKAE'ye, tezin hazırlanması sırasında çalışmamı gözden geçirerek görüşlerini bildiren tüm iş arkadaşlarıma ve Dr. Kıvanç DİNÇER'e teşekkür ederim.

Çalışmayı gerçekleştirirken yükümü hafifletmek için bana her zaman özverili bir şekilde yardımcı olan eşim Zeynep YÜREKTEN'e sonsuz teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
ABD	Amerika Birleşik Devletleri
AJAX	Asynchronous JavaScript and XML
CDATA	Character Data
CODASYL	Conference on Data Systems Language
DDL	Data Definition Language
DML	Data Manipulation Language
DOM	Document Object Model
DTD	Document Type Definition
EWIRDB	Electronic Warfare Integrated Reprogramming Database
GML	Generalized Markup Language
HTML	Hyper Text Markup Language
ICX	Indirect Clustering for XML Documents
JAXB	Java API for XML Binding
JAXP	Java API for XML Processing
JSF	Java Server Faces
LNAI	Lecture Notes in Artificial Intelligence
LNCS	Lecture Notes in Computer Science
M2DBMS	Multi-lingual, Multi-model Database Modeling System
NASA	National Aeronautics and Space Administration

Kısaltmalar	Açıklama
OO	Object Oriented
OQL	Object Query Language
PAT	Practical Algorithm to Retrieve Information Coded in Alphanumeric
PCDATA	Parsed Character Data
RAHAT	Rafta Hazır Ticari Ürün
SGML	Standard Generalized Markup Language
SQL	Structural Query Language
UML	Unified Modeling Language
URI	Uniformed Resource Identifier
W3C	World Wide Web Consortium
XCLS	XML Document Clustering with Level Similarity
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Style Sheet Language Transformations

1. GİRİŞ

Elektronik harp, elektromanyetik spektrumu kullanan düşmanı engellemek, düşmanın spektrum kullanımını azaltmak veya spektrumu kontrol etmek için askeri faaliyetlerde elektromanyetik enerjinin kullanımıdır [1, 2]. Hem dost birlikler hem de düşman birlikleri elektromanyetik spektrumu kullanan cihazlara sahiptirler ve bu cihazları değişik amaçlarla kullanırlar. Bu cihazların kullanım amaçlarından bazıları; yön bulmak, etraftaki değişiklikleri izlemek veya elektronik taarruz yapmak olarak sıralanabilir. Teknolojik gelişmelerle birlikte, elektronik harp günümüzde yapılan veya yapılacak savaşlarda kullanılabilecek önemli taktiklerden biri olmuştur.

Elektronik harp alanındaki başarının zaferin ön şartı olduğu ve elektronik harp alanındaki başarısızlığın ise askeri yenilgiyi getirebileceği vurgulanmaktadır [3]. Elektronik harp alanında başarıyı sağlamak için, öncelikle düşmanın sahip olduğu kabiliyetleri öğrenmek gerekmektedir. Elektronik istihbarat ve elektronik harp için yayınlanan sinyallerin elde edilmesi ve izlenmesi en önemli faaliyetlerdendir. Bu faaliyetlerde elde edilen verilerin:

- Saklanması,
- Sorgulanması,
- Analiz edilmesi,
- Karşı taarruz yapılması için gerekli verilerin tanımlanmasında kullanmak üzere bir veritabanında bulunması gerekmektedir.

Bu bilgileri saklayan veritabanına genel olarak *elektronik harp veritabanı* [6] denir. Elektronik harp veritabanlarından en çok bilineni EWIRDB (Electronic Warfare Integrated Reprogramming Database) veritabanıdır [3-7].

EWIRDB, elektronik harp alanında 1970 yılından beri kullanılan, genişleyerek günümüzde de kullanılmaya devam eden ve verileri hiyerarşik yapıda saklayan büyük çaplı bir veri kaynağıdır. Veritabanında radarların istihbarat, teknik ve performans verileri saklanmaktadır. Başlangıçta ABD (Amerika Birleşik Devletleri) Hava Kuvvetleri tarafından oluşturulmuş olan veritabanı, daha sonra diğer kuvvetleri

de destekleyecek şekilde genişletilmiş ve tüm elektronik harp verilerinin tek bir havuzda toplanması sağlanmıştır [3].

Yeni teknolojilerin gerisinde kalan, ihtiyaçları karşılama konusunda yetersiz olan ve hala kullanılmak zorunda olunan yazılımlara ve veritabanlarına miras sistemler denir. Miras sistemlerin veritabanları genellikle görev-kritik veritabanlarıdır. Bu veritabanlarında veriler, ağ yapılarında, hiyerarşik yapılarda veya düz dosyalarda saklanmaktadırlar [8]. Bir miras sistemin yenisi ile değiştirilebilmesi için hem yazılımının güncellenmesi hem de verilerinin taşınması gerekmektedir [9, 10]. Fakat, miras sistemler yeni ve düzenli olarak değişen iş isterlerinin karşılanması için yeniden değerlendirmeye ve güncellemeye karşı direnç göstermektedirler [11]. Veritabanlarının gelişiminde ilk olarak hiyerarşik veritabanları kullanılmasına rağmen, EWIRDB verilerinin modellenmesi için hiyerarşik veritabanı kullanılmamıştır. Bunun yerine hiyerarşik veriler düz dosyalarda saklanmıştır. İlerleyen tarihlerde hiyerarşik veritabanlarının yerini ilişkisel ve nesne yönelimli veritabanları almış ve günümüzde XML (Extensible Markup Language) veritabanları [12] ortaya çıkmıştır. Bütün bu gelişmeler olurken EWIRDB veritabanının yapısı ilk tanımlanan şekliyle kalmıştır. Bu veritabanının verilerinin düz dosyalarda olmasından ve veritabanı üzerinde sorgulamalar yapılamamasından dolayı, EWIRDB günümüzde etkin olarak kullanılamaz hale gelmiştir. Bu sebeplerden dolayı EWIRDB de artık miras bir sistemdir. Bu tez çalışması ile EWIRDB verilerinin daha etkin yönetiminin ve kullanımının sağlanması hedeflenmiştir.

EWIRDB verilerinin etkin yönetimi için ilişkisel, nesne-yönelimli ve XML veritabanı yönetim sistemleri ile modellenebilme alternatiflerinin değerlendirilmesi gerekmektedir. Çok büyük hiyerarşik veriler, veritabanına taşındığında, verilere erişim konusunda ve veriler üzerinde yapılan analizlerde performans problemleri olmaktadır. Verilerin uygun bir veritabanına taşınmasının ardından, karşılaşılacak olan bu performans problemlerinin giderilmesi gerekmektedir. Uygun veritabanı modelinin seçilmesinde ve kullanılmasında EWIRDB veri yapısının kritik özellikleri dikkate alınmalı ve bu özelliklerin tamamını destekleyen bir çözüm bulunmalıdır. EWIRDB veri yapısının kritik özellikleri:

- Hiyerarşik yapıda olması,
- Parametre değerlerinin sadece metin tipinde olmayıp değişik veri tipinde olabilmesi,
- Dinamik olarak genişleyebilmesi,
- 2400'ün üzerinde parametresi olması,
- Bir parametre için birden fazla değer tanımlanabilmesi,
- Hiyerarşik yapıdaki parametrelerin bir kısmının boş kalabilmesidir.

EWIRDB veritabanı, elektronik harp alanında operasyonel olarak aktif kullanılmaktadır. Bu veritabanı ayrıca:

- Elektronik harp sistemlerinin araştırma ve geliştirmesinde,
- Elektronik harp sistemlerinin modellenmesinde
- Elektronik harp sistemlerinin testlerinde ve değerlendirilmesinde,
- Elektronik harp sistemi simülasyonlarında,
- Elektronik harp sistemi eğitimlerinde de kullanılmaktadır [3].

EWIRDB tarafından desteklenen elektronik harp sistemlerinin değerinin 30 milyar Amerikan Dolar, operasyonel sistemlerin, eğitim programlarının ve diğer programların EWIRDB üzerinde harcadıkları iş gücünün 30 trilyon Amerikan Dolar değerinde olduğu tahmin edilmektedir [9]. EWIRDB veritabanının bu kadar yaygın kullanılmasına rağmen, veri dosyalarının anlaşılmasında ve kullanımında zorluklar bulunmaktadır [4]. Bunun için bu verilerin etkin kullanımına ihtiyaç duyulmaktadır. Veriler üzerinde değişik sorguların yapılabilmesi gerekmektedir. Sorgular, istenilen bir veriye doğrudan erişimi sağlamanın yanı sıra veriler üzerinde değişik analizler yapılabilmesini de içermelidir.

Analizler için öncelikle hiyerarşik yapıda olan iki ağacın benzerliğinin bulunması gerekmektedir. Hiyerarşik veriler arasındaki benzerliği bulurken ağaçların yapısal özellikleri, içerikleri, parametrelerin öncelik değerleri dikkate alınmalıdır. Ağaçların benzerlikleri kullanılarak:

- İki radar verisi arasındaki benzerliğin bulunması,

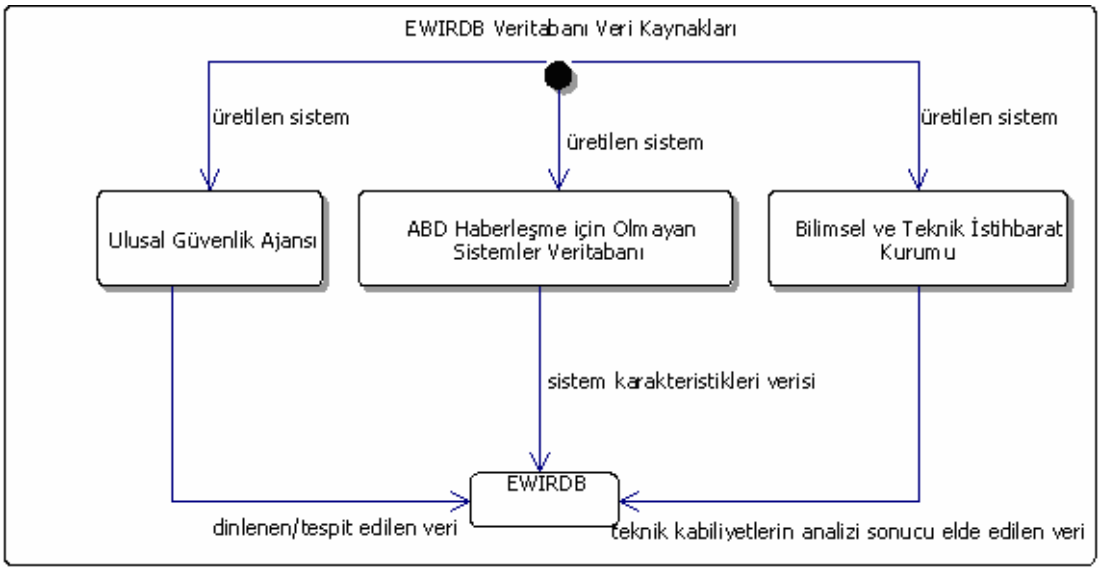
– Adı veya kaynağı bilinmeyen bir elektronik harp verisinin veritabanından sorgulanarak adının bulunması,
– Birbiri ile yakın parametrelere sahip olan verilerin veritabanına eklenmeden önce tespit edilmesi ve böylece veritabanında bulunan veriler arasında belirsizlik durumunun önlenmesi,
– Farklı zamanlarda elde edilen ve aynı kaynağa ait olan EWIRDB verilerinin incelenerek değişikliklerin tespit edilmesi sağlanmalıdır.

Elektronik harp konusunda her ülkenin kendi alt yapısını hazırlaması gerçeği göz önünde bulundurulduğunda, Türkiye'de bu konuda yapılacak çalışmalar elektronik harp konusundaki yetkinlik için büyük önem taşımaktadır.

İkinci Bölüm'de, bu çalışmada kullanılan EWIRDB veritabanı yapısı açıklanmıştır. Üçüncü Bölüm'de, hiyerarşik verilerin modellenmesi, gruplandırılması ve benzerliklerin bulunması konusunda yapılan mevcut çalışmalar anlatılmıştır. Dördüncü Bölüm'de, genişleyebilir işaretleme dili (XML) konusunda genel bilgi verilmiş ve Beşinci Bölüm'de de hiyerarşik verilerin veritabanı ile modellenmesi alternatifleri detaylandırılmıştır. Altıncı Bölüm'de, XML dokümanlarının sorgulanmasında kullanılan sorgulama dili hakkında temel bilgiler verilmiştir. Yedinci Bölüm'de, hiyerarşik verilerin benzerliklerinin bulunması için kullanılabilecek yaklaşımlar açıklanmıştır. Sekizinci Bölüm'de, bu çalışmada yapılanlar detaylı olarak anlatılmış ve sonuçlar Dokuzuncu Bölüm'de verilmiştir.

2. ÇALIŞMADA KULLANILAN VERİTABANI (EWIRDB)

EWIRDB veritabanı 1970 yılında ABD Hava Kuvvetleri tarafından oluşturulmuş, daha sonra diğer kuvvetlere de destekleyecek şekilde genişletilmiştir [3]. Karışık hiyerarşik modele sahip olan EWIRDB veritabanı, *Ulusal Güvenlik Ajansı*, *Bilimsel ve Teknik İstihbarat Kurumu* ve *ABD Haberleşme için Olamayan Sistemler Veritabanı* tarafından kullanılabilecek şekilde düzenlenmiştir. Veritabanına üç kaynaktan veri gelmekte ve veriler üç kaynak tarafından kullanılmaktadır [3].



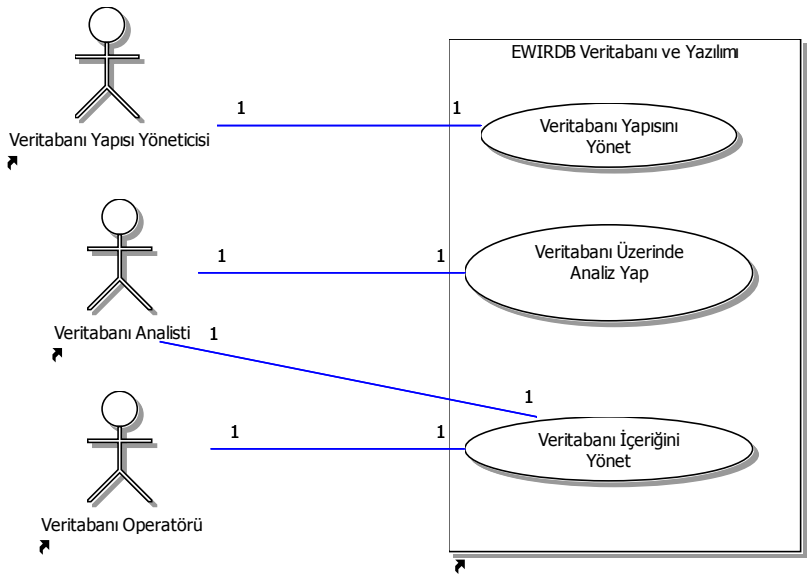
Şekil 2.1. EWIRDB veritabanının veri kaynakları

Yukarıda (Şekil 2.1) EWIRDB veritabanının veri kaynakları verilmiştir. Veritabanının nasıl kullanıldığı ve bilginin nasıl sağlandığı konusunda izlenen yol şu şekildedir: *Ulusal Güvenlik Ajansı* elektromanyetik spektrumda bir radar tarafından gönderilen sinyal izlerini doğrudan dinleyip analiz ederek elde ettiği bilgileri EWIRDB veritabanına girmekle sorumludur. Bu görev için yerel veritabanı üzerinde işlem yapılır ve sonuçlar EWIRDB veritabanına gönderilir. *Bilimsel ve Teknik İstihbarat Kurumu* bir radarın teknik kabiliyetlerini analiz ederek radar sinyallerine ait karakteristik parametreleri EWIRDB veritabanına sağlamakla görevlidir. *ABD Haberleşme için Olmayan Sistemler Veritabanı* diğer iki kurum tarafından sağlanan verileri kullanarak sahip olunan ve üretilen radarların sistem karakteristiklerini belirleyerek elde edilen verileri EWIRDB veritabanına sağlar.

2.1. EWIRDB Veritabanının Kullanım Durumları

Operasyonel olarak veritabanını yöneten veya veritabanından faydalanan yazılımlar EWIRDB veritabanını değişik amaçlar için kullanabilmektedir. Bu bölümde operasyonel kullanımları UML (Unified Modeling Language) kullanım durumu diyagramları ile anlatılacaktır.

Öncelikli kullanım durumları veritabanının yapısal ve içerik olarak yönetilmesidir. Verilerin benzerliklerinin belirlenmesi ve elde bulunan bir verinin veritabanında kayıtlı olan verilerin hangilerine benzediğinin bulunması en çok ihtiyaç duyulan diğer kullanım durumlarıdır. Belirtilen kullanım durumları aşağıda (Şekil 2.2) UML kullanım durumu diyagramı olarak verilmiştir. Kullanım durumlarında belirtilen rollere, yazılımı kullanan kişiler, analiz yazılımları veya simülatörler sahip olabilirler.



Şekil 2.2. EWIRDB veritabanı ve yazılımı kullanım durumları

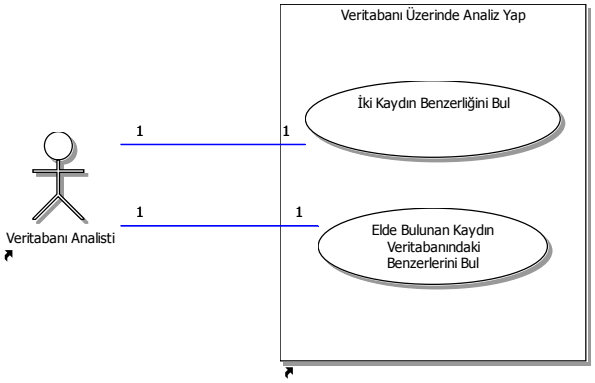
EWIRDB veritabanı ve yazılımını kullanmak için üç rol tanımlanmıştır:

– *Veritabanı Yapısı Yöneticisi*: Veritabanının yapısında olacak değişiklikleri gerçekleştirmekle sorumludur.

– *Veritabanı Analisti*: Veritabanında bulunan her türlü veriyi analiz etmek, alınan / elde edilen verileri veritabanına aktarmakla görevlidir.

– *Veritabanı Operatörü*: Veritabanında bulunan verilerin güncellenmesinden sorumludur.

EWIRDB Veritabanında bulunan verilerin 2400'ün üzerinde parametresi bulunabilmektedir. Bu verilerin bazıları ölçüm sonuçları ile bazıları ise değişik analizlerle elde edilmektedir. Daha önce de belirtildiği üzere veritabanın veri kaynakları birden fazladır. Bunun için *Veritabanı İçeriğini Yönet* kullanım durumu birden fazla rol tarafından gerçekleştirilmektedir. *Veritabanı Analisti* veritabanında bulunan verileri analiz ederek yeni veri girişi yapabilmekte, *Veritabanı Operatörü* ise sadece dinlenen / tespit edilen verilerin girişini yapabilmektedir. *Veritabanı Analisti*, elinde bulunan iki kaydın benzerliğini ve elde bulunan bir kaydın veritabanındaki benzerlerini (Şekil 2.3) bulabilmektedir. Şekil 2.3'te belirtilen kullanım durumları ile analiz aşamasında verilerin benzerliklerinin tanımlanmasında *Veritabanı Analisti*'ne yardımcı olunmaktadır.

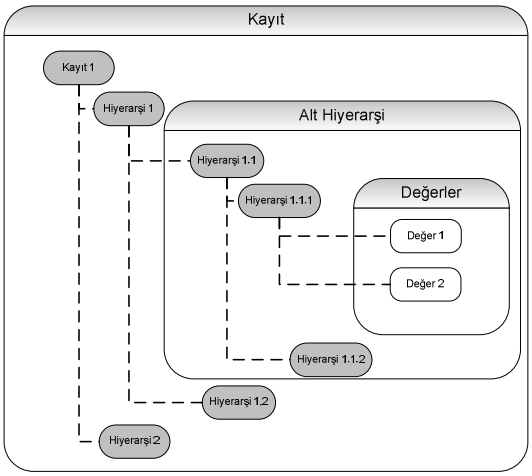


Şekil 2.3. Veritabanı Üzerinde Analiz Yap kullanım durumunun detayları

İki Kaydın Benzerliğini Bul kullanım durumu ile iki kaydın birbirine olan yakınlıkları tespit edilebilmektedir. EWIRDB veritabanında bulunan veriler, elektromanyetik spektrumu kullanan iki ayrı cihazın birbirinden ayırt edilmesi için kullanılmaktadır. Veritabanında bulunan farklı verilerin içeriğinin birbirine çok benzemesi durumunda, bu verileri kullanan elektronik harp cihazları aldıkları sinyallerin kaynağını tespit edememekte veya kaynağı ayırt edememektedir. Bu sebepten dolayı *Veritabanı Analisti'*nin veritabanında bulunan farklı verilerin benzerliğini azaltması ve ayıt

edilebilir hale getirmesi gerekmektedir. Bu kullanım durumu, bu konuda *Veritabanı Analisti* için kritik önem taşımaktadır.

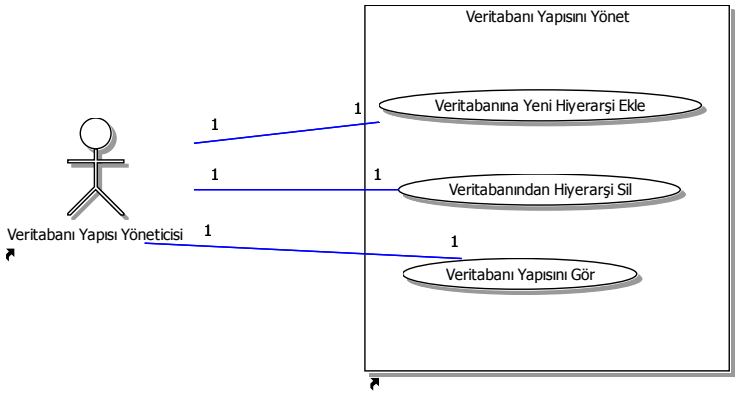
Elde Bulunan Kaydın Veritabanındaki Benzerlerini Bul kullanım durumu da veritabanında bulunmayan bir kaydın veritabanında başka bir isimle tanımlanıp tanımlanmadığını kontrol etmekte kullanılır. Diğer bir değişle tespit edilen ve adı bilinmeyen bir kaydın veritabanında olup olmadığı belirlenir, ve var ise hangi kayda karşılık geldiği tespit edilir. Bu kullanım durumu ile veritabanında farklı isimlerle tekrar eden kayıtların bulunmasının önüne geçilir. Yukarıda tanımlanan kullanım durumları, veritabanı üzerinde gerçekleştirilebilecek en temel analizlerdir. Daha detaylı analizler tanımlanarak, veritabanının özel amaçlarla kullanılması da sağlanabilir.



Şekil 2.4. EWIRDB veritabanında bulunan bir kaydın yapısı

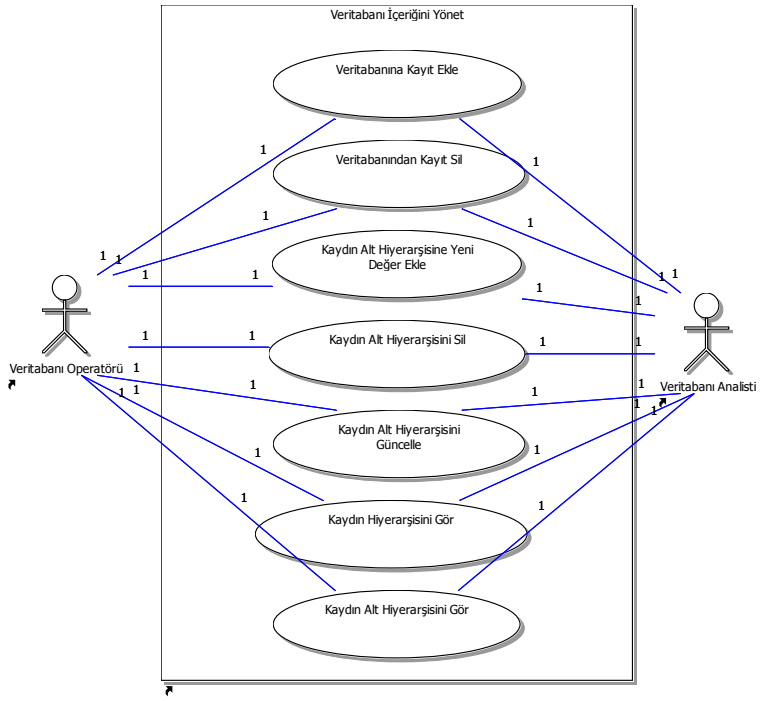
Yukarıda (Şekil 2.4) EWIRDB veritabanında bulunan bir kaydın yapısı verilmiştir. Veritabanında bulunan bir kayıt hiyerarşik olarak saklanır. Kayıt, ağaç yapısının en üstünde kök olarak bulunur ve değerler hiyerarşik yapılardır. Ağacın herhangi bir çocuğu ve altındaki bütün değerler alt hiyerarşiyi oluşturur. Hiyerarşinin en sonunda değerler bulunur. EWIRDB veritabanında standart bir hiyerarşi belirlenmiştir. Bu standart yapıdaki hiyerarşideki alanlar bir veya birden fazla değere sahip olabilir veya hiçbir değere sahip olmayabilir.

Şekil 2.5, veritabanını yönetebilmek için tanımlanan kullanım durumunun detaylarını yine UML kullanım durumu diyagramı olarak göstermektedir.



Şekil 2.5. Veritabanı Yapısını Yönet kullanım durumunun detayları

Yeni analizler/çalışmalar sonucu elde edilen veya elektronik harp için gerekli görülen alanların veritabanına yeni bir hiyerarşi olarak eklenmesi için veritabanında yapısal değişikliklerin yapılması ihtiyacı doğmaktadır. *Veritabanı Yapısı Yöneticisi* en çok *Veritabanına Yeni Hiyerarşi Ekle* kullanım durumunu gerçekleştirir. Bu kullanım durumu ile veritabanı yapısının güncel tutulması ve gerektiğinde genişletilmesi sağlanır. *Veritabanından Hiyerarşi Sil* kullanım durumu, güncelliğini yitiren hiyerarşilerin silinmesini sağlar. *Veritabanı Yapısını Gör* kullanım durumu ile veritabanının yapısal durumu hiyerarşik olarak görüntülenir. Özet olarak; bu kullanım durumları, veritabanı yapısının yönetilmesi, veritabanı yapısında değişikliklerin yapılmasını, güncelliğini korumasını, yapının genişletilebilir olmasını ve hiyerarşilerin değiştirilebilmesini sağlarlar. Yapısal değişikliklerin yönetilebilmesine ek olarak; veritabanında bulunan verilerin ekleme, güncelleme, silme ve görüntüleme işlemlerinin yapılması ve veritabanı kayıtlarının alt hiyerarşilerinin yönetimi de önemli kullanım durumlarıdır. Veritabanı içeriğinin yönetiminin detayları Şekil 2.6'da kullanım durumu diyagramı olarak verilmiştir. Şekil 2.6'da verilen *Veritabanına Kayıt Ekle*, *Veritabanından Kayıt Sil* kullanım durumları ilişkisel veritabanlarında sıkça bulunan işlemlerdir. Bu kullanım durumları yardımı ile veri yapısında 2400'ün üzerinde parametresi bulunabilen kayıtlar bir seferde veritabanına eklenebilir veya bir seferde veritabanından silinebilir.



Şekil 2.6. Veritabanı İçeriğini Yönet kullanım durumunun detayları

Veritabanında bulunan bir verinin EWIRDB yapısında tanımlanan bütün alanları dolu olmayabilir. Eksik olan bilgiler daha sonra elde edilirse, kaydın ilgili alt hiyerarşisine eklenebilir. Kaydın ilgili alt hiyerarşisinde sadece bir değer olma zorunluluğu yoktur. Alt hiyerarşide birden fazla değer olabilir. Veritabanında kayıtlı olmayan yeni bir değer elde edildiğinde, veritabanında kayıtlı olan aynı hiyerarşideki değerlerin yanına ek olarak yeni bir değer olarak eklenir. *Kaydın Alt Hiyerarşisine Yeni Değer Ekle* kullanım durumu bu işlemleri gerçekleştirmek için kullanılır. *Kaydın Alt Hiyerarşisini Sil* kullanım durumu verinin ilgili alt hiyerarşisindeki bütün değerlerin silinmesi ve *Kaydın Alt Hiyerarşisini Güncelle* kullanım durumu alt hiyerarşide bulunan herhangi bir değerinin güncellenmesi içindir. *Kaydın Hiyerarşisini Gör* ve *Kaydın Alt Hiyerarşisini Gör* kullanım durumları seçilen kaydın tüm hiyerarşisinin ve alt hiyerarşisinin görüntülenmesini sağlarlar.

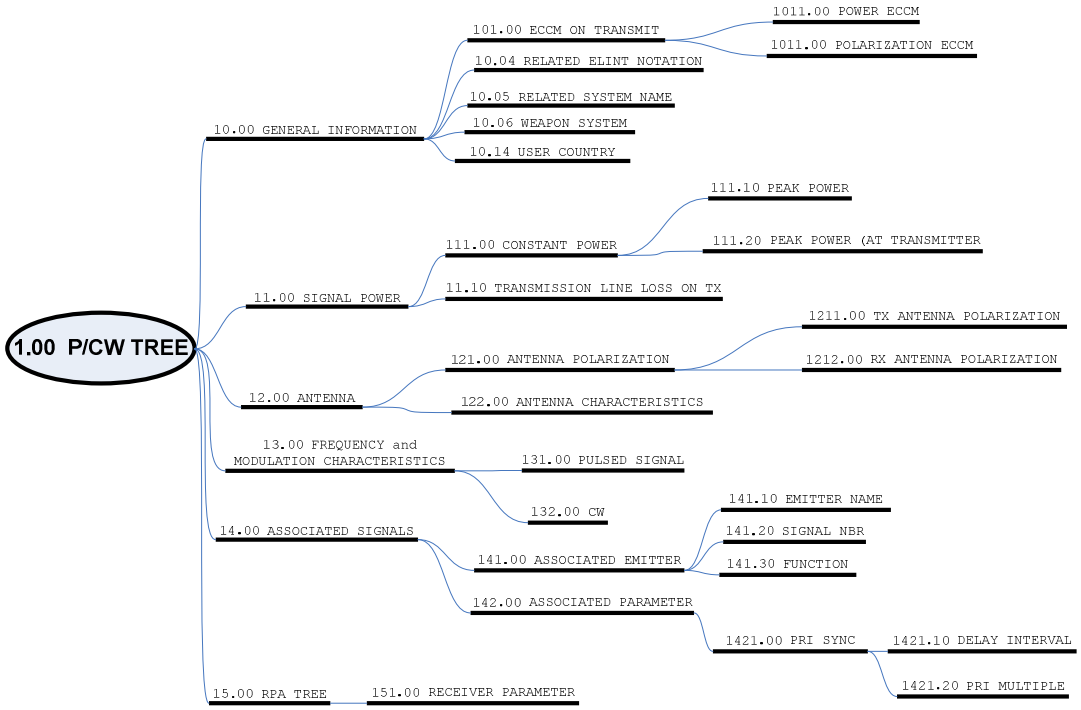
2.2. Kullanım Durumlarının Gerçekleştirilmesi

Kullanım durumlarını gerçekleştirmek için yapılan çalışmalar sonucu XML veritabanı ile EWIRDB veritabanı yapısı modellenmiş ve örnek bir uygulama geliştirilmiştir. XML veritabanının yapısının güncellenebiliyor olması ile *Veritabanı*

Yapısını Yönet kullanım durumu gerçekleştirilmiştir. *Veritabanı İçeriğini Yönet* kullanım durumunu gerçekleştirmek için XML veritabanını sorgulama dili olan XQuery [13, 14] kullanılmıştır. *Veritabanı Üzerinde Analiz Yap* kullanım durumunu gerçekleştirmek için daha önceden yapılan çalışmalar incelenmiş ve hiyerarşik veriler için benzerlik ve gruplama metotları sunulmuştur.

2.3. EWIRDB Veritabanı Yapısı

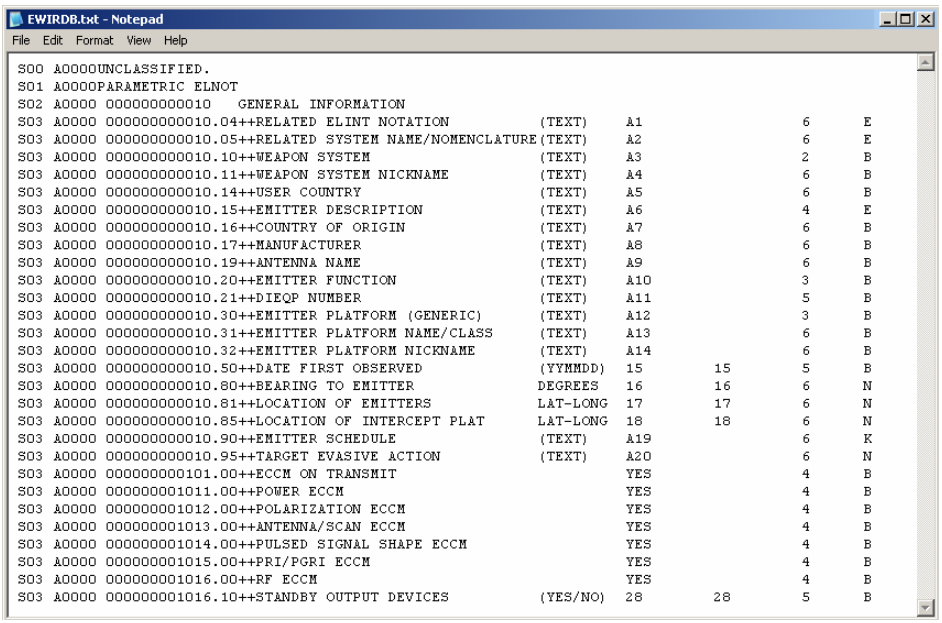
Şekil 2.7'de bir radar verisinin örnek hiyerarşisi verilmiştir.



Şekil 2.7. EWIRDB veritabanında bulunan bir verinin örnek hiyerarşisi

EWIRDB Veritabanının verileri düz dosyalarda saklanmaktadır [3]. Her bir dosyada bir radar bilgisi bulunmaktadır. Dosyalarda bulunan verilerin hiyerarşik bir yapıya sahip olabilmesi için dosyadaki her bir değere bir *hiyerarşi numarası* verilmiştir. Şekilde de görüldüğü gibi veriler bir ağaç yapısındadır. Her bir verinin hiyerarşisi *hiyerarşi numarası* ile anlaşılır. Emiterin parametreleri için kökten itibaren kategorizasyon tanımlanmış ve bu kategorizasyona göre alt dallar oluşturulmuştur. Kategorizasyon için tanımlanan her eleman için bir üst seviyede bulunan elemanın

numarası on ile çarpılıp o hiyerarşide daha önce kullanılmayan bir ardışık sayı (1, 2, 3, ... şeklinde ardışık sayılardan biri) eklenerek *hiyerarşi numarası* verilmiştir. Örneğin verinin *ANTENNA* (anten) elemanı *ANTENNA POLARIZATION (anten polarizasyonu)* ve *ANTENNA CHARACTERISTICS* (anten karakteristikleri) adında alt elemanlara bölünmüştür. *ANTENNA* elemanının *hiyerarşi numarası* 12.00'dir. *ANTENNA POLARIZATION* elemanı, bir üst elemanın numarası olan 12 sayısının on ile çarpılarak ardışık bir sayı eklenmesi ile "121.00" *hiyerarşi numarası*nı almıştır. *ANTENNA CHARACTERISTICS* elemanının *hiyerarşi numarası* da bir üst elemanın numarasının on ile çarpılıp o hiyerarşide daha önce kullanılmayan sıradaki ardışık numaranın eklenmesi ile "122.00" olmuştur. ".00" ile biten veriler ağacın dalları için kullanılan alanlardır. Bu alanlar alt elemanlara veya hiyerarşilere sahip olabilir. ".00" ile bitmeyen her bir eleman gerçek değerlerin saklandığı alanlardır.


EWIRDB.txt - Notepad

File Edit Format View Help

```
S00 A0000UNCLASSIFIED.
S01 A0000PARAMETRIC ELNOT
S02 A0000 000000000010   GENERAL INFORMATION
S03 A0000 000000000010.04++RELATED ELINT NOTATION          (TEXT)     A1                6      E
S03 A0000 000000000010.05++RELATED SYSTEM NAME/NOMENCLATURE(TEXT)     A2                6      E
S03 A0000 000000000010.10++WEAPON SYSTEM                   (TEXT)     A3                2      B
S03 A0000 000000000010.11++WEAPON SYSTEM NICKNAME          (TEXT)     A4                6      B
S03 A0000 000000000010.14++USER COUNTRY                    (TEXT)     A5                6      B
S03 A0000 000000000010.15++EMITTER DESCRIPTION             (TEXT)     A6                4      E
S03 A0000 000000000010.16++COUNTRY OF ORIGIN               (TEXT)     A7                6      B
S03 A0000 000000000010.17++MANUFACTURER                    (TEXT)     A8                6      B
S03 A0000 000000000010.19++ANTENNA NAME                    (TEXT)     A9                6      B
S03 A0000 000000000010.20++EMITTER FUNCTION                (TEXT)     A10               3      B
S03 A0000 000000000010.21++DIEQP NUMBER                    (TEXT)     A11               5      B
S03 A0000 000000000010.30++EMITTER PLATFORM (GENERIC)      (TEXT)     A12               3      B
S03 A0000 000000000010.31++EMITTER PLATFORM NAME/CLASS     (TEXT)     A13               6      B
S03 A0000 000000000010.32++EMITTER PLATFORM NICKNAME       (TEXT)     A14               6      B
S03 A0000 000000000010.50++DATE FIRST OBSERVED             (YYMMDD)   15        15      5      B
S03 A0000 000000000010.80++BEARING TO EMITTER              DEGREES    16        16      6      N
S03 A0000 000000000010.81++LOCATION OF EMITTERS            LAT-LONG   17        17      6      N
S03 A0000 000000000010.85++LOCATION OF INTERCEPT PLAT      LAT-LONG   18        18      6      N
S03 A0000 000000000010.90++EMITTER SCHEDULE                (TEXT)     A19               6      K
S03 A0000 000000000010.95++TARGET EVASIVE ACTION           (TEXT)     A20               6      N
S03 A0000 000000000101.00++ECCM ON TRANSMIT                           YES               4      B
S03 A0000 000000001011.00++POWER ECCM                                 YES               4      B
S03 A0000 000000001012.00++POLARIZATION ECCM                          YES               4      B
S03 A0000 000000001013.00++ANTENNA/SCAN ECCM                          YES               4      B
S03 A0000 000000001014.00++PULSED SIGNAL SHAPE ECCM                   YES               4      B
S03 A0000 000000001015.00++PRI/PGRI ECCM                              YES               4      B
S03 A0000 000000001016.00++RF ECCM                                    YES               4      B
S03 A0000 000000001016.10++STANDBY OUTPUT DEVICES          (YES/NO)   28        28      5      B
```

Şekil 2.8. Örnek EWIRDB Dosyası

EWIRDB veritabanının yapısı hiyerarşik olarak tanımlanmasına karşın veriler dosyalarda saklanmaktadır. Şekil 2.8'de örnek bir dosya içeriği verilmiştir. Bu dosyalar özel olarak formatlanarak hangi verinin ve veri alanının nerede bulunacağı tanımlanmıştır.

Her bir emiter bilgisi yukarıda belirtilen formattaki dosyalarda saklanmaktadır. Her bir emiter için bir dosya oluşturulmakta ve bu dosyalar işlenerek istenilen veri bulunmaya çalışılmaktadır. Bu verilerin veri tipleri, alabilecekleri minimum maksimum değerler ayrı bir dosyada tanımlanmakta fakat aralarında doğrudan bir bağlantı bulunmamaktadır.

Emiterin bir parametresi, çevrede yayılan sinyallerin dinlenmesiyle veya emiterin teknik özelliklerinin incelenmesiyle elde edilen bilgilerle doldurulabilir. Ayrıca emiterin her zaman aynı konfigürasyonda çalışmayacağı, farklı amaçlar için farklı modlarda çalışabileceği düşünülmüştür. Bu durumları desteklemek için bir dala birden fazla değer girişi zorunlu hale gelmiştir. Dosyanın yapısı, elde edilen ve birden fazla tekrarlanabilen bilgilerin bir yerde saklanmasını da destekleyecek şekilde tanımlanmıştır. Tanımlanan yapıda bir alana birden fazla değer girişi yapılması olanağı sağlanınca, emiterle ilgili tüm verilerin bir yerde saklanması kolaylaşmış fakat bu parametrelerin ne zaman ve hangi kombinasyonlarda anlamlı olduğunun belirlenmesi problemi ortaya çıkmıştır. Birden fazla değer girişi yapılan alanların birbirinden ayırt edilmesi için *SUFFIX CODE* (ön ek kodu) adında bir alan tanımlanmıştır ve bu alana dosyada daha önce kullanılmayan iki karakterli değerler girilmesi zorunlu hale getirilmiştir. Dosyaların sonlarında *SUFFIX TABLE* (ön ek tablosu) adında bir matris eklenmiş ve *SUFFIX CODE* alanları kullanılarak bu matris içinde hangi değerin ne zaman ve hangi kombinasyonda geçerli olduğu gösterilmiştir.

EWIRDB veritabanının hiyerarşik yapıda oluşunun bir sebebi de hiyerarşiye yeni parametrelerin eklenmesini kolaylaştırmaktır. Teknolojinin ilerlemesi ve elektronik harp alanında çalışmaların genişlemesi üzerine 1970 yılından bu yana veritabanına birçok parametre eklenmiştir. Bu parametreler ağacın alt dallarına yeni parametre olarak eklenmiş ve sıklıkla kullanılmıştır. Yeni parametrelerin eklenebilmesi veritabanının genişleyerek güncel kalmasını sağlamıştır. Hiyerarşiye eklenen yeni alanların dosyalara eklenmesi için sistematik bir yol tanımlanmamış, bütün dosyalar güncellenerek bu özelliğin eklenmesi sağlanmıştır.

EWIRDB veritabanında çok fazla sayıda parametre tanımlanmasına karşın, tüm radarlar için tüm parametrelerin doldurulması zorunlu değildir. Bunun için her bir radar, tanımlanan yapının bir alt kümesini oluşturacak şekilde yapılandırılmıştır. Örneğin bazı radarların *14.00 ASSOCIATED EMITTER* (ilişkili radar) parametresi ve alt hiyerarşisi olmayabilir, bazı radarlar için bu daldaki temel parametrelerini tanımlanabilir ve bazı radarlar ise bu daldaki tüm parametreleri doldurabilir. Doldurulmayan parametreler dosya içerisine eklenmemekte ve o radar için bu parametrelerin olmadığı veya bilinmediği kabul edilmektedir.

EWIRDB veritabanında 2400'ün üzerinde parametre bulunduğu göz önüne alındığında bütün bu parametrelerin elde edilip veritabanına girilmesi mümkün olmamaktadır. Öncelikle emiterin hangi parametrelerinin girilmesi gerektiği bütün parametrelere 1 ile 6 (1 ve 6 dahildir) arasında *öncelik numarası* verilerek tanımlanmıştır. Veritabanında sadece bilimsel çalışmalarda kullanılabilecek veya test amaçlı parametreler de bulunmaktadır. Bu parametrelerin operasyonel ortamda doğrudan kullanılamayacağı da ayrıca belirtilmiştir. Bu parametrelere düşük *öncelik değerleri* verilmektedir. Operasyonel olarak kritik olan parametrelere ise yüksek *öncelik değeri* verilmektedir. Bu *öncelik değerleri* ağacın doldurulması sırasında bir yol haritası çizmektedir.

3. MEVCUT ÇALIŞMALAR

Bu bölümde, sırasıyla; hiyerarşik verilerin modellenmesi, hiyerarşik verilerin benzerliklerinin hesaplaması ve hiyerarşik verilerin gruplandırması konusunda yapılan çalışmalar anlatılacaktır.

3.1. Hiyerarşik Verilerin Modellenmesi

EWIRDB veritabanının tekrar modellenmesi konusunda daha önce yapılan çok az çalışma bulunmaktadır. Daha önce yapılan çalışmalarda EWIRDB veritabanının hiyerarşik yapısının ilişkisel, nesne-yönelimli, ağ modellerine nasıl dönüştürüleceği üzerinde durulmuştur. Bu bölümde, EWIRDB veritabanı yapısının belirli bir parçası üzerinde ve EWIRDB veritabanı yapısına benzer diğer veritabanlarının modellenmesi konusunda yapılan çalışmalar anlatılacaktır.

Barnes [7] EWIRDB veritabanı yapısının nesne-yönelimli olarak modellenmesi için yaptığı çalışmada veritabanı formatını açıklamıştır. Bu formatın nesne-yönelimli olarak nasıl modellenebileceği konusunda veritabanının mantıksal analizini yapmıştır. Çalışmada EWIRDB yapısının seçilmesini üç sebebe dayandırmıştır:

- Bu veritabanının elektronik harp için kritik bir veritabanı olması,
- Veritabanı yapısının etkin ve kullanılabilir olamaması,
- Veritabanının dosyalar içinde ve ilişkisel olarak saklanmasıdır.

Çalışmada, veritabanının ilişkisel olarak tanımlanmış olduğu söylenmiş fakat bu ilişkisel yapının standart ilişkisel yapı olmayıp süper sınıf, alt sınıf, genelleme, özelleştirme, soya çekim gibi nesne-yönelimli modellemenin özelliklerini de taşıyan gelişmiş ilişkisel model olduğu belirtilmiştir. Veritabanı, öncelikle özel olarak tanımlanmış gelişmiş ilişkisel modelle modellenmiş ve ardından nesne-yönelimli olarak modellenmesi yapılmıştır. Çalışmada veritabanının mantıksal olarak bir kısmının modellenmesi hedeflenmiştir.

EWIRDB veritabanının tekrar modellenmesi konusundaki araştırmalardan bir tanesi de Coyne [3] tarafından yapılan çalışmadır. Çalışmada, Coyne EWIRDB

veritabanının dosya yapısı, veritabanında bulunan verilerin nesne yönelimli olarak analiz edilmesi ve tasarlanması üzerinde durmuştur. EWIRDB yapısını, biçimini, kullanımını, bu verilerin nesnelerle nasıl ifade edilebileceğini ve nesnelerin birbirleri ile olan ilişkilerini tanımlamıştır. Nesneler ve aralarındaki ilişkileri tanımlarken nesne-yönelimli programlamanın soya çekim, genelleştirme, özelleştirme ve birleştirme gibi yeteneklerinden faydalanmıştır. EWIRDB veritabanının nesne-yönelimli olarak modellenebilmesi için elektronik harp konusunda alan uzmanı olmayı gerektirdiğini belirtmiş ve kendi çalışması ile elde edilen nesne yönelimli model ile alan uzmanı olmayanların da EWIRDB veritabanının yapısını anlamalarını kolaylaştıracağını söylemiştir. Çalışma, sadece analiz ve tasarım aktivitelerini içerdiği için nesne yönelimli olarak EWIRDB veritabanı modeli tamamlanmamıştır. Buna karşın çalışmada ortaya konulan model ile tüm EWIRDB veritabanını nesne yönelimli olarak gerçekleştirilebileceği savunulmuştur.

Coyne [3]'nin çalışması sonucu elde ettiği nesne-yönelimli veritabanı modelini Lee ve McKenna [4] kendi çalışmalarında gerçekleştirmişler ve test etmişlerdir. Modelin gerçekleştirilmesi sırasında Coyne [3]'nin tanımladığı modelde bazı hatalar bulmuşlar ve bu hatalar giderildikten sonra EWIRDB veritabanı yapısının ancak %20'si üzerinde çalışmalarına devam etmişlerdir. Çalışma sonucunda tüm veritabanının gerçekleştirilmediği, nesneler arası ilişkilerin basitleşmediği ve karmaşıklığın hala devam ettiği belirtilmiştir. Ayrıca nesne yönelimli olarak modelin gerçekleştirilmesinde çoklu soya çekim özelliklerinin kullanıldığı ve bunun karmaşıklığı arttırdığı üzerinde durulmuştur. EWIRDB veritabanı için tanımlanan nesneler arasında bulunan ilişkiler netleştirilerek veritabanı yapısının daha anlaşılabilir hale geldiği ifade edilmiştir. EWIRDB veritabanı yapısındaki bir radar yeni bir alan ekleme ve bir alana değer girebilme özellikleri gerçekleştirilmemiş sadece dokuz çeşit sorgu formatı tanımlanmış ve bu sorguları çalıştıran uygulama hazırlanmıştır. Bu dokuz sorgu formatının EWIRDB içindeki her türlü veriye ulaşmayı sağlamadığı fakat veritabanının genel kullanım amacının büyük bir çoğunluğunu karşıladığı belirtilmiştir.

Bu konudaki diğer bir araştırma da Werre ve Diehl [6]'nin çalışmasıdır. Werre ve Diehl çalışmalarında, Coyne [3]'nin oluşturduğu modeli ağ veritabanı modeline çevirmeyi ve elde edilen ağ modeli ile EWIRDB veritabanı yapısının bir bölümünü gerçekleştirmeyi amaçlamışlardır. Fakat gerçekleştirilen modelin istenilen şekilde testi yapılamamış, ağ modelinin nesne yönelimli, ilişkisel veya hiyerarşik modeller ile karşılaştırılması gerçekleştirilememiştir.

EWIRDB veritabanının yeniden modellenmesi üzerine yapılan Scrivener ve Edwards [5]'ın çalışması, EWIRDB veritabanının ilişkisel olarak modellenmesine yöneliktir. EWIRDB yapısı öncelikle ilişkisel tablolarla ifade edilmiştir. Bu aşamada hiyerarşik verinin birçok soya çekim, genelleme ve özelleştirme ile ifade edilmesi gerektiği ve bunun ilişkisel olarak ifadesinde zorlukların olduğunu belirtmişlerdir. Çalışmada EWIRDB veritabanı yapısının sadece bir bölümü üzerinde modelleme gerçekleştirilmiştir. Yedi çeşit sorguyu gerçekleştirebilecek şekilde bir yazılım hazırlanmış ve test edilmiştir. EWIRDB veritabanı için ilişkisel veri modelinin nesne yönelimli modelden daha kullanışsız olduğu ve nesne yönelimli yaklaşımın daha etkili ve daha kolay olabileceği belirtilmiştir.

EWIRDB veritabanını hiyerarşik modelden ilişkisel, nesne yönelimli ve ağ modellerine taşınması için yapılan tüm çalışmalarda [3-7] teorik ve pratik olarak gerçekleştirilebilirlik üzerinde durulmuş, fakat hiçbir çalışmada tüm EWIRDB veritabanı modellenmemiş ve veritabanı üzerinde istenilen her sorgunun gerçekleştirilebilmesi olanağı sunulmamıştır. İlişkisel veritabanı modeli için SQL (Structural Query Language), ağ veritabanı modeli için CODASYL-DML (Conference on Data Systems Language, Data Manipulation Language) ve nesne yönelimli veritabanı modeli için de OO-DML (Object Oriented - Data Manipulation Language) arayüzü kullanılmıştır. Çalışmalar tüm veri modellerini ve tüm veritabanı arayüzlerini destekleyecek M2DBMS (Multi-lingual, Multi-model Database Management System) adında bir sistem üzerinde test edilmiştir. Çalışmalarda veritabanının standart bir arayüzle sorgulanabilmesini sağlamak yerine birden fazla arayüzü destekleyecek bir sistem ortaya çıkarmaya çalışılmıştır.

Farklı problem alanlarındaki hiyerarşik veritabanlarının XML veritabanı olarak modellenebileceğini belirten çalışmalardan bir tanesi Some ve arkadaşlarının çalışmalarıdır [15, 16]. Çalışmada, NASA (National Aeronautics and Space Administration)'nın bilimsel misyonunu gerçekleştirmesine yardımcı olacak, değerli ve çığır açacak teknolojilerin seçiminde, bununla beraber bu teknolojileri prototip seviyesinden kullanılabilir seviyesine gelinceye kadar olgunlaştırılmalarını takip edecek XML veritabanı ve bu veritabanını kullanan bir görev yazılımı anlatılmaktadır. Çalışma genel olarak görev yazılımının geliştirilmesinde XML veritabanının seçimindeki gerekçeler üzerinedir. Çalışmalarında şu anki teknolojilerin NASA için fonksiyonel ve yapısal olarak önemini, teknolojilerin şu anki yeteneklerini, NASA'nın fonksiyonel/yapısal isterlerini karşılamaları için sahip olmaları gereken kriterleri belirlemişler ve takipte kullanılacak veritabanını tasarlamışlardır. Problem alanını yukarıda belirtilen isterleri karşılamak için dört parçaya bölmüşlerdir. Öncelikle NASA'nın örgütlenme şemasının tanımlanmasını ve organizasyondaki bölümlerin birbirleri ile ilişkilendirilmesini sağlamışlardır. Daha sonra NASA'nın misyonunu bağımsız olarak fonksiyonel ve yapısal olarak ayrıştırmışlar, her fonksiyonun ve yapının isterlerini nicel parametrelerle ifade etmişlerdir. Takip edilecek teknolojileri tanımlamışlar ve nicel yetenekleri belirlemişlerdir. Bu yapı ile EWIRDB veritabanı yapısı arasında benzerlikler bulunmaktadır. Her iki veritabanında da veriler değişik hiyerarşilerde tutulmakta, yeni alanlar eklenebilmekte, bir alana birden fazla değer girilebilmektedir. Çalışmada, tanımlanan veri yapılarının modellenmesi öncesinde, analiz araçları tarafından kullanılabilecek standart bir arayüzü destekleyebilmesi ve kullanıcılar için standart sorgulama / veri girişi arayüzü olanağı sağlayabilmesi gibi kriterler ön planda tutulmuştur. Bütün yukarıda belirtilen kriterler dikkate alındığında, verilerin hiyerarşik olarak ifade edilebilmesi, tüm alanların insanlar tarafından okunabilir ve makine tarafından anlaşılabilir olması, internet üzerinden ulaşılabilmesi, kullanıcı arayüzlerinin açık, sezgisel olarak kavranabilen ve kullanıcı dostu olması, güvenli erişilebilmesi, veri yapısının derinliğinin istenildiği kadar arttırılabilmesi, hızlı erişilebilmesi ve tarihsel verilere erişilebilmesi gibi yeni isterlerin de karşılanmasının gerekliliği üzerinde durulmuştur. Tanımlanan veri yapılarının ilişkisel veritabanı ile modellenmesi durumunda, hiyerarşik verilerin ancak normalizasyonla ilişkisel olarak

ifade edilebildiği, çok fazla alan olduğu için normalizasyon sonucu veri yapısının karışacağı (normalizasyonla çok fazla tablo oluşacaktır) ve okunaklılığın kaybolacağı, veriler hiyerarşik yapıda olduğu için hiyerarşiye yeni bir alan, yeni bir seviye eklemenin zor olacağı (yeni alan ve hiyerarşi eklemek tasarlanan tabloların yapısında değişikliğe sebep olacaktır) ve kullanıcı arayüzlerinde hiyerarşik yapıyı görüntülemek için çok fazla işlem yapılması gerekeceği gibi kısıtlar ve zorluklar olduğu anlatılmıştır. Buna karşılık tanımlanan veri yapılarının ilişkisel veri tabanı yerine XML veritabanı ile modellenmesi durumunda verileri hiyerarşik olarak saklanmanın ve ulaşılmanın daha kolay olduğu, insanlar tarafından okunabilen ve makine tarafından anlaşılabilen yapıların tanımlanabildiği, daha anlaşılır ve kolay şekilde hiyerarşik verilerin kullanıcı arayüzünde gösterilebildiği, yeni alanların ve seviyelerin eklenebildiği belirtilmiştir. Çalışma sonucunda, veri yapılarına uygun bir XML veritabanı tasarlanmış, tasarlanan XML veritabanını kullanan bir görev yazılımı hazırlanmış ve XQuery sorgulama dili ile analiz araçlarının veritabanına ulaşıp kullanabilecekleri bir arayüz sağlanmıştır.

XML veritabanları öğrenci veritabanı [17], biyoloji veritabanı [18] ve internet sayfası içeriği veritabanı [19] gibi değişik alanlarda da kullanılmıştır. EWIRDB veritabanı askeri alanda kullanılan bir veritabanıdır ve günümüzde askeri alanda da XML ve internet teknolojilerinin kullanılması gerektiği belirtilmiştir [20, 21].

3.2. Hiyerarşik Verilerin Benzerliğinin Bulunması

Hiyerarşik verilerin benzerliği üzerine çok fazla araştırma bulunmaktadır. Bu araştırmaların bazıları sadece hiyerarşilerin benzerliğini, bazıları da hem hiyerarşi hem de değerlerin benzerliklerini bulmaya odaklanmıştır. Aşağıda bu konuda daha önce yapılan araştırmalar verilmiştir.

Myers [22] çalışmasında, iki ağacın benzerliğini bulmak için graf teorisine göre A ve B şeklindeki iki grafın ortak olan en uzun yollarını tanımlayan bir yöntem önermiştir. Bu çalışmada iki graf arasındaki benzerliğin, birbirine dönüştürebilmek için gerekli olan en kısa betiğin bulunması veya elemanların birbirlerine benzetilmesi için yapılması gereken iş olarak belirtilmiştir.

Chawathe ve arkadaşlarının [23] bir çalışmasında iki hiyerarşik verinin birbirine dönüştürülebilmesi için gerekli olan en az değişikliği, ağaç güncelleme mesafesi algoritması temelli, en az değişiklik için gerekli betiği bulan algoritma önerilmiştir. Bu çalışmada, algoritmanın bir ağacı diğerine benzetebilmesi için değişik adımlardan geçmesi gerekmektedir. Bu adımlar sırası ile: *güncelleme*, *dengeleme*, *ekleme*, *hareket ettirme*, *silme*, *alt elemanları dengeleme* ve *sonuçlandırma* adımlarıdır. Bu basamaklardan sonra betik oluşturulmakta, betikte bulunan adımlar ağaçlar arasındaki değişiklikleri vermektedir. Chawathe ve Garcia-Molina [24] diğer bir çalışmalarında bu algoritmanın daha da etkin olması için sezgisel yaklaşım eklemişlerdir.

Lee ve arkadaşlarının [25] XML dokümanları arasında benzerliklerin bulunması için yaptıkları çalışma bu konudaki ilk çalışmalardan biridir. Çalışmada, XML dokümanı yapıları genişletilmiş eleman vektörlerine dönüştürülmüş ve bu vektörler ile benzerlik hesaplanmıştır. Bu çalışmada belirtilen yöntem XML dokümanları arasında ortak ve benzer olan yolların tanımlanması temeline dayanmaktadır. Sık olarak tekrarlanan yolların belirlenmesinde, ardışık madencilik yaklaşımı kullanılmış ve çalışma sonuçları raporlanmıştır.

Nierman ve Jagadish [26] XML verileri arasındaki benzerliği tanımlamak için çalışma yapmışlardır. XML verisini etiketlendirilmiş ağaç olarak tanımlayarak güncelleme mesafesi yönteminden adapte edilmiş bir metot önerilmiştir. Bu metot ile XML verilerinin yapılarının birbirlerine olan benzerlikleri tanımlanmış ve sonuçları raporlanmıştır.

Ma ve Chbeir [27] çalışmalarında, hiyerarşik yapıda bulunan XML verilerinin benzerliklerinin tanımlanmasında içerik ve yapısal benzerliklere birlikte baktıklarını belirtmişlerdir. Tanımlanan benzerlik metodunun simetrik olmadığı vurgulanmış ve A XML verisinin B XML verisine benzerliği ile B XML verisinin A XML verisi arasındaki benzerliğin aynı olamayacağı belirtilmiştir. Karşılaştırmada kullanılan verilerin aynı şemaya sahip olduğu kabul edilmiş ve hiyerarşinin aşağısından – yukarıya doğru benzerlik hesaplaması yapan bir metot tanımlanmıştır. Çalışmada

önerilen metodun performansının daha iyi olmasını vektör temelli olmamasına bağlamıştır.

Chang ve arkadaşları [28] veri madenciliği ile internet dokümanlarını inceleyip dokümanlardan bilgi çıkarmaya çalışmışlardır. Çalışmada, internet dokümanlarının ağaç yapılarında tekrarlanan örüntüleri bularak benzer veriler ve kurallar bulunmaya çalışılmıştır. İnternet sayfalarının tüm yazılarının PAT (Practical Algorithm to Retrieve Information Coded in Alphanumeric) ağacına dönüştürülerek benzerliğin bulunabileceği önerilmiştir.

Miyahara ve arkadaşları [29] XML dokümanlarını tanımlamak için etiketlenmiş ağaç örüntüsü kullanmışlar ve XML dokümanları arasındaki benzerlikleri bu yöntemle belirlemeye çalışmışlardır. Çalışmalarında, sonuçlar raporlanmış fakat önerilen metodun etkin bir yöntem olmadığı belirtilmiştir.

Lee ve Lee [30] farklı şemalara sahip olan veya şemaları olmayan XML dokümanlarının ortak şemalarını belirlemeye çalışmalardır. Çalışmalarında dinamik ontoloji yöntemi kullanmışlardır. XML dokümanlarının benzerliklerinin hesaplanmasında öncelikle XML dokümanları içindeki etiketlerin benzerlikleri belirlenmiş ve benzer etiketler ortak etikete dönüştürülmüştür. Daha sonra XML dokümanında bulunan verilerin benzerlikleri bulunmuş ve bu benzerlik sonuçları gruplandırılmıştır. Verilerin benzerlikleri hesaplanırken veri tiplerinin uyumluluğu göz önüne alınmış ve benzerlikler *bire bir aynı, birbirine dönüştürülebilir, dönüşüm sonucunda bilgi kaybı olur* ve *birbirine dönüştürülemez* olarak dört gruba ayrılmıştır. Bu bilgiler kullanılarak verilerin bulundukları yerlerin XML dokümanlarında aynı olup olmadığının benzerliği bulunmuştur. Elde edilen benzerlikler belirli bir eşik değerinin üzerinde ise ortak bir şema oluşturulmuştur. Çalışma sonuçları raporlanmıştır.

Ceravolo ve arkadaşları [31] XML dokümanlarının benzerliklerini bularak bilgi çıkartmak için bulanık mantık tekniğini kullanmışlardır. Benzerliğin bulunmasında iki nesne arasındaki mesafenin hesaplanmasının yerine iki nesnenin özelliklerinin karşılaştırılmasının daha doğru olduğunu savunmuşlardır. Kullandıkları metotta

XML dokümanı matematiksel olarak modellenmiş ve tanımlanan eşitlik ile ağacın köküne göre ağacın yaprakları karşılaştırılmıştır. Bu çalışmada sadece XML dokümanlarının yapısal benzerlikleri bulunmaya çalışılmıştır.

Leung ve arkadaşları [32] XML dokümanlarının benzerliğini bulmak için veri madenciliği yaklaşımını kullanmışlardır. Çalışmada XML dokümanları etiketlendirilmiş ağaç olarak düşünülmüş, ağacın yapraklarının özellikleri benzerlik hesaplamasına dahil edilmemiş ve XML dokümanlarının sadece yapısal benzerlikleri bulunmaya çalışılmıştır. XML dokümanlarının benzerliğini bulurken ağacın yollarındaki benzerliklerin kullanılmasının benzerliği tam olarak yansıtmadığı savunulmuştur. Çalışmalarında kullanılan benzerlik hesaplamasında kullanılacak matematiksel eşitlik, ağacın tüm seviyelerindeki elemanların diğer seviyelerdeki elemanlarla karşılaştırılması sağlanmıştır. Ağacın belirli bir yolunda olmayan elemanlar da benzerlik hesaplanmasında göz önünde bulundurulmuştur.

Leung ve arkadaşları [33] diğer bir çalışmalarında, ardışık madencilik yaklaşımı ile başka bir XML dokümanı benzerlik metodu tanımlamışlardır. Bu çalışmada, XML dokümanlarının yollarının çözümlenmesi ve tekrarlayan yolların belirlenmesi için ardışık örüntü madenciliği kullanılması önerilmiştir. Bu yöntemde, XML dokümanları bir ön işlemden geçmekte ve benzerlik bulma yöntemi ile mantıksal ve yapısal benzerliğe sahip elemanlardan benzerlik kümesi oluşturulmaktadır. Elde edilen benzerlik kümesi, son bir işlemden daha geçirilmiş ve benzerlik sonuçları çıkarılmıştır.

Zhang ve arkadaşları [34] geleneksel ağaç güncelleme algoritmasını temel alan, kenar kısıtları, yol kısıtları, eleman kısıtları ve mantık kısıtları gibi kriterleri de kullanan bir metot önermişlerdir. Ağaç güncelleme mesafesi algoritması ile elde edilen sonuçların belirtilen kısıtlara göre kullanıcının belirlediği katsayılarla çarpılması ile yeni bir benzerlik metodu tanımlanmıştır. Çalışmada mantıksal kısıt için gerekli eşitlikler tanımlanmış fakat çalışmada bu kısıt kullanılmamıştır. Çalışma sonucu elde edilen sonuçlar raporlanmıştır.

Canfora ve arkadaşları [35] XML dokümanlarının benzerliklerinin bulunmasında alt ağaç tespit etme metodu önermişlerdir. Çalışmada, XML dokümanlarının içeriklerinin, yapılarının ve elemanların XML dokümanlarındaki yerlerinin benzerlikleri farklı eşitliklerle hesaplanmıştır. Bu çalışmada benzerlikler karşılaştırılan XML dokümanlarının ortak elemanlarından elde edilmiştir. Her üç benzerlik için farklı matematiksel eşitlik kullanılmış ve grafiksel olarak benzerlikler raporlanmıştır.

Xie ve arkadaşları [36] XML dokümanlarının birbirine en çok benzeyen kısımlarını belirlemek için yaptıkları çalışmada, XML dokümanlarının elemanlarının benzerlikleri bulurken ağaç güncelleme mesafesi algoritmasının temel alınmasını ve her bir elemanın seviyesinin benzerliğinin hesaplanmasında farklı katsayılarla çarpılması gerektiğini önermişlerdir. Bu yöntemle iki dokümanın birbirine en çok benzeyen özelliklerini daha etkin şekilde bulunabileceğini savunmuşlardır. Önerilen benzerlik bulma yönteminin ilk basamağında ağacın özetinin çıkarılmasının, daha etkili sonuçlar verdiği savunulmuştur. Özet ağaçlar oluşturulduktan sonra bu ağaçların benzerliğinin bulunmasında her eleman seviyesi için farklı katsayı belirlenebilmesi dikkate alınmış ve matematiksel olarak benzerlik değerleri hesaplanmıştır. Bu benzerlikler hesaplanırken iki ağacın birbirine benzemesi için ağaç güncelleme mesafesinde olduğu gibi *ekleme, çıkarma* ve *adını değiştirme* işlemlerinden hangisinin yapılması gerektiği de belirtilmiştir. Bu işlemler benzerlikte ağırlık değeri olarak kullanılmıştır. Tüm bu işlemler her bir ağaç elemanı için gerçekleştirilmiş ve en çok benzerlik değerine sahip elemanlar iki XML dokümanında birbirine en çok benzeyen elemanlar olarak tanımlanmıştır.

Patrick ve Vincent [37] bir XML dokümanı ile bir DTD (Document Type Definition) dokümanının birbirine benzerlikleri üzerine çalışmışlardır. Bu çalışmada, DTD dokümanı yapısı da bir ağaç şekline dönüştürülmüş ve ağaç güncelleme mesafesi kullanılarak benzerlikler tanımlanmıştır. Çalışmada, benzerlikleri bulurken her bir eleman için metot içinde tanımlanan ağırlıklar ile çarpılarak ağaçtaki yerlerinin benzerlikleri de göz önünde bulundurulmuştur. Çalışmada kullanılan yöntem ve çalışma sonuçları açıklanmıştır.

Bertino ve arkadaşları [38, 39] XML dokümanlarının DTD dokümanlarına ne kadar benzediğini belirlemek için yaptıkları çalışmada, Patrick ve Vincent [37]'in önerdikleri yönteme benzer bir yöntem önermişler ve kullanılan metodu açıklamışlardır.

Kriegel ve Schönauer [40] XML dokümanlarını grafa dönüştürerek, grafik üzerinde kenar eşleştirme algoritması ile benzerliğinin bulunabileceğini önermişlerdir. Ağaç güncelleme mesafesi algoritmasına alternatif olarak önerilen yöntemde, grafın ve grafın özelliklerinin benzerlikleri dikkate alınmıştır. Tanımlanan algoritmada benzerliklerin belirlenmesinde etkili olacak bir çarpan tanımlanmış ve bu çarpan ile yöntemin değişik uygulamalara adapte edilebileceği savunulmuştur. Bu yöntemin sadece XML dokümanlarının benzerliğinin bulunmasında değil moleküler biyolojide, resim bulmada da kullanılabileceği belirtilmiştir. Resimlerin benzerliğinin bulunması üzerine yapılan çalışmalar raporlanmıştır.

Lee ve Park [41] hiyerarşik yapıya sahip olan internet sayfalarının benzerliği üzerinde çalışmışlar ve benzerlik bulmak için bir yöntem önermişlerdir. Bu yöntem için yapısal olarak XML dokümanlarının ön işlemden geçmesi, elemanların belirlenmesi ve ortak özelliklerin belirlenmesi gerektiği anlatılmıştır. Ortak ağaç yolları belirlenmiş ve tanımlanan metot ile benzerlik bulunmuştur. Çalışmada, internetten rasgele seçilen sayfalar üzerinde çalışma sonuçları raporlanmıştır.

3.3. Hiyerarşik Verilerin Gruplanması

Hiyerarşik verilerin gruplandırılması konusu, XML dokümanlarının çok sık kullanılır olmasından sonra hız kazanmış ve bu konuda birçok yöntem önerilmiştir. Aşağıda bu konuda yapılan çalışmalar anlatılmıştır.

Nayak ve Iryadi [42] homojen olmayan XML dokümanlarının gruplandırılması için XMine adında bir yazılım önermiştir. XML dokümanlarının gruplandırılmasında izledikleri yöntem birkaç basamaktan oluşmaktadır. Tanımlanan yöntemde öncelikle XML dokümanları yapısal olarak analiz edilmiş, ardından doküman içinde bulunan elemanların analizi yapılmıştır. Bu işlemlerin ardından ağacın birbirine en çok benzeyen yolları bulunmuş ve şema benzerliklerine bakılmıştır. Bu bilgiler

kullanılarak gruplandırma işlemi yapılmış ve XML dokümanları değişik gruplara ayrıştırılmıştır. Bu işlemler gerçekleştirilirken kullanılan yöntemde kendilerinin tanımladıkları benzerlik bulma yöntemi kullanılmıştır. Çalışma sonuçları raporlanmış ve önerilen yöntemin etkin bir yöntem olduğu savunulmuştur.

Shen ve Wang [43] şeması olmayan dokümanların gruplandırılması için bir yöntem önermiştir. Gruplandırmada kullanılacak benzerlik bulma yöntemi yine bu çalışmada tanımlanmıştır. Benzerlik bulma yönteminde, isimlerin benzerliği, ağaç yolu benzerliği, içerik benzerliği ayrı ayrı hesaplanmakta ve bir eşitlikle bütün benzerlikler birleştirilmektedir. Tanımlanan benzerlik bulma yöntemi ile bütün dokümanların birbirleri ile olan benzerlikleri hesaplanmakta ve benzerlik matrisi oluşturulmaktadır. Oluşturulan benzerlik matrisi ve hiyerarşik gruplama algoritması ile XML dokümanı grupları elde edilmektedir.

Nayak ve Xu [44, 45] ile Nayak ve Brisbane [46] homojen olmayan XML dokümanlarının XCLS (XML Document Clustering with Level Similarity) olarak adlandırdıkları yöntemle gruplandırılabileceğini belirtmişlerdir. Bu çalışmalarda, XML dokümanlarının elemanları seviyelere göre gruplandırılarak dokümanlarının benzerliği tanımlanmış ve bu benzerlikler kullanarak XML dokümanlarının gruplandırılması yapılmıştır. Bu işlemlerin gerçekleştirilmesinde kullanılan denklemler yine aynı çalışmada verilmiştir.

Popovici ve arkadaşları [47] XML dokümanlarından bilgi elde etmekte kullanılmak amacı ile dokümanların gruplandırılmasında ardışık şema yöntemini önermişlerdir. Homojen olmayan XML dokümanlarının indekslenmesi ve dokümanlar üzerinde arama yapılabilmesi için önerdikleri yöntemle, XML dokümanlarının yapısal olarak benzer olanlar, benzer alt parçalar ve benzer ağaç yolları bulunmuştur.

Lee ve Hwang [48, 49] XML dokümanlarının gruplandırılmasında üç boyutlu *Bitmap* indeksleme metodu önermişlerdir. Önerilen yöntemde, XML dokümanı öncelikle analiz modülüne girmekte ve burada analiz edilmektedir. Analiz sonucunda ortak ağaç yollarının indekslenmesi yapılmaktadır. XML dokümanları ağaç yollarına göre,

ağaç yolları da *Bitmap* indeksleme yöntemine göre gruplandırılarak, dokümanlara hızlı erişme imkanı sağlandığı belirtilmiştir.

Hwang ve Ryu [50-52] ardışık örüntüsü madenciliği algoritması ile XML dokümanlarının gruplandırılabileceğini belirtmişlerdir. Çalışmada, şeması olmayan XML dokümanlarının bu yöntemle etkili bir şekilde gruplandırılabileceği belirtmişlerdir. Önerilen yöntemde, XML dokümanları ağaç halinde açılarak numaralandırılmış ve etiketlendirilmiştir. Aynı isimli elemanlara aynı etiketler verilmiştir. Bu etiketler kullanılarak ortak ağaç yolları belirlenmiştir. Bu işlem sonunda, her bir etiketlenen elemanın benzerliği bulunmuş ve bu benzerlikler kullanılarak dokümanların benzerlikleri hesaplanmıştır. Elde edilen değerlere göre gruplar belirlenmiştir. Gerekli durumlarda bu işlemler tekrarlanarak daha etkin çözümler elde edilmeye çalışılmıştır. Çalışmada, örnek veriler üzerinde elde edilen sonuçlar raporlanmıştır.

Wang ve arkadaşları [53] bilgi çıkarmak amacı ile XML dokümanların gruplandırması üzerine çalışma yapmışlardır. Çalışmalarında, ICX (Indirect Clustering for XML Documents) adında bir yöntem önermişlerdir. Gruplandırma için ağaçların benzerlikleri ağaç yolları olarak ifade edilmiş ve C-Means temelli bir yöntem tanımlanmıştır. Çalışmada, çok sayıda XML dokümanının hızlı bir şekilde gruplandırılabildiği belirtilmiştir.

Dalamagas ve arkadaşları [54-56] XML dokümanlarının özeti çıkarılarak ağaç güncelleme mesafesi algoritmasından adapte edilen bir yöntemle gruplandırılabileceğini önermişlerdir. Önerilen yöntemde, XML dokümanları özetlenmiş ve ardından ağaç güncelleme mesafeleri hesaplanmıştır. Hesaplama sonucu elde edilen benzerlikler, tek bağlantılı hiyerarşik gruplandırma algoritması kullanılarak dokümanların gruplandırılmasında kullanılmıştır.

Hatano ve arkadaşları [57] internette XML dokümanı olarak bulunan bilgilerin bir bölümünü hızlıca bulabilmek için gerekli gruplama yöntemi önermişlerdir. Önerdikleri yöntemde, XML formatındaki sorgular parçalanıp DOM (Document Object Model) ağacı haline getirilmiş ve yazıların analizi yapılmıştır. Analiz

sonucunda XML dokümanları arasındaki benzerlikler bulunmuştur. Doküman benzerliği için kullanılan yöntem çalışmada tanımlanmıştır.

Sanz ve arkadaşları [58] XML veritabanlarındaki verilerin otomatik olarak gruplandırılması ve şemaların yönetimi üzerine metot önermişlerdir. Çalışmada, veritabanına eklenen bir doküman öncelikle veritabanında bulunan dokümanlarla yapısal olarak benzeyip benzemediği belirlenmiş, benzerlik oranı eşik değeri üzerinde olanlar benzer dokümanın yanına eklenmiştir. Çalışmada tanımlanan metodun ticari bir veritabanında kullanıldığı belirtilmiştir.

Costa ve arkadaşları [59] XML dokümanlarının ağaç olarak gösterimi ile yapısal benzerliklerin bulunabileceğini ve dokümanların gruplandırılabileceğini belirtmişlerdir. Çalışmada, XML dokümanlarının yapısı matrise dönüştürülmüş ve bu matrisler birleştirilerek gruplar oluşturulmuştur. Önerilen yöntemde, tüm dokümanlar yukarıda belirtilen işlemlerden geçirilmesi ve tüm dokümanlar bir gruba dahil oluncaya kadar devam etmesi önerilmiştir.

4. GENİŞLEYEBİLİR İŞARETLEME DİLİ (XML)

Bu bölümde XML ve XML teknolojileri hakkında genel bilgi verilecek ve örneklerle açıklanacaktır.

1960'lı yıllarda IBM tarafından önerilen, dokümanların organizasyonu için GML (Generalize Markup Language) işaretleme dili temel alınarak SGML (Standard Generalized Markup Language) işaretleme dili geliştirilmiş ve bu uzun süre standart doküman tanımla dili olarak kullanılmıştır. XML dili de W3C (World Wide Web Consortium) tarafından, SGML diline uygun bir işaretleme dili olarak ortaya çıkmıştır [60-64]. Bu dilin geliştirilmesinde HTML (Hyper Text Markup Language) dilinin çok hızlı şekilde kullanılması ve bu kullanımın çok karmaşık ve kontrol edilemez boyutlara gelmesi etkili olmuştur. Çünkü HTML dili dokümanların tanımlanmasında veriler ile verinin gösterimini ayırmamakta ve verilerin saklanmasında sabit etiketler kullanmaktadır. XML dili ile sabit etiket kullanımı gereksinimi ortadan kaldırılmış ve istenilen etiketin tanımlanarak verilerin saklanması mümkün olmuştur. Ayrıca HTML dokümanlarının iyi organize edilmiş olmasına gerek yoktur. HTML dilinde, oluşturulan bir etiketin kapatılmasına gerek yoktur. Örneğin, bir HTML dokümanında paragraf başı tanımlamak için <*p*> etiketi oluşturulunca bu etiketin kapatılmasına ihtiyaç yoktur. XML dokümanlarında ise oluşturulan her etiket kapatılmalıdır. XML dokümanı içinde paragraf oluşturmak için <*p*> etiketi tanımlanırsa ardından </*p*> ile kapatılması gerekmektedir.

XML dilinin geliştirilmesinde aşağıda belirtilen hedefler göz önünde bulundurulmuş ve bu hedefleri sağlayacak işaret dili tanımlanmıştır:

- İnternet üzerinden rahatlıkla kullanılabilmelidir,
- Birçok uygulama tarafından kolayca kullanılabilmelidir,
- SGML diline uyumlu olmalıdır,
- Dokümanlarının işlenmesi için yazılması gereken programlar kolayca hazırlanabilmelidir,
- Dilde istisnai durumlar az olmalı veya hiç olmamalıdır,
- Dokümanlar insanlar tarafından okunabilir ve kolay anlaşılır olmalıdır,

- Doküman tasarımı çok kolay yapılabilmelidir,
- Doküman tasarımı kurallara uygun ve kısa olmalıdır,
- Kolayca doküman oluşturulabilmelidir,
- Tanımlanması gereken etiketler kısa ve özlü olmalıdır.

XML dili yukarıda belirtilen hedefler doğrultusunda tanımlanmış, yapısını değiştiren yeni özelliklere ve yeni bir sürüm tanımlamaya ihtiyacı duyulmamıştır. XML dili üzerine kurulan tüm teknolojilerin gelişmesinde, ilerlemesinde ve standartlaşmasında XML dilinin hala ilk sürümünün geçerli olması büyük önem taşımaktadır. XML dilinde tanımlanacak yeni bir sürüm, XML diline dayanan tüm teknolojilerin değişmesine sebep olacaktır.

XML dilinin tanımlanması ve kullanılmasının ardından XML dilinin geliştirilmesinin hedefleri doğrultusunda günümüzde pek çok avantajları ortaya çıkmıştır:

- Evrensel kod birliğini desteklediği için uluslararası tüm diller ile kullanılabilmektedir,
- İşletim sistemi ve donanım bağımlılığı olmadığı için her hangi bir platforma taşınabilir ve herhangi bir platformda işlenebilir,
- İnsanlar tarafından okunabilir bir formatta olduğu için geliştiriciler dokümanda bulunan hataları kolay bir şekilde bulunabilir,
- XML dilinde bir değişiklik olmadığı için XML dokümanlarının işlenmesinde kullanılabilecek birçok RAHAT (Rafta Hazır Ticari Ürün) bulunmaktadır. Bu sayede XML dokümanlarının işlenmesi için geliştiricilerin kendi yazılımlarını hazırlamalarına gerek kalmamaktadır [63].

Günümüzde verilerin saklanması için XML dokümanlarının kullanılması konusunda pek çok çalışma yapılmakta hatta bazı ilişkisel veritabanları XML dokümanlarını yönetebilir hale gelmektedir. XML dokümanlarının saklanması ve XML dokümanları üzerinde sorgulamaların yapılabilmesi için yapılan çalışmalar sonuçlanmaya başlamış ve ilişkisel veritabanları ile yarışabilmek için ilk adım atılmıştır.

Aşağıda örnek XML dokümanı verilmiş ve ardından XML dokümanı oluşturmak için gerekli kurallar anlatılmıştır.

```
<?xml version="1.0" encoding="ISO-8859-9"?>
<!—XML dokümanı içinde tanımlanan açıklama -->
<tez>
        <isim>Ozgur Yurekten</isim>
        <konu ingilizce= "hayır">Hiyerarşik Veriler<konu>
        <yayınlandı/>
        <üniversite>
                <üniversite-adı>Gazi Üniversitesi</üniversite-adı>
                <anabilim-dalı>Bilgisayar Mühendisliği</anabilim-dalı>
        </üniversite>
        <özet>
                <![CDATA[
                        Bu çalışmada hiyerarşik veriler üzerinde bir ...
                ]]>
        </özet>
</tez>
```

XML dokümanı oluşturmak için dokümanın en üstünde *<?xml version="1.0" ?>* tanımlanmalıdır. Bu ifade tanımlamadan önce, dokümanda açıklama dahi bulunmamalıdır. Bundan sonra veriler etiketler arasında tanımlanmalıdır. Etiketler < ve > karakterleri ile çevrelenmelidir. Örnekte *tez* adında etiket tanımlanmış ve bu etiket içinde *isim, konu, yayınlandı, üniversite* ve *özet* adında etiketler eklenmiştir. Veriler bu etiketler arasında bulunmaktadır. Veriler hiyerarşik bir şekilde tanımlanması gerektiği için her bir verinin bir kök etiketi bulunması ve diğer etiketlerin kök etiketin altında veya daha alt etiketlerinde tanımlanması gerekmektedir. Bir etiket arasında, başka etiketlerden istenildiği kadar tanımlanabilmektedir. Tanımlanan her etiketin veri girildikten sonra kapatılması gerekmektedir.

İşaretleyici olan veya verisi bulunmayan etiketler aynı etiket içinde kapatılabilmektedir. Örnekte *yayınlandı* etiketi veri içermediği için <*yayınlandı/*> olarak tanımlanmıştır. Burada etiket tanımlanmış ve aynı etiket içinde kapatılmıştır. Bu gösterim <*yayınlandı*></*yayınlandı*> ile aynı anlama gelebilmektedir.

Etiketler arasındaki veriler XML dili içinde kullanılan özel karakterleri (< , > v.b.) içermesi veya etiket içinde tanımlanan yazının aynen (paragraf, boşluk ve satır başlarının yazıldığı şekilde) kullanılması istendiğinde, bu veriler <*!CDATA[[* ile *]]*> karakter dizisi arasına yazılmalıdır.

Etiketlerin verileri olabileceği gibi bir takım özellikleri de olabilir. Tanımlanan bir özellik sadece o etikete özeldir ve bu özellik etiket içinde bir kere kullanılabilir. Örneğin *konu* isimli etiket bir veriye sahip olmasının yanında tezin içeriğinin İngilizce olup olmadığını belirtmek için bu etikete *ingilizce* adında bir özellik eklenmiştir.

4.1. XML Dokümanlarında İsim Uzayı

XML dokümanlarında kullanılacak etiketler geliştiriciler tarafından istenildiği gibi tanımlanabilmektedir. Fakat tanımlanan etiketlerin başka bir XML dokümanında başka bir amaçla kullanılıyor olma ihtimaline karşı etiketlerin isim uzayı içinde tek olması gerekliliği ortaya çıkmıştır [63, 64]. XML dilinde isim uzayı için bir tanımlama yapılmamış fakat tanımlama yapılması da engellenmemiştir. Bunun için XML dokümanı hazırlanırken isim uzaylarının da tanımlanmasının ve standartlaştırılmasının gerekliliği ortaya çıkmıştır. Bu konuda W3C içinde XML dokümanlarında isim uzayı tanımlamak için bir grup oluşturulmuş ve XML dokümanlarında isim uzayı kullanımı standartlaştırılmıştır. Etiketlere isim uzayı oluşturulmak için URI (Uniform Resource Identifier) kullanılması önerilmiştir. URI, uygulama için anlamlı ve benzeri olmayan bir yazı olmalıdır. Bu yazı bir internet adresi olabileceği gibi herhangi bir yazı da olabilir.

<onek:tez xmlns:onek='http://www.gazi.edu.tr/bm24290291/thesis'> … </onek:tez>

Yukarıda *tez* adında bir etiket tanımlanmış ve bu etiket *onek* kısaltması ile tanımlanan bir isim uzayında bulunmaktadır. Bu isim uzayının adı da *xmlns* şeklinde bir sabit kullanılarak *http://www.gazi.edu.tr/bm24290291/thesis* olarak tanımlanmıştır. Verilen bu ismin dünyada tek olacağı kabul edilmiştir.

İsim uzayı sadece etiketlerde değil etiketlerin özellikleri için de kullanılır. İsim uzayının adı boş olamaz. Adı yukarıda belirtildiği gibi bir URI olarak tanımlanmalıdır. İsim uzayları için istenilen kısaltmalar kullanılabilir. Yukarıda *onek* olarak tanımlanan kısaltma yerine *gazi* kullanmak istendiğinde *xmlns:onek* yerine *xmlns:gazi* kullanılmalı ve tüm *onek* kısaltmaları *gazi* olarak değiştirilmelidir. Aşağıda isim uzayı kullanılarak hazırlanan örnek bir XML dokümanı bulunmaktadır.

```
<?xml version="1.0" encoding="ISO-8859-9"?>
<!—XML dokümanı içinde tanımlanan açıklama -->
<gazi:tez xmlns=" http://www.gazi.edu.tr/bm24290291/thesis">
        <gazi:isim>Ozgur Yurekten</ gazi:isim>
        < gazi:konu ingilizce= "hayır">Hiyerarşik Veriler<gazi:konu>
        <gazi:yayınlandı/>
        <gazi:üniversite>
                <gazi:üniversite-adı>Gazi Üniversitesi</ gazi:üniversite-adı>
                <gazi:anabilim-dalı>Bilgisayar Mühendisliği</gazi:anabilim-dalı>
        </gazi:üniversite>
        <gazi:özet>
                <![CDATA[
                        Bu çalışmada hiyerarşik veriler üzerinde bir ...
                ]]>
        </gazi:özet>
</gazi:tez>
```

Yukarıda verilen örnek, XML dokümanın isim uzayı kullanılmış halidir. Bu dokümanda isim uzayı *http://www.gazi.edu.tr/bm24290291/thesis* olarak kullanılmış ve bu isim uzayının kısaltması olarak *gazi* kelimesi kullanılmıştır. Bu sayede başka

XML dokümanları *tez* kelimesini etiket olarak kullanmış olsa bile isim uzayı ile birbirinden ayırt edilebilir olmuştur.

4.2. XML Doküman Tipini Tanımlama (DTD)

XML dokümanlarının yapısını tanımlamak için ilk önce DTD dokümanı kullanılmıştır. Bu dokümanda kök etiket ve kök etiketin altında tanımlanan etiketlerin isimleri, genel olarak tipleri ve kaç kere kullanılacağı bilgisi bulunur. XML dokümanlarını tanımlamak amacı ile oluşturulan bu dokümanın kendi yapısı standart XML yapısına uygun değildir [64]. Tanımlanan sabit değişkenler ile doküman içeriği tanımlanır. Örnek DTD dokümanı aşağıda verilmiştir.

```
<!DOCTYPE tez  [
<!ELEMENT tez (isim, konu, yayınlandı, üniversite, özet)>
<!ELEMENT üniversite (üniversite-adı, anabilim-dalı)>
<!ELEMENT üniversite-adı (#PCDATA) >
<!ELEMENT anabilim-dalı (#PCDATA) >
<!ELEMENT isim (#PCDATA) >
<!ELEMENT konu (#PCDATA) >
<!ELEMENT yayınlandı (#PCDATA) >
<!ELEMENT özet (#CDATA) >
<!ATTLIST konu ingilizce CDATA #REQUIRED >
]>
```

XML dokümanını tanımlamak için başlangıçta *!DOCTYPE* etiketi tanımlanır. Etiketin yanında XML dokümanında bulunacak kök etiket tanımlanır. Kök etiketin altında bulunan tüm etiketler *!ELEMENT* ile tanımlanır. Etiketlerin hiyerarşisi bu etiketlerle belirlenir. Örneğin *üniversite* etiketinin yapısında *üniversite-adı* ve *anabilim-dalı* etiketleri bulunur. Etiketler verilere sahipse bunlar *#PCDATA* (Parsed Character Data) veya *#CDATA* (Character Data) olarak tanımlanır. *#PCDATA* sabiti etiket arasına yazılan verinin işlemden geçirileceğini belirtir. *#CDATA* sabiti etiket arasında yazılan verinin işlenmeden, tanımlandığı şekilde kullanılacağını belirtir.

Örnekte *konu* etiketi arasındaki veri işlenerek okunacak fakat *özet* etiketi arasındaki veri XML dokümanında tanımlandığı şekilde kullanılacaktır.

<?xml version="1.0" encoding="ISO-8859-9"?>
<!—XML dokümanı içinde tanımlanan açıklama -->
<! DOCTYPE tez SYSTEM "tez.dtd">
<tez>
....
</tez>

DTD içeriği XML dokümanlarının içerisinde veya ayrı bir dokümanda tanımlanabilir. DTD verisi ayrı bir dokümanda tanımlanırsa, yukarıda tanımlanan XML dokümanında olduğu gibi, dokümanın içinde DTD dokümanının bulunduğu yolun ve adının yazılması gerekmektedir.

4.3. XML Doküman Şeması Tanımlama (XSD)

XML dokümanlarının yapısını tanımlamak için kullanılan XSD dokümanları DTD dokümanlarına alternatif olarak geliştirilmiş ve W3C tarafından standartlaştırılmıştır. XSD dokümanının tanımlanmasının amacı [64]:

- XML dokümanında bulunan etiketlerin ve özelliklerinin tanımlanması,
- Etiketlerin alt etiketlerinin tanımlanması,
- Etiketin alt etiketlerinin sırasının ve sayısının belirlenmesi,
- Etiketin veriye sahip olup olamayacağının belirlenmesi,
- Etiketlerin ve özelliklerinin veri tiplerinin belirlenmesi,
- Etiketlerin ve özelliklerinin varsayılan değerlerini ve sabit değerlerinin belirlenmesidir.

XSD dokümanları DTD dokümanlarından daha zengin ve güçlü özelliklere sahiptir. XSD dokümanları gelecekte yapılacak değişikliklere açıktır, genişletilebilir bir yapıya sahiptir. XSD dokümanları XML dili ile yazılır ve etiketlerin veri tipleri tanımlanabilir. Ayrıca XSD dokümanları isim uzaylarını desteklemektedir. XSD dokümanları ile XML dokümanının içeriğinin hangi tip verilere sahip olacağı

belirlenebilir, verinin doğruluğu kontrol edilebilir, girilebilecek veriler ve veri formatları tanımlanabilir, veritabanı ile haberleşmeyi kolaylaşır ve farklı tiplerde bulunan veriler birbirlerine dönüştürülebilir. Bütün bu özellikler DTD dokümanlarında bulunmadığı için XSD dokümanları kullanılmaya başlamıştır.

XSD dokümanını tanımlamak için yine XML formatı kullanılmaktadır. Dokümanın hazırlanması için gerekli olan etiketler W3C tarafından tanımlanmıştır. Bu etiketler kullanılarak XML dokümanında bulunacak etiketlerin isimleri, veri tipleri, tekrarlanma sayıları, sıraları ve doğrulama yöntemleri belirlenir. Aşağıda örnek XSD dokümanı içeriği verilmiştir:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace=xmlns="http://www.gazi.edu.tr/bm24290291/thesis"
xmlns="http://www.gazi.edu.tr/bm24290291/thesis"
elementFormDefault="qualified">
   <xs:element name="tez">
      <xs:complexType> <xs:sequence>
            <xs:element name="isim" type="xs:string"/>
            <xs:complexType name="konu" type="xs:string">
               <xs:attribute name="ingilizce" type="xs:boolean"/>
            </xs:complexType>
            <xs:element name="yayınlandı" type:boolean/>
            <xs:element name="özet" type:string/>
            <xs:complexType name="üniversite" type="xs:string">
               <xs:sequence>
                  <xs:element name="üniversite-adı" type:string/>
                  <xs:element name="anabilim-dalı" type:string/>
               </xs:sequence>
            </xs:complexType>
         </xs:sequence> </xs:complexType>
   </xs:element>
</xs:schema>
```

Yukarıda XSD etiketleri *http://www.w3.org/2001/XMLSchema* isim uzayında tanımlanan *xs* kısaltması ile kullanılmıştır. XSD dokümanın tanımlanması için *schema* etiketi, kök etiket olarak tanımlanması gerekmektedir. Dokümanda *xmlns="http://www.gazi.edu.tr/bm24290291/thesis"* ile tanımlanan satır dokümanda isim uzayı ile belirtilen etiketlerin bu isim uzayında varsayılması gerektiğini belirtir. Dokümanda tanımlanan *elementFormDefault="qualified"* satırı ile doküman içinde kullanılan bütün etiketlerin bir isim uzayında olması gerektiği belirtilmiştir.

XSD dokümanı etiketleri *http://www.w3.org/2001/XMLSchema* isim uzayı içinde bulunmaktadır. Bu isim uzayında tanımlanmış olan *schema* etiketi, kök olarak kullanılır. Bunun içine *element* adı ile basit veya karmaşık değişik etiketler tanımlanabilir. Etiketlerin hangi isim uzayında olacağını belirtmek için *targetNamespace* özelliği tanımlanır. Örnekte, oluşturulacak olan etiketlerin isim uzayının *http://www.gazi.edu.tr/bm24290291/thesis* olacağı belirtilmiştir.

XML dokümanında tanımlanacak olan etiketlerden kök etiket, *element* etiketi ile belirtilmiştir. Bu etiketin adının *tez* olacağı *name* özelliği ile tanımlanmıştır. Her bir etiket bir nesne olarak düşünüldüğü için basit yapıda ise (yazı, tarih, sayı v.b.) *basit tip*, basit yapıda değilse *karmaşık veri tipi* olarak adlandırılmaktadır. Kök etiket, birden fazla etiket içereceği için *karmaşık veri tipi*ne sahiptir. Bunun için *complexType* etiketi arasında diğer etiketler tanımlanmıştır.

Hazırlanan XSD dokümanın XML dosyasında kullanılması için dokümanın içindeki kök etikete *xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"* özelliği eklenmelidir. Hazırlanan XSD dokümanın adı *tez.xsd* olduğu varsayılırsa, *xsi:schemaLocation="http://www.gazi.edu.tr/bm24290291/thesis tez.xsd"* özelliği yine kök etikete özellik olarak eklenmedir. Bu tanımlama yapıldıktan sonra XML dokümanı işlenirken XSD dokümanında tanımlı olan kurallara uyulup uyulmadığı kontrol edilir ve bir uygunsuzluk varsa hata raporlanır. Bu özellik XML verilerinin güvenilirliğini sağlama açısından kritiktir. Bu özellik kullanılarak rahatlıkla veritabanları ile veri paylaşımı yapılabilir.

XSD dokümanında basit elemanların tanımlanabildiği daha önce vurgulanmıştı. Basit elemanı tanımlamak için *<xs:element name="etiket adı" type="veri tipi" default="varsayılan değer"/>* şeklinde bir satır eklenir. Burada xs kısaltması, W3C XSD şemasında tanımlanan etiketlerin bulunduğu isim uzayını temsil etmektedir. "*name"* özelliği kısmına hedef XML dokümanında tanımlanacak olan etiketin adı, *type* özelliği kısmına verinin tipi yazılır. Tipik veri tipleri XSD şemasında tanımlanmıştır (*xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time*). "*default"* özelliği ise bu alan tanımlanmadığında varsayılan değerin ne olacağını belirtir. Etiketlere özellik eklemek için dokümana *<xs:attribute name="etiket adı" type="veri tipi" default="varsayılan değer"/>* satırı eklenir. Özelliklere girilmesi gereken değerler basit elemanlara girilebilecek değerler ile aynıdır.

```
<xs:element name= "sayfa-sayısı">
        <xs:simpleType> <xs:restriction base= "xs:integer">
                <xs:minInclusive value= "50"/>
                <xs:maxInclusive value= "220"/>
        </xs:restriction> </xs:simpleType>
</xs:element>
<xs: attribute name= "ingilizce">
        <xs:simpleType> <xs:restriction base= "xs:string">
                <xs:enumeration value= "evet"/>
                <xs:enumeration value= "hayır"/>
        </xs:restriction> </xs:simpleType>
</xs:attribute>
<xs:element name= "üniversite-adı">
        <xs:simpleType> <xs:restriction base= "xs:string">
                <xs:pattern value= "([A-Z])*"/>
        </xs:restriction> </xs:simpleType>
</xs:element>
<xs:element name= "konu">
        <xs:simpleType> <xs:restriction base= "xs:string">
                <xs:minLength value= "10"/>
```

```
                <xs:maxLength value= “255”/>
        </xs:restriction> </xs:simpleType>
</xs:element>
```

Yukarıda XSD dokümanlarında sınırlamaların nasıl tanımlanabileceğine ait bir örnek bulunmaktadır. Basit elemanlara ve özellikler için tanımlanan veri tiplerine sınırlamalar eklenebilir. Bunun için XSD şeması etiketleri arasında tanımlı olan *restriction* etiketi kullanılır.. Bu örnekte sayı değerlerinin minimum ve maksimum değerleri verilmekte, yazı özelliğine sahip değerlerin alabilecekleri değerler kümesi sunulmakta, yazıların içinde bulunması gereken karakterler, karakter dizilişleri tanımlanmakta ve yazıların uzunlukları belirlenmektedir. Örnekte verilen *sayfa-sayısı* elemanının alabileceği değer 50 ile 220 aralığındadır. Örnekte verilen *ingilizce* özelliğinin alabileceği değerler *“evet”* ve *“hayır”* yazılarıdır. Yazı tipinde değeri olan elemanların, yazının içinde bulunabilecek karakter dizisi ve yapısı tanımlanabilmektedir. Örnekte *üniversite-adı* olarak verilen elemanın sadece büyük harflerden oluşabileceği belirtilmiştir. *konu* adındaki elemanın en az 10 karakter ve en fazla 255 karakter olabileceği ifade edilmiştir.

Verilen kısıtlamaların doküman içerisindeki kontrolü, XML dokümanını işleyen yazılımlar tarafından yapılmakta ve bu yazılımlar uygunsuzlukları raporlamaktadır.

4.4. XML Dokümanlarının İşlenmesi

XML dokümanlarının işlenmesi için temel olarak iki farklı yöntem bulunmaktadır [63]. Bunların birincisi DOM (Document Object Model) modelidir. Bu modelde, XML dokümanın içeriğinde bulunan her eleman bir nesneye dönüştürülür ve bu nesneler birbirleri ile ağaç oluşturacak şekilde bağlanır. Bu model ile işlenen XML dokümanının çıktısı, tüm elemanları hiyerarşik olarak dizilmiş bir nesnedir. Bu nesne üzerinde kökten başlayarak istenilen elemana ulaşılabilir ve özellikleri öğrenilebilir. Bu model aynı zamanda XML dokümanının oluşturulmasında da kullanılabilir. XML dokümanında yer alan veriler DOM ağacı nesnesine dönüştürülerek dosyaya yazılır. Bu işlemi yapmak için hiyerarşiye eleman ekleme, hiyerarşiden eleman çıkarma ve herhangi bir elemanı güncelleme yeteneklerine sahip olunmalıdır. Bu model W3C

tarafından yönetilmekte ve standartları belirlenmektedir. XML dokümanı oluşturulurken veya XML dokümanı işlenerek DOM nesnesine dönüştürülürken dokümanın doğruluğunu kontrol etmek için tanımlanan XSD veya DTD dokümanları kullanılmaktadır.

XML dokümanlarının işlenmesinde kullanılabilecek diğer bir yöntem de XML dokümanının etiketleri ve verileri okunurken okunan veri hakkındaki bilgilerin bu modeli kullanan yazılıma gönderilmesi şeklindedir. Bu modelde DOM ağacı gibi bir nesne oluşturulmaz sadece dokümanın üzerinden geçerken içeriği hakkındaki bilgiler yazılıma geri dönülür. Bu model, DOM ağacı yerine kişisel bir yapının oluşturulmasına olanak sağlar.

Her iki modelin birbirine göre farklı avantajları vardır. DOM modelinde XML dokümanları tamamen ağaca dönüştürüldüğü için çok basit ve sadedir. Fakat dokümanların büyüklükleri göz önüne alındığında çok büyük dokümanlardan ağaç yapısı oluşturmak için çok fazla hafızaya ve zamana ihtiyaç duyulur. Diğer modelde ise ağaç oluşturulmadığı için fazla hafızaya ihtiyaç duyulmadan hızlı bir şekilde işlem tamamlanabilir. Fakat dokümanda bulunan bütün veriler hazırlanan yazılım tarafından kontrol edilmesi gerekmekte ve yapılan işlem daha karmaşık olmaktadır.

Günümüzde, bir modeli veya her iki modeli kullanan bir çok XML dokümanı işleyen yazılım bulunmaktadır. Bu yazılımlardan bir çoğu açık kaynaktır. Örneğin Java programlama dilinin içinde standart olarak bu iki modeli de kullanarak XML dokümanını işleyebilecek bir alt yapı sunulmuştur. JAXP (Java API for XML Processing) teknolojisi ile DOM ve SAX (Simple API for XML) alt yapıları kullanılarak XML dokümanları oluşturulabilmekte ve XML dokümanları işlenebilmektedir [63].

5. HİYERARŞİK VERİLERİN VERİTABANI İLE MODELLENMESİ

Bu bölümde, hiyerarşik verilerin veritabanı ile modellenmesi alternetifleri anlatılacaktır. Alternatiflere geçmeden önce ilişkisel veritabanı tabloları ile XML dosyaları arasındaki farklar anlatılacak ve ilişkisel veritabanı modellemelerinde kullanılan normal formlar hakkında özet bilgi verilecektir.

İlişkisel veritabanlarında saklanan veriler ile XML dokümanlarından saklanan veriler yapısal olarak farklılık göstermektedir. İlişkisel veritabanlarında veriler tablosal olarak saklanıp yönetilirken XML dokümanlarında veriler hiyerarşilerini belirleyecek şekilde ağaç formatında saklanmaktadır. İlişkisel veritabanları ile XML dosyalarının yönetilmesi arasındaki farklar Çizelge 5.1'de verilmiş [64] ve çizelgenin sonunda ayrıntılı olarak açıklanmıştır.

Çizelge 5.1. İlişkisel veritabanı ve XML dosyaları arasındaki farklar

Özellik	**İlişkisel Veritabanı**	**XML Dosyası**
Fiziksel ortam	Kontrollü veri saklama fiziksel ortamı vardır.	Dosyalarada saklanmaktadır.
Güvenlik	Entegre güvenlik yöntemleri vardır.	Uygulamalarla güvenlik mekanizmaları vardır.
Veri modeli	Tablosal olarak veri saklanmaktadır.	Hiyerarşik olarak veri saklamaktadır.
Veri tipleri	İkili verileri de saklama olanağı veren geniş kapsamlı veri tipi vardır.	XSD şemalarında tanımlanan veri tipleri vardır.
Veriler arasındaki ilişki	Tablolarda bulunan sütunlar başka tablolarla DDL ile ilişkilendirilebilmektedir.	XSD dokümanı ile sağlanabilmektedir.
Şema	Esnektir.	Sıkı kurallara sahiptir.
Referansların entegrasyonu	Verilerin diğer tablolarla entegrasyonu desteklenmektedir.	Verilerin entegrasyonu için zorunluluk yoktur.

Çizelge 5.1. (Devam) İlişkisel veritabanı ve XML dosyaları arasındaki farklar

Özellik	**İlişkisel Veritabanı**	**XML Dosyası**
Ek doğrulama metotları	Tetikleyiciler ile ek doğrulama yapılabilmektedir.	XSD şemalarında doğrulama yöntemleri tanımlanabilmektedir.
Sorgulama	İlişkisel veritabanları, endüstriyel standart olan SQL dilini desteklemektedir. Veritabanları ayrıca ek özellikler tanımlayabilmektedir.	Bir XML dosyası DOM, SAX veya XPath [57] dili ile sorgulanabilmektedir. Birden fazla XML dosyasının sorgulanması için XQuery dili kullanılmaktadır.
Sorgu Sonuçları Üzerinde Yapılabilecek İşlemler	SQL dili, sorgu sonuçları üzerinde gruplama, sıralama ve ek sorgular tanımlanabilmesini sağlamaktadır.	XSLT (Extensible Style Sheet Language Transformations) ve XQuery ile sorgu sonuçları üzerinde sıralama, gruplama ve daha karmaşık işlemler yapılabilmesini sağlamaktadır.
Birden fazla kaynaktan sorgulama	Birden fazla veritabanının aynı anda sorgulanması desteklenmektedir.	XSLT ve XQuery birden fazla dosyanın sorgulanabilmesini sağlamaktadır.
İndeksleme	Gelişmiş indeksleme yöntemlerine sahiptir.	Uygulamalarla sağlanmaktadır.
İş tamamlama	Veritabanları işlemi tamamlama ve yapılan işlemleri geri alma özelliğine sahiptir. Atomik işlem yapılabilmesine destek olmaktadır.	Şu anda bu işlemler XML teknolojilerinde yapılamamaktadır.

Çizelge 5.1. (Devam) İlişkisel veritabanı ve XML dosyaları arasındaki farklar

Özellik	**İlişkisel Veritabanı**	**XML Dosyası**
Çok kullanıcılık	Birden fazla kullanıcının aynı anda verilere erişebilmesini ve başkası tarafından kullanılan verilerin kilitlenebilmesini desteklemektedir.	XML dokümanlarının işlenmesinde bu özellik bulunmamaktadır.
Platform bağımsızlık	Veritabanları platform bağımlıdır, fakat diğer platformlara taşınabilmesini sağlamaktadırlar.	Platform bağımsızdır.
Şemalara bağımlılık	Veritabanlarında şemaların (tablo yapılarının) tanımlanması zorunludur.	XML dokümanlarında şema tanımlanması zorunlu değildir.
Şemaların tekrar kullanımı	Tanımlanan şemalar veritabanının örneğine bağımlıdır. Taşınması zordur.	Aynı şema birden fazla XML dokümanı tarafından kullanılabilir.
Alt elemana sahip olma	Tablolarda bulunan sütunların alt sütunları tanımlanamaz.	XML dokümanlarında bulunan elemanlara alt öğe tanımlanabilir.

XML verileri fiziksel ortam olarak düz dosyalarda saklanmaktadır. Dosyaların insanlar tarafından okunabilmesi kolay olduğu için dosyaların yönetimi bu açıdan kolaydır. Fakat çok fazla ve büyük verilerin yönetiminde bu dosyaların yönetilmesi zorlaşmaktadır. İlişkisel veritabanlarında veriler kendi içinde saklanmakta ve yönetilmektedir. Bu sayede verilerin yönetimi konusunda kullanıcıların yapacakları işlemler azalmaktadır. Fakat verilerin nasıl saklandığı ve yönetildiği kullanıcılar tarafından bilinmemekte ve verilerin ulaşılabilirliği konusunda veritabanına bağımlılık ortaya çıkmaktadır.

İlişkisel veritabanları, kullanıcı yönetimini, kullanıcıların yetkilerini ve verilerinin güvenliğini sağlama konusunda çok gelişmişlerdir. Örneğin kullanıcıların veritabanına bağlanmasına izin verilmeyebilir, tablolara erişim engellenebilir hatta tablolara erişim sağlansa bile bazı verilerin kullanıcıya gösterilmesi engellenebilir. XML dosyasında bulunan veriler için bu şekilde bir güvenlik mekanizması ancak XML dosyalarına doğrudan erişim imkanı vermeyen yazılımların geliştirilmesi gerekmektedir. Geliştirilen bu yazılımlar ile kullanıcı yetkileri tanımlanabilir ve yetkiye göre dosyalara erişim sağlanabilir.

İlişkisel veritabanları verilerin tablosal olarak saklanması temeline dayanmaktadır. Bu veritabanlarında, verilerin saklanması için verilerin tablosal olarak ifade edilmesi zorunluluğu ortaya çıkmaktadır. Hiyerarşik veriler, tablosal yapılara dönüştürülerek ilişkisel veritabanlarında saklanabilirken, tablosal verilerin XML dokümanlarında saklanması için hiyerarşik yapıya dönüştürülmesi gerekmektedir. Burada en önemli nokta tüm veriler hiyerarşik yapıya kolayca dönüştürülebilirken, her hiyerarşik veriyi tablosal yapıya dönüştürmenin mümkün olmaması veya dönüştürme işleminin çok zor olmasıdır. Tablosal yapıda olan veriler, kök elemandan sonra bir seviyeli elemanlara çevrilerek hiyerarşik yapıya kolayca dönüştürülür. Hiyerarşik verilerin tablosal yapıya dönüştürülmesi için ise her bir elemanın bir sütuna ve her bir hiyerarşinin bir tabloya dönüştürülmesi gibi işlemlerin detaylı bir şekilde tasarlanması gerekmektedir.

İlişkisel veritabanları, sütunlarda saklanacak veriler için çok fazla veri tipi tanımlamışlardır. Bu veri tipleri basit veri tipleri ve ikili veri tiplerinden oluşmaktadır. Genel veri tipleri SQL dilinde tanımlanmış olmasına rağmen bazı veritabanları kendi veri tiplerini de tanımlayabilmektedirler. XML dosyalarında kullanılacak veri tipleri XSD dokümanlarında tanımlanan veri tiplerini kullanmak zorundadırlar. Bu veri tipleri basit veri tipleri olabileceği gibi hiyerarşik bilgiye sahip veri tipleri ve kişiselleştirilmiş veri tipleri de olabilir.

İlişkisel veritabanlarında bulunan sütunlar başka bir tablodaki bir veri ile ilişkilendirilerek, verilerin birbirleri ile bağlantısı tanımlanabilir. XML dokümanlarında ilişkiler, XSD dokümanlarında doğrudan tanımlanabileceği gibi

doküman içinde referanslar tanımlanarak ta belirtilebilir. İlişkisel veritabanlarında birbirleri ile ilişkisi olması gerektiği belirtilen alanların, gerçekten doğru bilgileri içerip içermediği veritabanı tarafından kontrol edilir. Kural ihlalleri olduğunda bu hata mesajı olarak kullanıcıya iletilir. Fakat XML dokümanlarında bir elemanın başka bir elemana referans vermesi durumunda bu referans bilgisinin doğruluğu otomatik olarak kontrol edilmemektedir. Ancak iki eleman arasındaki ilişki XSD dokümanında tanımlanmış ise bu ilişkilerin doğruluğu kontrol edilebilmektedir.

XSD dokümanlarının hazırlanması için belirli kurallara uyulması gerekmektedir. XML dokümanları, XSD dokümanında tanımlanan yapıları kullanacaksa bu dokümanda tanımlanan bütün kurallara uymak zorundadır. İlişkisel veritabanlarında bir tablo tanımlamak ve tablo üzerinde değişiklik yapabilmek için tanımlanan kurallar daha esnektir.

XML dokümanlarında verilerin doğruluğu için bir çok kural tanımlanabilmekte ve bu kurallara göre dokümanın oluşturulması sağlanmaktadır. Fakat bu kurallar sadece statik kurallardır. İlişkisel veritabanlarında, tetikleyiciler ile hem statik kural kontrolleri yapılabilmekte hem de bir olaydan sonra bu tetikleyiciler kendileri bir işlem gerçekleştirebilmektedirler. Örneğin XML dokümanlarında oluşturulan verinin oluşturulma tarihi dokümanda yer alacaksa bu tarihin o anki tarih olması gerektiği kuralı tanımlanamazken, ilişkisel veritabanlarında tetikleyiciler ile verinin veritabanına eklenmesi anında bu alana o anki tarih yazılabilmektedir.

İlişkisel veritabanlarının sorgulanması SQL dili ile gerçekleştirilmektedir. Bu dil ilişkisel veritabanları için standart hale gelmiştir. Bu dil ile yapısal değişiklikler yapılabildiği gibi verilerin yönetimi de yapılabilmektedir. Verilerin sorgulanması için kullanılan SQL dili ile veritabanında bulunan tablolar üzerinde istenilen sorgulamalar yapılabilmektedir. XML dokümanlarının sorgulanması için ise DOM, SAX ve XPath gibi XML dokümanını işleyen teknolojiler bulunduğu gibi SQL dilinden daha kabiliyette olan XQuery [13, 14] dili kullanılmaktadır. Bu dil W3C tarafından önerilen XPath [63, 64] dili üzerine kurulmuş ve yine W3C tarafından önerilen bir dildir. Bu dil, programlama dillerinde bulunan özelliklere de sahip olduğu için dil ile

döngüler, koşul durumları kolay şekilde tanımlanmakta ve hiyerarşi içinde sorgulamalar desteklenmektedir.

İlişkisel veritabanları ile sorgu sonuçları üzerinde sadece sıralama ve gruplama yapılabilirken, XML dokümanlarının sorgulanması sonucu dönen veriler üzerinde daha geniş işlemler yapılabilmektedir. Ayrıca, sorgu sonuçları XSLT [63] kullanılarak istenilen formata dönüştürülebilmektedir. Sorgu sonuçlarının görüntülenmesinde SQL diline göre XQuery ve XSLT çok güçlü ve gelişmiştir. SQL dili veritabanı üzerinde güncelleme yapılabilmesini de sağlamakta fakat XML dokümanlarının güncellenmesi için W3C tarafından önerilen XUpdate [64] kullanılmaktadır. Bu açıdan XML dokümanlarının güncellenmesi, ilişkisel veritabanlarında bulunan verileri güncellemekten daha zordur.

XML dokümanlarının hepsi üzerinde sorgu yapabilmek için XQuery ve XSLT, bir doküman üzerinde yapılacak sorgularda XPath kullanılabilmektedir. İlişkisel veritabanlarında SQL dili ile birden fazla kaynaktan sorgu yapılabilmektedir. XQuery ve SQL dilleri bu özellikleri bakımından benzerdir.

İlişkisel veritabanları verilere hızlı erişim konusunda gelişmiş indeksleme mekanizmalarına sahiptir. Bu indeksleme mekanizmaları veritabanı tarafından yönetilmekte ve çok büyük performans kazanımları sağlamaktadır. XML dokümanlarının sorgulanması için indeksleme mekanizmaları ancak yazılımlarla sağlanabilmektedir. İlişkisel veritabanlarında kullanılan indeksleme mekanizmaları, XML dokümanlarının indekslenmesinde aynen kullanılamamaktadır. İlişkisel veritabanlarında indeksleme, tablosal veriler üzerinde yapıldığı için daha kolaydır. Hiyerarşik verilerin indekslenmesi konusunda farklı yaklaşımların kullanılması gerekmektedir.

İş tamamlama konusunda ilişkisel veritabanları güçlü bir alt yapıya sahiptir. Atomik işlem gerçekleştirilmesi, bu atomik işlem sırasında bir hata oluşması durumunda yapılan işlemlerin geri alınması ve işlem sırasında hata oluşmadıysa değişikliklerin veritabanına yansıtılması, başarı ile gerçekleştirilebilmektedir. XML dokümanları üzerinde yapılan işlemlerde bu özelliğin doğrudan gerçekleştirilmesi mümkün

değildir. İnternet servislerinde bu özellikleri sağlamak amacı ile bazı standartlar tanımlanmaya çalışılmaktadır. İş tamamlama konusunda ilişkisel veritabanlarının güçlü olmasının en büyük sebebi verilerin kendi yönetiminde olması ve kullanıcılara doğrudan müdahale imkanı vermemesidir. XML dokümanları dosyalarda saklandığı için dosyaların yönetimi net olarak tanımlanamamaktadır. XML dokümanlarının bu özelliklerinden dolayı birden çok kullanıcının erişebileceği bir alt yapı oluşturulamamaktadır. İlişkisel veritabanları ise yönetimin kendilerinde olmasından dolayı verileri birden fazla kullanıcıya sunabilmekte ve aynı anda erişime açabilmektedir.

XML dokümanlarının dosyalarda saklanması ve yazı formatında olmasından dolayı hiçbir platforma bağımlılığı bulunmamaktadır. İstenilen işletim sistemine ve istenilen donanıma rahatlıkla taşınabilmektedir. İlişkisel veritabanları ise verileri kendileri yönettiği ve kullanıcıdan sakladıkları için kendilerine bağımlı hale getirmektedir. Aynı veritabanının diğer platformlarda çalışan sürümleri olsa bile bu veritabanına bağımlılık devam etmektedir. Veritabanı değiştirilmek istendiğinde, standart SQL dili özellikleri kullanılarak verilerin taşınması mümkündür. Fakat tasarlanan veritabanlarında verilerin doğrulanması ve bazı olaylarda bazı işlemlerin yapılması için tetikleyiciler tanımlandıysa, verilerin doğrudan taşınabilmesi olanağı ortadan kalkmaktadır.

İlişkisel veritabanlarında verilerin saklanabilmesi için önce tabloların ve tablolar arasındaki ilişkilerin tanımlanması gerekmektedir. Bu işlemlerin ardından veriler veritabanında saklanabilmektedir. XML dokümanlarında veri saklamak için şemalar kullanılmaktadır. Bu şemalar XML dokümanı içinde bulunan verilerin sağlıklı şekilde saklanmasını sağlamaktadır. Fakat şema tanımlamak, XML dokümanı oluşturmak için zorunlu değildir.

İlişkisel veritabanlarında tanımlanan şemalar (tablolar ve ilişkiler) sadece o veritabanında kullanılmaktadır. Başka veritabanları bu şemayı kullanarak verileri saklayamamaktadır. Bu işlem için şemanın tanımlandığı veritabanından bu şemanın bir kopyasının alınması gerekmektedir. XML dokümanlarında kullanılmak için

hazırlanan şemalarda bu şekilde bir sınırlama bulunmamaktadır. Tanımlanan bir şema, herhangi bir XML dokümanı içinde rahatlıkla kullanılabilmektedir.

Tablolarda bulunan sütunlar sadece bir bilgi saklamaktadır. XML dokümanlarında bir eleman bilgi ve kendi hiyerarşisi altında bulunan bütün elemanları taşımaktadır. Bu özelliği sayesinde XML dokümanının bir elemanının altına istenildiği kadar alt eleman tanımlanabilmekte, özellikler eklenebilmektedir. XML dokümanında bulunan bir elemanı, ilişkisel veritabanında bulunan bir sütun olarak düşünmek pek mümkün olamamaktadır. Örneğin XML dokümanındaki bir eleman bir değere ve beş özelliğe sahip olabilir. Bütün bu bilgilerin veritabanında saklanması için birden fazla sütun kullanılması gerekmektedir. Ayrıca XML dokümanında, bir elemana birden fazla alt eleman eklemek için XML dokümanına bir etiket eklenmesi yeterli olurken, ilişkisel veritabanında bu işlem için yapılması gereken analiz edilerek bir sütun, bir tablo veya bir ilişki kurulması seçeneklerinden birinin seçilmesi gerekmektedir.

Yukarıda ilişkisel veriler ile XML dokümanında saklanan verilerin nasıl yönetildiği konusu anlatılmıştır. Bütün anlatılan özellikler göz önüne alındığında ilişkisel veritabanlarının güçlü yanları ile XML dokümanlarının güçlü yanları birleştirilerek bir yöntem bulunması gerekliliği ortaya çıkmaktadır. XML dokümanlarının ilişkisel veritabanlarında saklanabilmesi bir seçenek olarak ortaya çıkarken, hiyerarşik verilerin ilişkisel veritabanları ile yönetiminin çok kolay olmadığı görülmektedir. Öncelikle XML dokümanlarının düz dosyalardan veritabanında yönetilen dosyalara geçilmesi gerekmektedir. Bu şekilde birden fazla XML dokümanı aynı anda yönetilebilir ve ilişkisel veritabanlarının güçlü yanlarına sahip çözüm ortaya çıkarılabilir.

5.1. İlişkisel Veritabanları

Hiyerarşik verilerin ilişkisel veritabanlarında saklanabilmesi için çeşitli alternatifler bulunmaktadır. Hiyerarşik verilerin tablosal yapılara dönüştürülmesi öncelikli problemdir. Bu problemin aşılması için ilişkisel veritabanı oluşturmak için tanımlanan yöntemlerin göz önünde bulundurulması gerekmektedir. İlişkisel veritabanı tasarımı gerçekleştirilirken dikkat edilecek hususlar ve uygulanacak

basamamaklar genel olarak normalizasyon olarak adlandırılır [65-67]. Normalizasyon iki önemli konuya odaklanmıştır. Öncelikle veritabanı tablolarında bulunan tekrarlı grupların (sütunların) ortadan kaldırılmasıdır. Veritabanına yeni bir kayıt eklemek istendiğinde belirli bir grup verinin tekrarlanması gerektiğinde bu durum problemlere sebep olmaktadır. Normalizasyonun diğer bir amacı ise tekrarlayan herhangi bir alanın tekrarını ortadan kaldırmaktadır. Grup tekrarlarının ve belirli bir alanın tekrarının ortadan kaldırılması için yapılan normalizasyon işlemi, normal form teorilerini kullanmaktadır.

Bir tablonun normalizasyon yapıldığını söyleyebilmek için bazı kriterleri sağlaması gerekmektedir. Normalizasyonun hedefi bir grup veriyi tekrarlı veri içermeyen ve düzgün şekilde güncellenebilen tablolara bölmektir. Bu normalizasyona üçüncü normal form denir. Veritabanında bulunan verilerin üçüncü normal formda olması kullanılabilir ve kolay yönetilebilir olması için yeterlidir. Üçüncü normal formun yanında birinci, ikinci, dördüncü ve beşinci normal formlar da bulunmaktadır fakat bu normal formlar ya tekrarlı verilerin olmasının önüne geçememekte ya da verileri çok fazla bölerek kolay kullanılabilirliğini ortadan kaldırmaktadır. Aşağıda her birinci, ikinci ve üçüncü normal formlar anlatılacaktır. Bu normal formlar hiyerarşik verilerin tablosal olarak ifade edilebilmesi için kullanılabilecek formlardır.

5.1.1. Normal Formlar

Birinci normal form

Birinci normal formda tanımlanan bir tablonun içinde bulunan bütün veriler, atomik olmalıdır. Yani bir alan başka bir alanda tekrarlanmamalıdır [65, 67].

Birinci normal formda tablo içinde bulunan kayıtlarda tekrarlamalar bulunabilir fakat bir kaydın alanlarının atomik olma özelliği sağlanmalıdır. Bu özelliklere sahip tablo birinci normal formda tablo olarak isimlendirilir. Şekil 5.1'de birinci normal formda bulunan bir tablo örneği verilmiştir. Bu tablonun alanları tekrarlı olmadığı için bu kurala uygundur.

ID1	üniversite	bölüm	kontenjan	adres	bölüm kütüphanesi
1	Gazi Üniversitesi	Bilgisayar Mühendisliği	30	Ankara	BM101
2	Gazi Üniversitesi	Elektronik Mühendisliği	40	Ankara	BM101
3	Ankara Üniversitesi	Bilgisayar Mühendisliği	30	Ankara	BM401
4	Pamukkale Üniversitesi	Elektronik Mühendisliği	50	Denizli	EM101
5	Ondokuz Mayıs Ünversitesi	Endüstri Mühendisliği	20	Samsun	EM201
6	Gazi Üniversitesi	Makina Mühendisliği	50	Ankara	BM101

Şekil 5.1. Birinci normal formda olan bir tablo

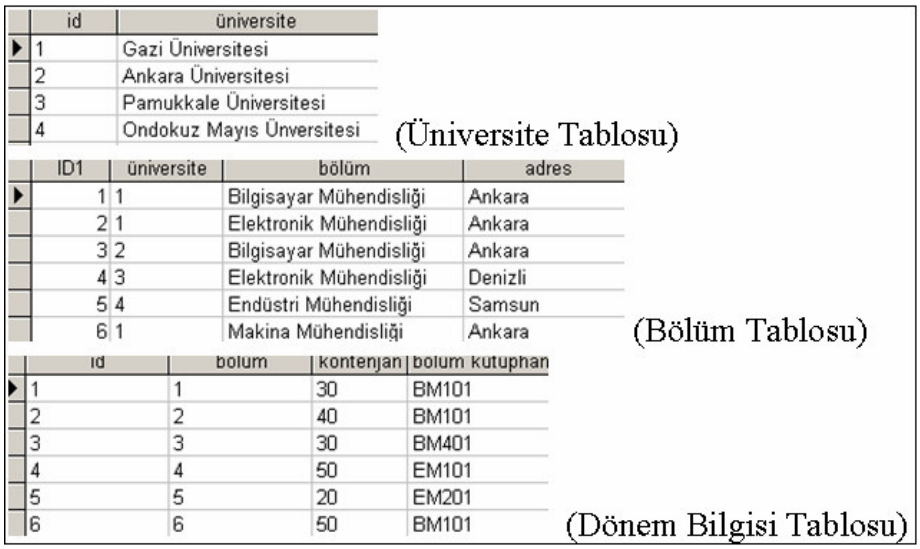
Birinci normal formda bulunan tablolarda bazı problemler görülmektedir. Bir kaydın silinmesi durumunda sadece o kayda ait bilginin yanında bazı kritik bilgiler de kaybedilmektedir. Örneğin *Pamukkale Üniversitesi*'nin kaydı silindiğinde *Pamukkale Üniversitesi'*nin hangi şehirde olduğu bilgisi de kaybedilir. Çünkü bu ilişki sadece bu kayıtta saklanmakta ve silindiğinde her iki bilgi birden kaybolmaktadır.

Ayrıca güncelleme işlemlerinde de bir kaydın güncellenmesinden sonra ek işlemlerin yapılması gerekebilmektedir. Örneğin *Gazi Üniversitesi Mühendislik Fakültesi*'nin adresi değiştirildiğinde bütün ilgili kayıtların ayrı ayrı güncellenmesi gerekmektedir.

Bunlara ek olarak, veri girişi yapılmak istendiğinde ve alanların girilmesi zorunluluğu bulunduğunda kayıt eklemek problemli olabilmektedir. Örnek tabloya bir kayıt girişi yapmak için *bölüm kütüphanesi* alanına giriş yapabilmek için fiziksel bir kütüphanenin ve kütüphanenin numarasının olması gerekmektedir.

İkinci normal form

İkinci normal formda, tablolar birinci normal formun şartlarını sağladıktan sonra her bir kaydın bir birincil anahtarı olmalı ve tabloda bulunan birincil anahtar olmayan bütün alanlar bu birincil anahtara bağlı olmalıdır [65, 67]. İkinci normal form ile bütün sütunların birincil anahtarla ilişkilendirilmiş olması problemlerin çözümü için yeterli değildir. Örnek tabloda (Bkz. Şekil 5.1) *üniversite*, *bölüm* ve *adres* ayrı bir tabloya taşınarak (*Bölüm Tablosu*) birincil anahtar oluşturulabilir. Başka bir tabloda da bölüm için tanımlanan birincil anahtar, kontenjan ve bölüm kütüphanesi (*Dönem Bilgisi Tablosu*) bulunabilir. Bu işlemden sonra üniversite isimleri de ayrı bir tabloda (*Üniversite Tablosu*) tutulup, bölümlerin tanımlandığı tabloda (*Bölüm Tablosu)* referansı tanımlandıktan sonra ikinci normal forma getirilmiş olur (Şekil 5.2).



id	üniversite
1	Gazi Üniversitesi
2	Ankara Üniversitesi
3	Pamukkale Üniversitesi
4	Ondokuz Mayıs Ünversitesi

(Üniversite Tablosu)

ID1	üniversite	bölüm	adres
1	1	Bilgisayar Mühendisliği	Ankara
2	1	Elektronik Mühendisliği	Ankara
3	2	Bilgisayar Mühendisliği	Ankara
4	3	Elektronik Mühendisliği	Denizli
5	4	Endüstri Mühendisliği	Samsun
6	1	Makina Mühendisliği	Ankara

(Bölüm Tablosu)

id	bolum	kontenjan	bolum kutuphan
1	1	30	BM101
2	2	40	BM101
3	3	30	BM401
4	4	50	EM101
5	5	20	EM201
6	6	50	BM101

(Dönem Bilgisi Tablosu)

Şekil 5.2. İkinci normal forma uygun hazırlanan tablolar

İkinci normal formda da bazı problemler görülebilir. Örneğin kontenjan bilgisinin silinmesi ile kütüphane bilgisi de silinmiş olacaktır. Diğer bir problemde bir bölüm eklemek için üniversite bilgisinin daha önceden girilmiş olması gerekmektedir. Bu durum problem gibi gözükse de bazı durumlarda bu özellik avantajlar sağlamaktadır. Örneğin üniversite bilgilerin kontrolü bir noktadan yapılabilir hale gelmektedir.

Üçüncü normal form

Üçüncü normal formda, bütün sütunların birincil anahtar ile ilişkili olması gerekmektedir [65, 67]. İkinci normal formda sütunlar birincil anahtara bağlı olmalarına rağmen birincil anahtarın ifade ettiği anlam farklılaşmaktadır. Üçüncü normal formda bulunan bir tablo ikinci normal formun bütün özelliklerine sahip olmalı ve tüm sütunlar anlamsal olarak birincil anahtara bağımlı olmalıdır.

Dönem Bilgisi tablosunda kütüphane bilgisi, birincil anahtarla doğrudan ilişkili değildir. Bunun için kütüphane adında bir tablo tanımlanıp bölümlerle bu tablo ilişkilendirilmelidir. Bir tablo daha tanımlama ihtiyacı doğacak ve üçüncü normal form özelliklerine sahip bir tablo yapısı oluşturulmuş olacaktır.

Üçüncü normal form ile birinci ve ikinci normal formlarda gözlenen problemler giderilmiş, bilgi tekrarı ortadan kaldırılmış olur. Veritabanına bir kayıt eklemek ve

verilerin bütünlüğü sağlamak için gerekli bilgilerin daha önceden girilmiş olması gerekmektedir. Örneğin bir bölüm eklemek için üniversite bilgisi daha önce eklenmiş olması gerekmektedir. Bir bölümün adını değiştirmek için bir kaydın değiştirilmesi mümkün olmaktadır. Ek olarak veritabanında bir üniversitenin referansları var ise bu verinin silinmesi engellenmekte ve veri bütünlüğü sağlanmaktadır.

Hiyerarşik verilerin tablosal yapıya dönüştürülebilmesi için birinci, ikinci ve üçüncü normal formların kullanılması bir seçenek olarak görülmektedir. Dördüncü ve beşinci normal formların hiyerarşik verilerin modellenmesi kullanılabilir değildir. Çünkü bu normal formlar ile üçüncü normal form için oluşturulacak olandan daha fazla tablo oluşacaktır.

Hiyerarşik verilerin tablosal yapıya getirilmesinde, öncelikle anlatılan normalizasyon yöntemleri üzerinden gidilecek ve EWIRDB veritabanı üzerinde kullanılabilirliği incelenecektir.

5.2. Hiyerarşik Verilerin İlişkisel Veritabanı İle Modellenmesinin Alternatifleri

Bu bölümde hiyerarşik verilerin ilişkisel veritabanında saklanabilmesi için uygulanabilecek beş alternatif anlatılmıştır.

5.2.1. Hiyerarşik verilerin birinci normal form ile modellenmesi

Hiyerarşik verilerin birinci normal form ile modellenmesi verilerin hiyerarşi seviyesine ve sahip olunan eleman sayısına bağlıdır. Bir veya iki seviye hiyerarşisi bulunan ve az sayıda elemanı bulunan hiyerarşik veriler tablosal formata çok kolay şekilde dönüştürülebilir. Bu işlem için tüm elemanların ve bu elemanların özelliklerinin veritabanında bir tabloda sütun olarak tanımlanması yeterlidir.

Az elemanı bulunan hiyerarşik veriler tablosal formata dönüştürüldükten sonra tüm sütunların veri tipleri belirlenmeli ve veri tiplerinde kısıtlamalar tanımlandıysa bu kısıtları gerçekleştirmek için tetikleyiciler tanımlanmalıdır.

Burada ortaya çıkan en büyük problem hiyerarşilerin nasıl yönetileceğidir. Bir seçenek olarak elemanların hiyerarşi bilgileri de tablolarda saklanabilir. Bu şekilde

birinci normal formun kriterleri hala korunmuş olur. Fakat hiyerarşilerin yönetimi tetikleyiciler veya dış yazılımlar tarafından yapılması gerekir [21].

Birinci normal formda bulunan tablonun her satırına bir hiyerarşik veri saklanır. Hiyerarşik verinin tablosal forma dönüştürülmesi ve tablosal formda bulunan verinin hiyerarşik veriye dönüştürülmesi ayrı bir işlem olarak durmakta ve bu işlem dış yazılımlar tarafından sağlanmaktadır.

Çok fazla hiyerarşisi ve elemanı bulunan hiyerarşik verilerin ilişkisel veritabanında saklanabilmesi için ilişkisel veritabanının kısıtları dikkate alınmalıdır. Örneğin birçok veritabanı, bir tabloda 255'ten fazla sütunun tanımlanmasına izin vermemektedir. Hiyerarşisi çok seviyeli ve elemanı fazla veriler için doğrudan bir tabloya dönüştürmek mümkün olmayacaktır. Bunun için ikinci normal form kurallarına uygun bir veritabanının tasarlanması gerekmektedir.

EWIRDB veritabanının birinci normal form ile modellenebilmesi imkansızdır [21]. EWIRDB veritabanında 2400'ün üzerinde parametre bulunmakta ve bu parametrelerin bir tabloya sığdırılabilme imkanı bulunmamaktadır.

5.2.2. Hiyerarşik verilerin ikinci normal form ile modellenmesi

Az elemanı bulunan hiyerarşik veriler ikinci normal form kurallarına uygundur. Fazla elemanı olan hiyerarşik veriler için bu normal formda olmak zorundadır. Hiyerarşisi çok seviyeli ve elemanı fazla olan veriler, ortak birincil anahtarlar kullanarak birden fazla tabloya dağıtılabilirler.

Bu modellemede, her bir hiyerarşik veri için bir birincil anahtar tanımlanır. Veri veritabanının kısıtlarına ve mantıksal gereksinimlere uygun olarak birden fazla parçaya bölünebilir. Bu işlemden sonra hiyerarşik veri, her tabloda bir kaydı bulunacak şekilde tablolara dağıtılır.

Burada en önemli nokta, hiyerarşik verilerin ilişkisel veritabanında saklanırken artık tek parça olarak değil dağınık olarak yerleştirilmektedir. Verinin tüm elemanlarına ulaşmak için bir çok tablonun sorgulanması gerekmektedir. Sorgulama sonucunda, bütün elemanlar birleştirilerek bir hiyerarşik yapıya dönüştürülmesi gerekmektedir.

EWIRDB veritabanının ikinci normal form ile modellenebilmesi mümkündür. EWIRDB veritabanında 2500 parametrenin bulunduğu ve bir tabloya en fazla 255 sütun eklenebildiği varsayılırsa yaklaşık on tablonun tasarlanması gerekecektir. Burada verilerin hiyerarşik bilgilerinin saklanmadığı düşünülmüştür. Elemanların hiyerarşi bilgisinin de tablolarda saklanacağı varsayıldığında en az yirmi tabloya ihtiyaç duyulacaktır. EWIRDB veritabanında bulunan her bir alanın birden fazla değeri bulunabilmektedir. Bunu desteklemek için ikinci normal form yetersiz kalmaktadır [21]. Çünkü bir alana elle bir veri girmek için diğer bütün alanların tekrar edilmesi gerekmektedir. İkinci normal form ancak hiç bir elemanı tekrarlamayan yapıda hiyerarşisi olan veriler için kullanılabilir.

5.2.3. Hiyerarşik verilerin üçüncü normal form ile modellenmesi

Az elemanı olan hiyerarşik veriler birinci ve ikinci normal form ile modellenebildiği gibi, üçüncü normal form ile de modellenebilir [21, 68]. Üçüncü normal formda, tüm alanların birincil anahtar ile ilişkili olması gerekmektedir. Bu normal form tekrarlanan verilerin yönetimi için uygundur. Eğer bir hiyerarşik verinin tekrarlayabilir elemanları var ise üçüncü normal form ile modellenmesi gerekmektedir.

Üçüncü normal formun tekrarlanan alanların yok edilmesine yönelik olarak geliştirildiği düşünülürse, hiyerarşik verilerin analiz edilmesi ve birlikte değişen elemanların bir tabloda yer alması gerekmektedir. Hiyerarşik verinin analiz edilebilmesi için de, sahip olunan verinin elemanlarının hangi amaçla ve nasıl birbirini etkilediği bilinmesi gerekmektedir. Yani ilgilenilen veri hangi alanda ise o alanda uzmanlığı gerektirmektedir.

Verilerin üçüncü normal formla modellenebilmesi az elemanı bulunan hiyerarşik veriler için kolaydır. Fakat çok fazla elemanı olan hiyerarşik verilerin parçalanması, başka problemlerle karşı karşıya kalınmasına sebep olur. Parçalama işlemi sonucunda küçük fakat çok fazla tablonun ortaya çıkması kaçınılmazdır. Örneğin, EWIRDB veritabanının üçüncü normal form ile modellenebilmesi için tüm tekrarların giderilmesi, tüm tekrarların giderilmesi için de her bir alan ve alanın

hiyerarşi bilgisini saklayan bir tablo oluşturulması gerekmektedir. Bu durumda EWIRDB veritabanını üçüncü normal formla modelleyebilmek için 2400'ün üzerinde tablo oluşturmak gerekecektir.

Çok sayıda küçük tablolarda hiyerarşik verinin saklanması verinin artık okunamaz hale gelmesine sebep olmaktadır. Verinin ilişkisel veritabanında saklanabilmesi için de bu zorunludur. Bir hiyerarşik verinin veritabanından sorgulanabilmesi için çok fazla tabloya ulaşılması gerekecektir. EWIRDB veritabanının 2400'ün üzerinde tablo ile modellenmesi hem az kullanışlı hem de performans açısından sorunlu hale getirecektir [21]. Üçüncü normal form ile 2400'ün üzerinde tablo tasarımı yapılması gerekirken dördüncü ve beşinci normal formlarla daha fazla tablo oluşturulacağı için bu modeller de kullanışsız olmaktadır.

5.2.4. Hiyerarşik verilerin ikili dosya olarak modellenmesi

XML dokümanları tek parça olarak bir tabloda ikili dosya şeklinde saklanıp yönetilebilir. Bu yöntem XML dokümanlarının saklanmasından sonra dosyaların yine ikili dosya olarak sorgulanabilmesini sağlamaktadır. Dosyaların içeriklerini sorgulamak için yine ek uygulamalara ihtiyaç duymaktadır.

Bilginin yönetilmesi öncelikli olduğu durumlarda, ikili dosya olarak saklanan veritabanının kullanımı imkansızdır. Bir verinin bir elemanının değerini alabilmek için tüm dosya, veritabanından alınıp, işlenip ve sonuçlar dönecektir. Bu durumda, basit bir tasarımla oluşturulan veritabanının kullanımı zorlaşacaktır.

5.2.5. Hiyerarşik verilerin dinamik olarak modellenmesi

İlişkisel veritabanları ile büyük hiyerarşik veriler dinamik olarak yönetilebilir. Bu yöntemle, hiyerarşilerin desteklenebilmesi ve çok fazla tablo kullanmadan tasarlanabilir olmaktadır. İlişkisel veritabanı içinde hiyerarşilerin tanımlandığı bir tablo ve ilişkilere uygun olarak veri girişi yapılabilecek bir tablo tanımlanmalıdır [21]. Bu şekilde tanımlanan ilişkisel veritabanlarına dinamik ilişkisel veritabanı denir.

Dinamik ilişkisel veritabanında, öncelikle verilerin mümkün olan en az sayıda tabloda saklanması ve saklanan verilerin hiyerarşisinin ve doğruluğunun yönetilmesi için bir yapı tablosu tanımlanır. Yapı tablosunda her bir hiyerarşik eleman için bir kayıt bulunur. Bu elemanın veri tipi, adı, etiketi, hiyerarşideki yeri bululur. Bu tablo XML dokümanları için tanımlanan XSD dokümanının görevini üstlenir.

Yapı tablosu kullanılarak bir verinin her elemanı için veri girişi yapılabilecek bir tablo hazırlanır. Bu tabloya veri girişi metin formatında tanımlanır. Bu modelde en büyük problem, tüm verilerin metin formatında saklanmasıdır. XSD dokümanında çok fazla veri tipi tanımlanıp bunların doğruluğu kontrol edilebilirken bu modelde bu ancak yazılım ile kontrol edilebilmektedir.

Bu modeldeki diğer bir problem de verilerin dağınık olmasıdır. Üçüncü normal formda veriler tablolara yayılmasına rağmen, burada kayıtlar halinde bulunmaktadır. Bir veriye erişebilmek için çok fazla kaydın sorgulanması ve bir araya getirilmesi gerekmektedir. Dinamik ilişkisel veritabanı modeli ile genişletilebilir özelliklere sahip veritabanı modeli tanımlanabilir, fakat dinamik modelin hiyerarşisini kontrol etme konusunda yetersizdir. Hiyerarşik verilerin ilişkisel veritabanına çevrilmesinde karşılaşılan bütün bu güçlüklere rağmen, hiyerarşik verilerin daha düzgün şekilde yönetilmesi konusunda çalışmalar devam etmektedir. İlişkisel veritabanlarını bu kadar cazip yapan özellikler; veri yönetimi, güvenlik, iş tamamlama, indeksleme kabiliyetleridir.

İlişkisel veritabanlarına alternatif olarak hiyerarşik verilerin yönetimi için nesne yönelimli veritabanları veya XML veritabanları kullanılabilir. Çünkü bu veritabanları doğası gereği verilerin hiyerarşik olarak saklanabilmesine yardımcı olmaktadırlar.

5.3. Hiyerarşik Verilerin Nesne Yönelimli Veritabanı ile Modellenmesi Alternatifi

Bu bölümde, nesne-yönelimli veritabanları ile hiyerarşik verilerin modellenmesi alternatifi anlatılacaktır.

Nesne-yönelimli veritabanlarının geçmişi, ilişkisel veritabanlarının geçmişi ile paralellik göstermektedir. 1985'li yıllarda bu konuda çalışmalar hızlanmış fakat ilişkisel veritabanı modelinin gölgesinde kalmıştır.

Nesne yönelimli veritabanları, nesne yönelimli programlama ile veritabanı yönetim sistemlerinin birleştirilmesi sonucu ortaya çıkan modellerdir. Nesne yönelimli programlama dillerinde kullanılan nesneler, veritabanlarında saklanarak ilişkisel veritabanına benzer şekilde yönetilmesi sağlanır.

Nesne yönelimli veritabanları, ilişkisel veritabanlarının tüm özelliklerine sahip olabilseler de bazı temel problemleri vardır. Örneğin, ilişkisel veritabanları kadar matematiksel temellere dayanmamakta ve standartlaşma konusunda çalışmalar olsa bile tam bir standartlaşma bulunmamaktadır. Nesne yönelimli veritabanlarında sorgulama için OQL (Object Query Language) kullanılmaktadır. Bu dil SQL diline benzer özelliklere sahiptir. Nesne yönelimli veritabanlarında indeksleme ve sorgulama yöntemleri ilişkisel veritabanlarından farklı olabilmektedir. Sorgulama için ilişkisel veritabanlarında olduğu gibi verilerin birleştirilmesi gerekmemekte, nesnelere olan referanslar kullanılarak verilere ulaşılmaktadır. Nesne yönelimli veritabanlarında hiyerarşik verilerin modellenmesi için hiyerarşik veride bulunan her bir elemanın nesne olarak oluşturulması gerekmektedir. Oluşturulan bu nesneler veritabanında saklanarak yönetilmektedir. Veritabanında saklanan veriler nesneler olarak saklandığı için verilere ulaşabilmek için bu veritabanına bağımlılık ortaya çıkmaktadır. Ayrıca bu konuda bir standartlaşma olmadığı için bağımlılık kalıcı olabilmektedir. Veritabanında saklanan verilerin ikili yapıda nesnelerde saklanmasından dolayı verilerin transfer edilmesi için de bu veritabanına bağımlılık bulunmaktadır.

Hiyerarşik verilerin nesnelere dönüştürülmesi, az elemanı olan veriler için çok kolay olabilirken, çok fazla elemana sahip hiyerarşik verilerin nesne yönelimli olarak modellenmesinin ardından, çok fazla nesne yapısı ortaya çıkacaktır. Örneğin, EWIRDB veritabanının nesne yönelimli olarak modellenmesi sonucu bu veritabanı yapısında tanımlanan parametre kadar nesnenin oluşturulması gerekecektir. Nesne yönelimli bir veritabanı ile hiyerarşik verileri ilişkisel veritabanlarına göre daha

kolay modellenebilmektedir [5, 21]. Bu şekilde modellenen veriler üzerinde, ilişkisel veritabanı ile yapılabilecek bütün sorgulamalar yapılabilmektedir. Fakat herhangi bir hiyerarşide bulunan veriler üzerinde karışık sorgulamalar yapılması bu modelle de mümkün olmamaktadır. Örneğin; herhangi bir hiyerarşisinde 5 değerine sahip olan verilerin sorgulanması nesne yönelimli veritabanlarında mümkün olmamaktadır.

Nesne yönelimli veritabanı olarak modellenen EWIRDB veritabanı yapısını yönetmek için çok fazla kullanıcı arayüzleri hazırlanacak ve çok fazla zamana ihtiyaç duyulacaktır [15, 16, 21]. Ayrıca nesne yönelimli veritabanı olarak modellenmesi durumunda veritabanında yeni bir alan ekleme özelliği bulunmayacaktır. Veritabanında yeni bir alan eklemek için tekrar tasarım yapılması, nesnelerin güncellenmesi ve eski verilerin yeni yapıya taşınması gerekecektir.

Hiyerarşik veriler için nesne yönelimli veritabanlarının ilişkisel veritabanlarına göre çok fazla avantajı olmasına rağmen dinamik alan eklenememesi hiyerarşik verilerin genişletilebilirliğine engel olmaktadır [21]. Genişleyebilir EWIRDB yapısının ilişkisel ve nesne yönelimli veritabanları ile modellenmesi bir çok problemi ortaya çıkarmaktadır.

5.4. Hiyerarşik Verilerin XML Veritabanı ile Modellenmesinin Alternatifleri

Bu bölümde hiyerarşik verilerin XML veritabanları kullanılarak modellenmesi anlatılacaktır.

XML veritabanı hiyerarşik verilerin modellenmesinde karşılaşılan problemlerin çözümü için ortaya atılmış, XML formatındaki hiyerarşik verilerin saklanması, sorgulanması ve yönetilmesi için gerekli altyapıyı oluşturmuştur [15, 16, 21].

XML dokümanlarının veritabanı olarak yönetilmesi için iki seçenek bulunmaktadır [15, 16, 21, 68]. Bu seçeneklerden bir tanesi, XML dokümanlarının girdi olarak alınıp işleyip kendi formatına dönüştüren ve dokümanlar sorgulandığında kendi formatından tekrar XML dokümanı formatına dönüştüren veritabanlarıdır (XML formatını destekleyen veritabanları). Diğer bir seçenek ise XML dokümanlarını orijinal formatında saklayarak bu dokümanlar üzerinde sorgulama yapılabilmesini

sağlayan veritabanlarıdır (doğal XML veritabanları). Hiyerarşik verilerin modellenmesi XML veritabanı seçeneklerine göre iki alternatifle yapılabilir. Bu alternatifler aşağıdaki alt bölümlerde detaylandırılmıştır.

5.4.1. Hiyerarşik verilerin XML formatını destekleyen veritabanları ile modellenmesi

Bu veritabanları, XML dokümanlarında bulunan verileri ilişkisel veya nesne yönelimli elemanlara dönüştürerek veritabanında saklarlar [69]. Bu verilerin veritabanında saklanma şekli kullanıcı tarafından kontrol edilemez. Verileri ilişkisel olarak saklayan veritabanları, XML dokümanları üzerinde hem SQL dili ile hem de XQuery dili ile sorgulama imkanı vermektedirler.

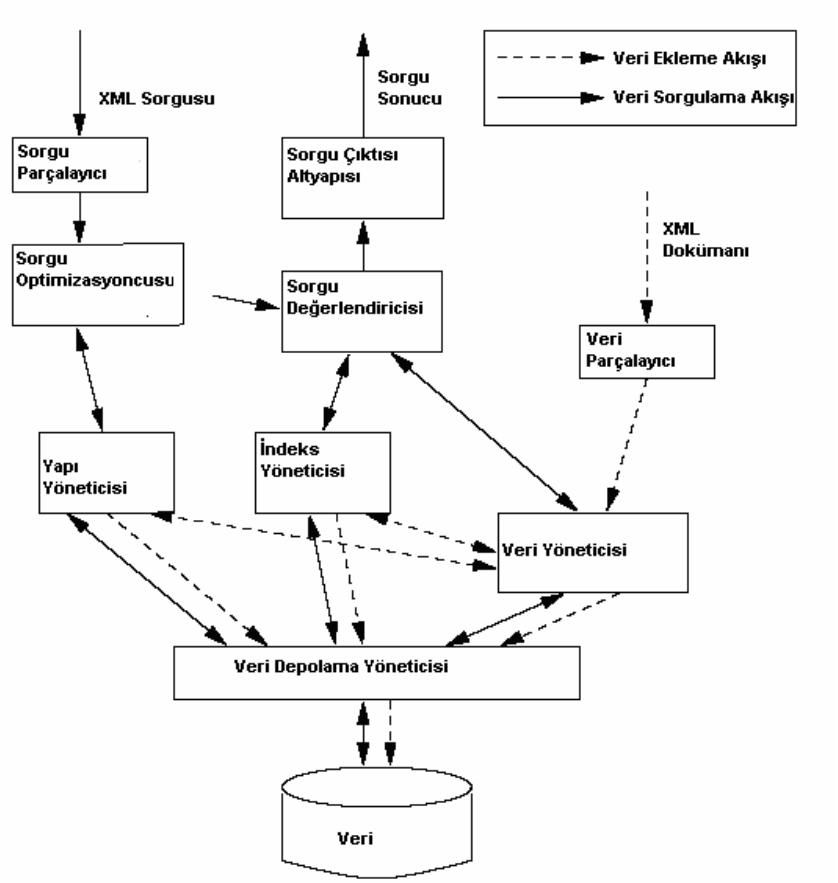
Nesne yönelimli veya ilişkisel veritabanları XML dokümanının saklanması için ek özellikler tanımlarlar. Bu veritabanlarının mimarileri XML dokümanlarının yapılarına uygun olmadığı için kendi iç mimarilerine uygun olacak şekilde XML dokümanlarını yönetmektedirler.

Bu şekilde hiyerarşik verilerin modellenmesi için veritabanlarının altyapı kısıtları dikkate alınması gerekmektedir. Örneğin, hiyerarşik veriler ilişkisel tablolarda saklanıyorsa, hiyerarşik verilerin ilişkisel formata dönüştürülmesinde karşılaşılan problemler yine ortaya çıkacaktır.

5.4.2. Hiyerarşik verilerin doğal XML veritabanları ile modellenmesi

Hiyerarşik verilerin doğrudan desteklenmesini sağlayacak bir model de doğal XML veritabanlarıdır [12, 64, 70]. Bu model mantıksal olarak hiyerarşik verilerin yönetilmesine, XML dokümanlarının elemanlarını bir birim olarak kabul edilmesine ve veritabanı fiziksel saklanma ortamı için bir modele dayandırmadan her türlü fiziksel ortamı destekleyebilecek bir ortam oluşturmaya odaklanmıştır. İlişkisel veya nesne yönelimli veritabanı modellerin kısıtlarına bağlı olmayan bir veritabanı oluşturarak hiyerarşik verilerin desteklenmesi sağlanmıştır.

Doğal XML veritabanlarının genel mimari örneği Şekil 5.3'te verilmiştir [70].



Şekil 5.3. Doğal XML veritabanı mimarisi

XML dokümanı veritabanına eklenmek istendiğinde, doküman işlenerek (*Veri Parçalayıcı*) ağaç yapısı çıkarılmakta ve oluşturulan ağaç veri yöneticisine gönderilmektedir. *Veri Yöneticisi* bu verileri kullanarak veritabanında saklanacak formata çevirmektedir. Bu format, ağacın elemanlarının bir birim olarak ifade edilmesini sağlayan küçük birimlerden oluşmaktadır. Oluşturulan bu birimler *Veri Depolama Yöneticisi'*ne gönderilmektedir. Aynı zamanda *Veri Yöneticisi*, ağaç halinde bulunan verinin ileride sorgulanabilmesi ve sorguların hızlı gerçekleştirebilmesi için gerekli bilgileri *Sorgu Değerlendiricisi'*ne göndermektedir. Bu bilgi *İndeks Yöneticisi* tarafından da kullanılmaktadır. *Veri Saklama Yöneticisi* verileri ağacın elemanlarını ve özelliklerini tanımlanan şekilde saklar.

İndeks Yöneticisi, ağacın elemanları ve elemanların özellikleri üzerinde indeksler oluşturulabilmektedir. Bu indekslemede etiketler ve değerler birlikte değerlendirilmektedir. İndekslemelerde değerler üzerinde indeksleme yaparken veri

tipi olarak sayı ve yazı tipleri kullanılmaktadır. Oluşturulan bu indeksleme, sorgulamalarda verilerin ve ağacın elemanlarının hızlı bulunması amacı ile kullanılmaktadır.

Sorgulamalar için işlem öncelikle *Sorgu Parçalayıcısı* ile başlamaktadır. XQuery ve XPath gibi sorgulama dilleri kullanılarak gönderilen sorgular burada matematiksel ağaç işlemlerinden geçirilerek anlamlandırılmaktadır. Parçalanan veri *Sorgu Optimizasyoncusu* tarafından tekrar organize edilmekte ve sorguda kullanılacak elemanlar ve eleman özellikleri *Yapı Yöneticisi* tarafından belirlenmektedir. Sorgu sonucu elde edilecek olan bilgiler, aynı zamanda *Sorgu Değerlendiricisi* tarafından da değerlendirilmektedir. Bu işlemlerin ardından fiziksel ortamda bulunan hiyerarşik veri elemanları bulunarak sorgu sonucu oluşturulmaktadır.

Bu işlemler, ilişkisel veritabanlarının sorgulama mekanizmasına benzemekte, fakat XML veritabanları daha karmaşık işlemler yapabilmektedir. Örneğin, *Yapı Yöneticisi* sorgulamalarda kullanılacak bütün elemanları bulmak için kendi içinde ve verileri kullanarak bir sorgu yapmaktadır. XML dokümanlarının sorgulanmasında, dokümanda bulunan elemanların isimlerinin bilinmesine gerek yoktur. "Veritabanında bulunan ve adı X ile başlayan bütün etiketlerin değerlerini dön" gibi bir sorgulamada, sorgu sonucu dönmeden önce veritabanında verilen kurala uygun hangi etiketlerin bulunduğunun belirlenmesi gerekmektedir. Bu işlemden sonra sorgu sonuçları üretilmektedir.

İlişkisel veritabanlarında sorgu yapmak için hangi alanların kullanılacağının kesin olarak belirtilmesi gerekmektedir. Yukarıda örnek olarak verilen sorgunun ilişkisel veritabanlarında SQL dili kullanılarak gerçekleştirilmesi mümkün değildir. Bu tür sorgulamalar ancak yazılımlarda gerçekleştirilebilmektedir. XML veritabanlarında bu çeşit sorgular çok kolay bir şekilde gerçekleştirilebilmektedir. XML veritabanlarında bu tür sorgulamaların mümkün olması ve kullanılması bu veritabanını ön plana çıkarmaktadır. Fakat bu tür sorgular için XML veritabanlarının performanslarının geliştirilmesi gerekmektedir. Çünkü normal SQL sorgulamalarının yaptıkları işlemlerin yanında bu özelliğin gerçekleştirilmesi için ek bir katman

ihtiyacı olmakta ve bu katman sorgu sonuçlarının raporlanmasında zaman kaybına sebep olmaktadır.

Doğal XML veritabanları ilişkisel veritabanlarının yetersiz kaldığı durumlarda öne çıkan bir seçenektir. Veritabanlarında saklanan XML dokümanları ağaç şekline getirilerek indekslenebilmekte ve bu sayede normal dosya işlemlerinden daha fazla performanslı olmakta ve dosyalar daha az yer kaplayabilmektedir [71-73]. XML veritabanlarının ilişkisel veritabanları ile veri alışverişi yapabilmesi de mümkündür. XML veritabanlarında yönetilen veriler kontrol altında bulunmakta ve verilerin istenilen formata dönüştürülmesi sağlanabilmektedir.

Sonuç olarak; EWIRDB veritabanının doğal XML veritabanları ile yönetilmesi, bu veritabanının bütün özelliklerinin desteklenmesini sağlamaktadır. EWIRDB veritabanının XML dokümanlarına dönüştürülerek bu veritabanına girdi olarak verilmesi ile yönetim sağlanabilecektir.

6. XML VERİTABANLARINDA SORGULAMA (XQuery)

XML veritabanları üzerinde sorgulama yapabilmek için XQuery [12, 14] dilinin kullanımı standartlaşmaktadır. XQuery dili XML dokümanları arasından ilgilenilen elemanları ve elemanların özelliklerini bulmak ve getirmek için kullanılmaktadır. Bu dil ilişkisel veritabanlarında kullanılan SQL diline benzemekte ve IBM, Oracle, Microsoft gibi büyük şirketlerin de desteği ile W3C tarafından yönetilmektedir. XPath dili tek bir dosya üzerinde işlem yapmayı sağladığı için tam bir sorgulama dili olmadığı ve XQuery dilin tanımlanması gerekliliği ortaya çıkmıştır. XQuery dili XPath üzerine kurulmuştur. Aşağıda adı "*kitaplık.xml*" olan örnek bir XML dokümanı içeriği gösterilmiştir:

```
<?xml version="1.0" encoding="ISO-8859-9"?>
<kitaplık>
        <kitap kategori="İnternet">
                <başlık dil= "Türkçe">İnternet Dünyası</başlık>
                <yazar>Emin Ankaralı</yazar>
                <yıl>2003</yıl>
                <fiyat>10,00</fiyat>
        </kitap>
        <kitap kategori="Çocuk">
                <başlık dil= "Türkçe">Ali baba</başlık>
                <yazar>Hasan Erzurumlu</yazar>
                <yıl>2004</yıl>
                <fiyat>12,00</fiyat>
        </kitap>
</kitaplık>
```

XQuery dili ile *doc("kitaplık.xml")/kitaplık/kitap/başlık* yazılırsa *kitaplık.xml* dokümanı içinde *kitaplık* elemanının *kitap* alt elemanı bulunur ve bu alt elemanın *başlık* etiketi ve değeri sorgu sonucu olarak döner. Burada *doc("kitaplık.xml")* ile hangi XML dokümanının kullanılacağı belirtilir. Bu doküman içinde / işareti ile hiyerarşideki hangi alt elemanın sorgulanacağı seçilir. Hiyerarşik veri içerisinde

ulaşılmak istenen bir eleman için tüm üst hiyerarşileri kök etiketinden başlanarak / operatörü ile belirlenmesi gerekmektedir. Bu örnekte kök etiketten sonra iki seviye daha aşağıya inilmiştir. Yine aynı doküman üzerinde *doc("kitaplık.xml")/kitaplık/kitap[fiyat < 11]* şeklinde bir sorgu yapılırsa kitaplar arasında fiyatı 11 den küçük veriler bulunacaktır. Bu sorgu sonucunda *İnternet Dünyası* adındaki kitap verisi dönecektir. Elemanların özellikleri üzerinde de bu şekilde sorgular yapılabilmektedir. Özelliklerin sorgulamada kullanılabilmesi için @ işareti kullanılmalıdır. Örnekte verilen sorgu *doc("kitaplık.xml")/kitaplık/kitap[@kategori < 'İnternet']* şeklinde değiştirilirse kitaplardan kategorisi "İnternet" olan kitaplar sorgu sonucu olarak dönecektir. Bu sorgunun sonucunda yine *İnternet Dünyası* adındaki kitap verisi dönecektir. Yukarıda anlatılan özellikleri kullanmadan SQL diline benzer bir yapı kullanılmak istendiğinde kriter bilgisinin *WHERE* anahtar kelimesinden sonra belirlenmesi gerekmektedir. Örnek olarak verilen sorgu cümleciği yerine:

FOR $x in doc(kitaplık.xml)/kitaplık/kitap WHERE $x/@kategori = 'İnternet'
RETURN $x

cümleciği kullanılabilir. Burada dokümanda bulunan her bir kitap ele alınmış ve *WHERE* anahtar kelimesinden sonra yazılan koşula uyanlar bulunmuştur. Bu işlemden sonra *RETURN* anahtar kelimesinden sonra belirtilen eleman sorgu sonucu olarak dönmüştür. Bu sorgu bir önceki sorgu cümleciği ile aynı sonucu dönmektedir. Doküman içinde bulunan herhangi bir elemanı değişken olarak ele alarak daha sonra sorgu cümleciği içinde kullanmak için bir değişken ismi tanımlanır ve bu değişken isminin başına $ işareti eklenir. Bu örnekte tüm kitapların bir bir ele alınabilmesi için *x* adında değişken tanımlanmış ve bu değişken *WHERE* ve *RETURN* anahtar kelimelerinden sonra kullanılmıştır.

Sorgu sonuçları üzerinde sıralama yapabilmek için SQL dilinde de bulunan *ORDER BY* anahtar kelimesi kullanılır. Yukarıdaki sorgunun sonuçlarını kitap başlıklarına göre sıralamak için cümlecik:

FOR $x in doc(kitaplık.xml)/kitaplık/kitap WHERE $x/@kategori = 'İnternet'
ORDER BY $x/başlık RETURN $x

şeklinde güncellenmesi gerekmektedir. XQuery dilin genel yapısı *FOR , WHERE, ORDER BY, RETURN* şeklinde olduğu söylenebilir. Bu kullanım sırası zorunludur. *WHERE* ve *ORDER BY* anahtar kelimelerinin kullanılması zorunlu değildir. *FOR* anahtar kelimesinin yerine *LET* anahtar kelimesi de kullanılabilir. *FOR* ile *LET* arasındaki fark hiyerarşik verilerin sorgulanmasında ortaya çıkan bir durum içindir. *FOR* anahtar kelimesi kullanılarak bir değişken tanımlandığında sorgu cümleciğinde bulunan *RETURN* anahtar kelimesinden sonra belirtilen işlemler, her bir eleman için ayrı ayrı çalıştırılır. *LET* anahtar kelimesi ile tanımlanan değişken *RETURN* anahtar kelimesinden sonra gelen işlemler bir kere gerçekleştirilir, tüm veri blok olarak sorgu sonucuna gönderilir.

XQuery ile belirtilen sorguların sorgu sonuçları istenilen şekilde formatlanabilir. Sorgu sonuçlarını HTML olarak görüntülemek için gerekli etiketler belirtilerek görüntülenebilir. Bu işlemi gerçekleştirebilmek için aşağıdakine benzer bir sorgu cümleciği hazırlanmalıdır:

* { FOR $x in doc(kitaplık.xml)/kitaplık/kitap/başlık RETURN data($x) } *

Bu sorgu cümleciğinde görüldüğü gibi $ işareti ile belirtilmeyen tüm karakterler hiç değiştirilmeden, doğrudan sorgu sonuçlarında görüntülenir. Örnekte verilen *data*() fonksiyonu verilen değişkenin sadece değerinin sorgu sonucuna yansıtılmasını sağlar. Bu sorgu sonucunda aşağıdakine benzer bir sonuç ortaya çıkar:

* İnternet Dünyası
Ali baba
*

XQuery dilinin bu özelliği ile herhangi bir XML dokümanının başka bir formata dönüştürülmesi mümkün olmaktadır. XQuery dilinin bir özelliği de koşullu cümleler tanımlanabilmesidir. Koşullu cümleler yazabilmek için *IF* ve *ELSE* anahtar kelimeleri kullanılması gerekmektedir. Örnek bir sorgu cümleciği:

FOR $x in doc(kitaplık.xml)/kitaplık/kitap RETURN
IF ($x/@kategori='İnternet') THEN 'internet kitabı' ELSE 'diğer'

formatında olması gerekmektedir. Bu örnekte kategorisi "*İnternet*" olan her bir kitap için "*internet kitabı*" diğer durumda ise "*diğer*" yazısı yazılmış olacaktır. Koşullu cümleler içinde =, <, >, *!*=, <=, >= gibi operatörler kullanılabileceği gibi değerlerin karşılaştırılmasında *eq, lt, ne, le, ge* gibi operatörlerde kullanılabilmektedir. Ayrıca birden fazla durumun kontrol edilmesi gerektiği durumlarda da *and* ve *or* operatörleri de kullanılabilmektedir. Hiyerarşi içinde bir elemanın yeri bilinmiyorsa veya değişik hiyerarşilerde birden fazla aynı eleman bulunabiliyorsa bu durumda tanımlanacak sorgularda / operatörü yerine // operatörü kullanılarak tüm dokümanın içinde ilgili elemanların bulunarak sorgulanması sağlanabilir. Bu operatör aynı zamanda birden fazla doküman içinde sorgulama yapılarak istenilen elemanların bulunmasında da kullanılabilir. Bu operatörün kullanım şekli aşağıdaki gibi olmalıdır:

FOR $x in doc(kitaplık.xml)//başlık RETURN data($x)

Örnekte, XML dokümanının herhangi yerinde bulunan *başlık* adına sahip tüm etiketler bulunur. Bu operatörle XML dokümanın iç yapısının tam olarak bilinmesi gerekmemektedir. Bu işlemin bir dezavantajı şudur: sorgu sırasında tüm ağaç yapısının üzerinden geçilerek ilgili etiketlerin bulunması gerekmektedir. Bu da performans problemlerine sebep olabilecek bir durumdur [71]. Bu problemin önüne geçmek için indeksleme mekanizmasının düzgün kurulması gerekmektedir.

XQuery dilinin programlama dillerinde olan özelliklere sahip olması ve bu dil ile sorgu sonuçları üzerinde istenilen değişikliğin yapılabilmesi hiyerarşik verilerin yönetimi açısından çok büyük öneme sahiptir. XQuery dili sorguları ile hiyerarşik veriler, istemcilere istedikleri şekilde sunulabilmektedir.

7. HİYERARŞİK VERİLERİN BENZERLİKLERİNİ HESAPLAMA YAKLAŞIMLARI

İki hiyerarşik verinin benzerliğinin bulunması; veri madenciliğinde [28, 29, 31-33, 35, 41, 42, 47-52, 74-76], ağaçların birbirine en çok benzeyen özelliklerinin bulunmasında [23-27, 34, 40, 77-80], iki ağacın kullandığı ortak şemaların otomatik tespit edilmesinde [30, 37-39], hiyerarşik verilerin gruplandırılmasında [43-46, 81-91], ağaçların ortak alt hiyerarşilerinin bulunmasında [36] kullanılabilmektedir. Ağaçların benzerliklerinin hesaplanması, yukarıda da belirtilen konularda ilk olarak yapılması gereken işlemdir.

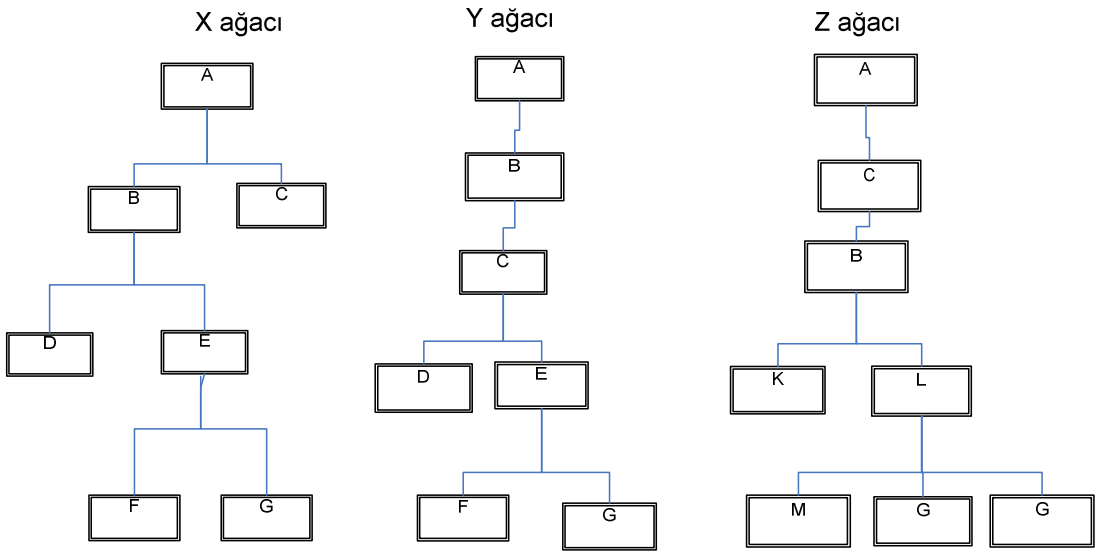
Hiyerarşik verilerin benzerliklerinin bulunmasında değişik yaklaşımlar kullanılmaktadır. Bu yaklaşımlar:

– Bir ağacı diğer ağaca benzetebilmek için gerekli olan işlemleri belirleyerek benzerliklerin bulunması,
– İki ağacın ortak dallarının belirlenmesi ile benzerliklerin bulunması,
– İki ağacın ortak elemanları ve seviyelerini kullanarak benzerliklerin bulunması,
– İki ağacın ortak elemanları ve elemanların değerlerini kullanarak benzerliklerin bulunması olarak özetlenebilir.

Benzerliklerin bulunması için kullanılan bu yaklaşımlar değişik çalışmalarda değişik şekillerde uygulanmıştır. Bazı çalışmalarda sadece yapısal benzerliklere odaklanılmış, bazılarında da yapısal ve içerik olarak benzerliklerine odaklanmıştır.

Şekil 7.1'de, yapısal olarak birbirine benzeyen ağaç yapıları verilmiştir. Bu ağaçlar aynı kök etiketlerine sahiptirler. Bu ağaçlar yapısal olarak birbirlerinden farklı olsalar da ortak özellikleri bulunmaktadır. X ağacı ile Y ağacı aynı etiketlere sahiptir. Fakat bu etiketlerin hiyerarşideki yerlerinde farklılıklar bulunmaktadır. C etiketi X ağacında ikinci seviyede (kök etiket birinci seviye kabul edilmiştir) Y ağacında da üçüncü seviyededir. X ağacındaki B etiketinin hiyerarşisi Y ağacında C etiketinde bulunmaktadır. Bu yüzden X ağacındaki D, E, F, G etiketlerinin seviyeleri Y

ağacında bir artmıştır. X ve Y ağaçlarının etiketleri benzemesine rağmen yapıları benzememektedir.



Şekil 7.1. Birbirleri arasında yapısal benzerlik olan ağaç yapıları

Y ve Z ağaçlarına bakıldığında hiyerarşilerin aynı olduğu fakat etiketlerin hepsinin ortak olmadığı görülmektedir. Z ağacında bulunan K, L ve M etiketleri Y ağacında bulunmamakta, Y ağacındaki D, E ve F etiketleri Z ağacında bulunmamaktadır. Bir diğer fark da B ve C etiketlerinin her iki ağaçta yer değiştirmiş olmalarıdır. Bunların yanında G etiketi Z ağacında birden fazla tekrarlanmıştır.

Örnekte verilen üç ağacın birbirlerine benzerliklerinin tanımlanmasında, ilgilenilen problem için gereksinimlerin dikkate alınması gerekmektedir. Y ağacı ile X arasındaki benzerlik ve Y ağacı ile Z arasındaki benzerliğin ne olduğu, bu iki benzerlikten hangisinin dikkate alınacağını belirlemek için bölümün başında tanımlanan yaklaşımlar kullanılmaktadır. Bu örnek ağaç yapıları ile benzerliklerin bulunması için kullanılan yaklaşımların açıklamaları aşağıda anlatılacaktır.

7.1. İki Ağacı Birbirine Dönüştürmek İçin İşlem Yaparak Benzerliğin Bulunması

Bu yöntem, bir ağacın diğer ağaca benzetilmesi için nelerin yapılması gerektiğinin belirlenmesi ve yapılacak her bir işlemin benzerliği ne kadar etkilediğinin katsayılar tanımlayarak bir sonucun elde edilmesine dayanmaktadır [23-26, 34, 54, 55].

İki ağacın farkının bulunması, *ağaç güncelleme mesafesi* olarak tanımlanan bir yöntemle gerçekleştirilmektedir. Bu yöntemde, ilk ağacın diğer ağaca benzemesi için yapılması gereken adımlar tanımlanır. Bir ağacın her bir elemanı diğer ağaçta aranır ve sonuçları aşağıdaki şekilde değerlendirilir:

– Ağaçtaki eleman diğer ağaçta var ve hiyerarşisi aynı ise bu eleman üzerinde değişiklik yapılmayacağına karar verilir, eğer elemanların sahip oldukları değerler karşılaştırılacaksa iki değerin yazı olarak benzerliklerine bakılır.
– Ağaçtaki bir eleman diğer ağaçta var fakat hiyerarşisi farklı ise bu elemanın hiyerarşisinin değiştirilmesi gerektiğine karar verilir, eğer elemanların sahip oldukları değerler karşılaştırılacaksa iki değerin yazı olarak benzerliklerine bakılır.
– Ağaçtaki bir eleman diğer ağaçta yoksa bu elemanın silinmesi gerektiğine karar verilir, her iki ağaçta da aynı eleman bulunmadığı için elemanların değerleri arasında karşılaştırma yapılamaz.
– Diğer ağaçta bulunan bir eleman ilk ağaçta yoksa bu elemanın birinci ağaca eklenmesi gerektiğine karar verilir, her iki ağaçta da aynı eleman bulunmadığı için elemanların değerleri arasında karşılaştırma yapılamaz.

Her bir ağaçtaki her bir eleman için yukarıda belirtilen adımlardan geçilir, işlem sonuçları bulunur ve elde edilen sonuçlar tanımlanan ağırlıklarla çarpılarak toplanır. Elde edilen sonuç iki ağacın birleşiminin eleman sayısına bölünerek 0-1 arasında bir benzerlik bulunur. Bu işlem için kullanılan eşitlik Eş 7.1'de verilmiştir.

$$benzerlik[X,Y] = \frac{\sum_{e \in X} islem[e,Y]}{toplam[X \cup Y]} \qquad (7.1)$$

Bu eşitlikte X ve Y karşılaştırma yapılacak ağaçları, *benzerlik[X, Y]* iki ağaç arasındaki benzerliği tanımlamaktadır. *islem[e,Y]* X ağacında bulunan bir eleman üzerinde yapılacak işlemin ağırlığını belirler. Her iki ağaçta bulunan elemanların birleşiminin toplamı *toplam[X U Y]* ile gösterilir.

X ve Y ağaçlarındaki (Bkz. Şekil 7.1) benzerliği bulmak için birebir benzeyen elemanlar için 1,0, yer değiştirme gereken işlemler için 0,5, silme veya ekleme gerektiren işlemler için 0,2 ağırlıkları kullanıldığında aşağıdaki çizelgede (Çizelge 7.1) belirtilen adımların gerçekleştirilmesi gerekmektedir.

Çizelge 7.1. X ağacının Y ağacına benzetilmesi

Yapılması Gereken İşlem	İşlem İçin Kullanılacak Ağırlık
A elemanı için hiçbir şey yapma	1,0
B elemanı için hiçbir şey yapma	1,0
C elemanını B elemanın altına taşı	0,5
D elemanını C'nin altına taşı	0,5
E elemanını C'nin altına taşı	0,5
F elemanı için hiçbir şey yapma	1,0
G elemanı için hiçbir şey yapma	1,0

X ağacının Y ağacına benzetilmesi için kullanılacak ağırlıkların toplamı 5,5 (1,0 + 1,0 + 0,5 + 0,5 + 0,5 + 1,0 + 1,0)'dır. İki ağacın tüm elemanları ortak olduğu için birleşimlerinin eleman sayısı 7'dir. Bu değerlerle iki ağacın benzerliği 5,5/7 = 0,785 olarak bulunur.

Aynı ağırlıklar kullanılarak Y ağacının Z ağacına (Bkz. Şekil 7.1) benzerliği bulunmak istendiğinde aşağıdaki çizelgede (Çizelge 7.2) verilen sonuçlar elde edilir. Aşağıda verilen işlem sonucunda yapılacak işlemlerin ağırlıkları toplamı 3,6 ve iki

ağacın birleşiminde bulunan eleman sayısı toplamı 11 olur. Bu işlem sonucunda iki ağacın benzerliği 3,6/11 = 0,327 olarak bulunur.

Çizelge 7.2. Y ağacının Z ağacına benzetilmesi

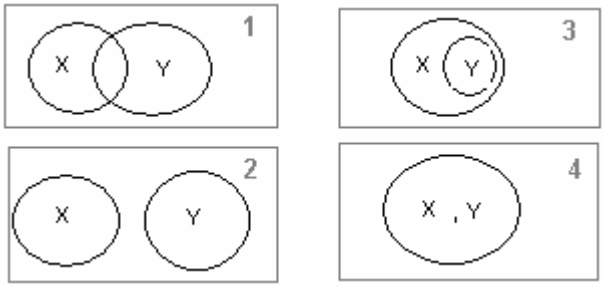
Yapılması Gereken İşlem	İşlem İçin Kullanılacak Ağırlık
A elemanı için hiçbir şey yapma	1,0
B elemanını C'nin altına taşı	0,5
C elemanını A elemanın altına taşı	0,5
D elemanını sil	0,2
E elemanını sil	0,2
F elemanını sil	0,2
K elemanını ağaca ekle	0,2
L elemanını ağaca ekle	0,2
G elemanı L'nin altına taşı	0,2
M elemanını ağaca ekle	0,2
İkinci bir G elemanını L'nin altına ekle	0,2

X ile Y ağaçlarının birbirine benzetilmesinin Y ile Z ağaçlarının birbirine benzetilmesinden daha kolay olduğu görülmektedir. Elde edilen benzerlik sonuçları da bunu göstermektedir (*benzerlik[X,Y]* = 0,785 ve *benzerlik [Y,Z]* = 0,327). Yukarıda verilen örneklerde farklı etiketlenmiş olan ağaç elemanlarının birbirinden farklı anlamlar ifade ettiği varsayılmıştır. Eğer iki eleman etiket olarak farklı, fakat sakladıkları bilgiler aynı ise ağaçların elemanlarının mantıksal olarak aynı şeyi mi ifade ettiklerini belirlemek gerekmektedir.

Yukarıda verilen örneklerde X ağacının Y ağacına benzetilmesi için yapılması gereken işlem ile Y ağacının X ağacına benzetilmesi için gerekli olan işlemler farklı olabilir. Bu durumu önlemek için her iki benzerliğin bulunup ortalamasının alınması ile simetrik olacak bir benzerlik tanımı yapılabilir:

$$benzerlik[X,Y] = \frac{\sum_{e \in X} islem[e,Y] + \sum_{e \in Y} islem[e,X]}{2 \times toplam[X \cup Y]} \tag{7.2}$$

İki ağacı birbirine benzetmek için yapılan işlemlerin simetrik olmayan sonuçlar vermesinin sebebi her iki ağaçta da aynı elemanların bulunmamasından kaynaklanmaktadır. İki ağacın elemanlarını bir küme olarak kabul edip, bu iki kümenin benzerliklerini bulmaya çalıştığımızı varsayarsak Şekil 7.2'de verilen durumlardan biri ile karşılaşılır [15].

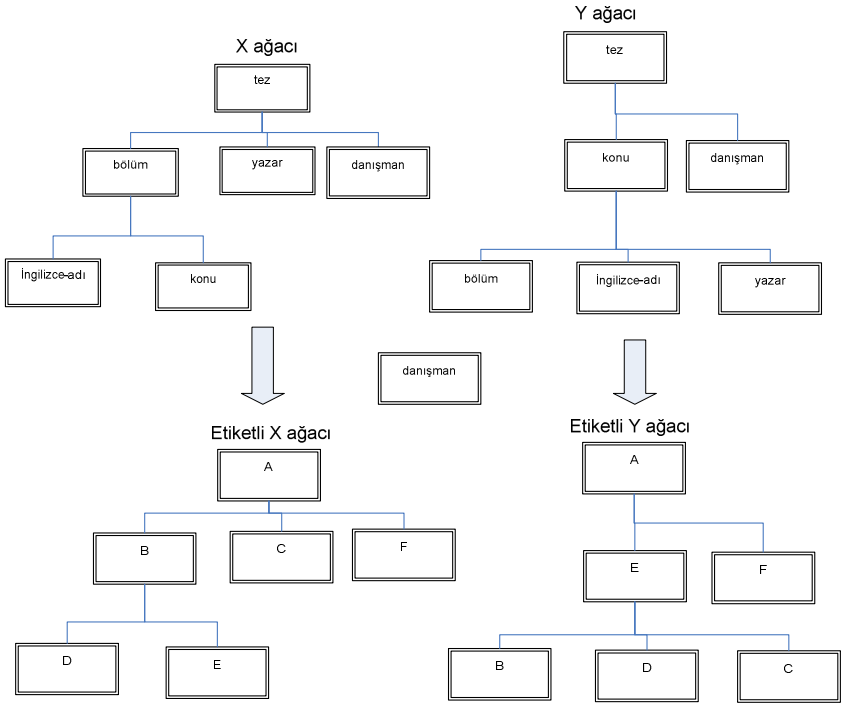


Şekil 7.2. X ve Y ağaç yapılarının benzerlik hesaplamalarındaki muhtemel durumları
1. X ve Y ağaçlarının ortak elemanları ve ortak olmayan elemanları var
2. X ve Y ağaçlarının hiç ortak elemanı yok
3. X ağacı Y ağacının bütün elemanlarına sahip fakat Y ağacında bulunmayan elemanlara sahip
4. X ve Y ağaçlarının elemanları birbirleri ile aynı

Bu şekilde X ve Y birer ağaç olarak düşünülmüştür. X ile Y arasındaki benzerlik ile Y ile X arasındaki benzerliğin aynı olması istenirse ağaçta yapılacak ekleme ve silme işlemlerine aynı ağırlık verilmesi gerekmektedir. Eğer farklı ağırlıklar verildiğinde Eş. 7.2'de belirtilen eşitliğe benzer bir eşitlik kullanılabilir. Bu eşitlik yerine her iki benzerlik hesaplandıktan sonra, en küçük değere sahip veya en büyük değere sahip değer benzerlik değeri olarak kullanılabilir.

7.2. Ağaçların Ortak Yolları Kullanılarak Benzerliğin Bulunması

Bu yaklaşımda [41, 50-52] ağaçların bütün yollarının dizi haline getirilmesi gerekmektedir. Diziler içinde işlem yapabilmek için ağacın her bir elemanı etiketlenir. Diğer ağaçta da aynı işlem gerçekleştirilir. Eğer her iki ağaçta aynı eleman var ise bunlara aynı etiketler verilir. Mantık olarak aynı olan elemanlar var ise bu elemanlar hazırlanan sözlüklerden aynı anlama gelip gelmediği bakılır. Aynı anlamda olanlar için aynı etiketler verilir. Bu işlemden sonra ağaç yapılarının yolları diziler haline getirilir. Şekil 7.3'de ağaçların nasıl etiketlendiği gösterilmiş ve Çizelge 7.3'de bu ağaçlarda bulunan yollar verilmiştir.



Şekil 7.3. Ağaçların etiketlenmesi

Şekilde ilk önce X ağacı etiketlenmeye başlanır. Daha önce tekrar etmeyen her bir elemana önceden kullanılmayan bir etiket verilir. Bu işlem ilk ağaç tamamlanıncaya kadar devam edilir. Diğer ağaçta etiketleme yaparken sıradaki elemanın diğer ağaçta bulunup bulunmadığı kontrol edilir. Daha önce kullanılmış ise aynı etiket, kullanılmamış ise yeni bir etiket verilir.

Çizelge 7.3. Etiketli X ve Y ağaçlarının yolları

Etiketli X Ağacının Yolları	Etiketli Y Ağacının Yolları
A, B, D	A, E, B
A, B, E	A, E, D
A, C	A, E, C
A, F	A, F

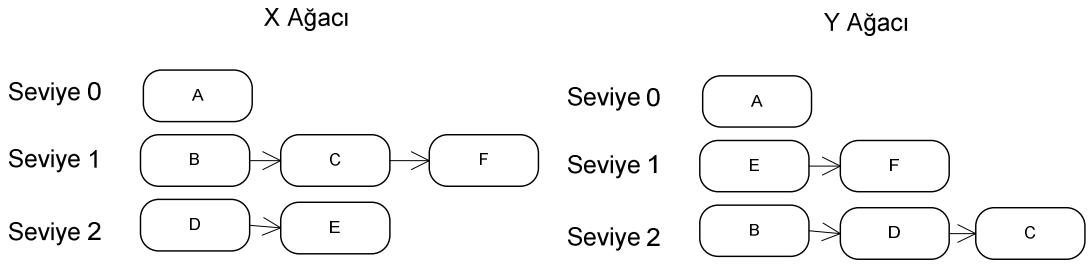
Etiketleme sonucu elde edilen ağaçların yolları belirlenir. Çizelge 7.3'te, örnekte verilen ağaçların yolları gösterilmektedir. Bu yollar arasındaki benzerlikleri bulmak için her iki ağaçta da ortak olan en uzun yollar belirlenmeye çalışılır. Belirlenen ortak yolların sayısı iki ağacın birleşimi ile elde edilen ortak yolların sayısına bölümü ile 0-1 aralığında bir benzerlik elde edilmektedir. Yukarıdaki örnekte sadece bir ortak yol bulunmaktadır. İki ağaçta toplam yedi farklı yol bulunmaktadır.

Ortak yolların bulunmasına alternatif olarak, yollarda bulunan elemanların yerleri farklı da olsa ortak olanların belirlenmesi de benzerlikte kullanılabilir. Örnekte verilen ağaçların yollarında sadece A ve F elemanlarından oluşan yol ortaktır. Buna karşılık etiketli X ağacındaki A, B, D yolu ile Y ağacındaki A, E, D yolu arasında iki eleman ortaktır.

7.3. İki Ağacın Ortak Elemanları ve Seviyeleri Kullanılarak Benzerliğin Bulunması

Bu yaklaşımda, ağaçların hiyerarşisi düşünülmeden her bir eleman diğer ağaç üzerinde aranmakta ve ortak elemanların sayısı ve hiyerarşik seviyeleri bulunarak benzerlik tanımlanmaya çalışılmaktadır [44].

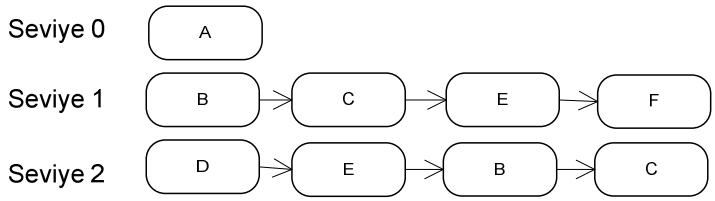
Bu yaklaşımda, ağacın elemanlarının hiyerarşideki yeri göz önünde bulundurulmamakta, sadece seviyelerine göre benzerlik tanımlanmaya çalışılmaktadır. Bu yöntemde en üstte bulunan elemanların benzerliğinin, alt seviyelerde bulunan benzerlikten daha yüksek olmasını sağlamak için her bir seviyeye değişik ağırlıklar verilmektedir.



Şekil 7.4. X ve Y ağaçlarının seviyelerine ayrıştırılması

Ağaçlar öncelikle etiketlenmekte ve ardından seviyelerine göre ayrıştırılarak bir matris oluşturulmaktadır. Bir önceki bölümde verilen ağaçların (Bkz. Şekil 7.3) seviyelerine göre nasıl ayrıştırıldığı Şekil 7.4'te verilmiştir.

Şekil 7.4'te X ve Y ağaçlarında aynı seviyede bulunan bütün elemanlar aynı seviyeye getirilmiştir. Benzerlik hesaplaması yapılırken bu iki ağacın birleşimi bulunur ve bir matrise dönüştürülür. Bu matris oluşturulurken aynı seviyede bulunan ve ortak olan elemanlar bir kere kullanılır. İki ağacın birleşiminden elde edilen matris Şekil 7.5'de verilmiştir.



Şekil 7.5. X ve Y ağacının birleşiminden elde edilen seviye matrisi

İki ağacın birleşiminden elde edilen matris ve ağaçların seviye bilgileri kullanılarak aşağıdaki eşitlikle (Eş. 7.3) benzerlik bulunur.

$$Benzerlik_{x\to y} = \frac{0{,}5\times\sum_{i=0}^{L-1} CN_x^i \times (r)^{L-i-1} + 0{,}5\times\sum_{j=0}^{L-1} CN_y^j \times (r)^{L-j-1}}{\left(\sum_{k=0}^{L-1} N^k \times (r)^{L-k-1}\right)\times Z} \tag{7.3}$$

Bu eşitlikte CN_x^i X ağacında bulunan ve i seviyesindeki ortak elemanların toplamı, CN_y^j Y ağacında bulunan ve j seviyesindeki ortak elemanların toplamı, L ağaçta bulunan seviyeyi, r 0-1 arasında bir ağırlık katsayısı, N^k ağaçta k seviyesinden

bulunan eleman sayısı, Z de grupta bulunan ağaç sayısıdır. İki ağacın karşılaştırılmasında Z 1 değerini alır. Elde edilen sonuç 0 ile 1 arasındadır. 0 iki ağacın hiç benzemediği, 1 iki ağacın birbiri ile aynı bilgileri içerdiği anlamına gelir. Bu eşitlikte X ağacının Y ağacına benzerliği bulunur.

Şekil 7.4 ve Şekil 7.5 kullanılarak X ağacının Y ağacına benzerliğini hesaplamak için ağırlık katsayısı olarak 2 değeri kullanıldığında 0,4375 değeri elde edilir (Eş. 7.4).

$$Benzerlik_{x \to y} = \frac{0,5 \times (1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) + 0,5 \times (1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)}{(1 \times 2^2 + 4 \times 2^1 + 4 \times 2^0) \times 1} = 0,4375 \qquad (7.4)$$

Bu yaklaşımda hangi elemanın hangi ağaç yolu üzerinde bulunduğu bilgisi kaybedilmiştir. Büyük ve içeriği bilinmeyen ağaç yapılarında, bu yaklaşımla hızlı sonuçlar elde edilmektedir.

7.4. İki Ağacın Elemanları ve Değerleri Kullanılarak Benzerliğin Bulunması

İki ağacın karşılaştırılmasında ağacın elemanlarının sakladıkları değerler çok önemlidir. Aynı hiyerarşik yapıya sahip olsalar bile değerleri birbirinden çok farklı olan elemanların benzerliği çok azdır. Bunun için ağacın değerlerinin benzerliğinin ön planda tutulması ve hiyerarşinin de benzerlikte yer alması gerekmektedir. Bu yaklaşımda ağacın değerlerinin karşılaştırılması ön plana çıkmaktadır [16, 86].

İki ağacın elemanları birbirlerinden farklı veri tipleri barındırabilirler. Örneğin, bazı elemanlar sadece yazı saklarken bazıları sayısal değerler saklayabilir. Bunun için her bir elemanın kendi tipi dikkate alınarak benzerlik hesaplaması yapılmalıdır. Elemanların veri tiplerini dikkate alarak benzerlik hesaplamak için verilerin ortak şemalara sahip olması gerekmektedir. Ortak şemada hangi verinin hangi tip değeri saklayacağı, hangi hiyerarşilerin oluşturulabileceği gibi bilgiler saklanır. Hiyerarşik yapıda olan XML dokümanları için bu bilgiler XSD dokümanlarında bulunur.

İki ağacın karşılaştırılmasında, veri tiplerinin yanında tekrar eden elemanların da benzerlikte göz önünde bulundurulması gerekmektedir. Diğer yaklaşımlarda birden fazla tekrar eden elemanlar çoğunlukla göz ardı edilmektedir. Çünkü diğer

yaklaşımlarda ağaçların yapısal benzerliklerinin belirlenmesine odaklanılmıştır. Bu yaklaşımda, verilerin şemaları tanımlı olduğu için muhtemel hiyerarşiler bilinebilmektedir.

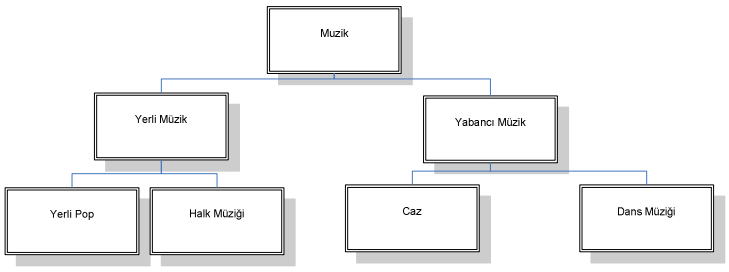
İki değerin karşılaştırılmasında veri tipleri göz önünde bulundurulduğunda her veri tipi için ayrı ayrı benzerlik tanımı yapılması gerekmektedir. Sayısal değerler için benzerlik aşağıda belirtilen eşitliğe göre (Eş. 7.5) belirlenebilir.

$$Benzerlik(X,Y) = 1 - \frac{|X - Y|}{|X| + |Y|} \tag{7.5}$$

Bu eşitlikte X ve Y karşılaştırılacak olan sayısal değerlerdir. Mantıksal değerler için de Eş.7.6'te belirtilen şekilde bulunabilir.

$$Benzerlik(x,y) = (x \otimes y)' \tag{7.6}$$

Yazıların karşılaştırılmasında ortak karakterlerin oranı bulunarak hesaplanabilir. Bu hesaplamada Eş. 7.5 kullanılabilir. X karşılaştırılacak ilk yazı, Y ikinci yazıdır. |X - Y| iki yazılın farklı olan karakterleri sayısı, |X| ilk yazılın karakter uzunluğu, |Y| ikinci yazının karakter uzunluğu olarak tanımlanması gerekmektedir.



Şekil 7.6. Yazıların kategorize edilmesi

Bu şekilde tanımlanan benzerliklerde, yazım farkları olan ve aynı anlama gelen yazılarda doğru sonuçlar alınmamaktadır. Yazıların benzerliklerinin hesaplanmasında yazılar önceden kategorize edilerek (Şekil 7.6) yeni bir benzerlik tanımı yapılabilir.

Şekil 7.6'de kişilerin hobilerini saklayan bir ağaç elemanının alabileceği değerler verilmiştir. Bir ağaç elemanının değeri ile başka bir ağaç elemanının değeri

karşılaştırılırken bu yapı kullanılarak benzerlik tanımlanabilir. Bu yöntemde kullanılan genel eşitlik Eş. 7.7'da verilmiştir.

$$Benzerlik(x,y) = \prod_{k}^{m} w_k \tag{7.7}$$

Bu eşitlikte x yazısının y yazısına benzerliği tanımlanmıştır. k hiyerarşi içinde x yazısından y yazısına ulaşılıncaya kadar üzerinden geçilecek elemanlar için bir indekstir. x elemanından y elemanına geçişte kaç tane farklı dalın bulunduğu w ile belirtilir. Geçişlerde bulunan her bir w çarpılarak x değerinin y değerine ne kadar benzediği bulunur. *Müzik* değerinin *Halk Müziği* değerine ne kadar benzediğini bulurken *Müzik* elemanından *Halk Müziği* elemanına ulaşmak için kullanılacak yollar bulunur. Bu örnekte, iki eleman arasında bir yol tanımlıdır ve benzerlik 1'dir. Halk müziğinin *Müzik* değerine ne kadar benzediği bulunurken de yollar belirlenmelidir. *Halk Müziği Türkçe Müziğe* 1/2 benzemektedir. *Türkçe Müzik* değeri *Müzik* değerine 1/2 benzemektedir. Bu değerlerin çarpımında 0,25 benzerlik değeri bulunur. Her bir elemanın değeri hesaplandıktan sonra hiyerarşik yapıda tüm elemanların değerleri toplanarak iki ağacın benzerliği bulunur. Bu yaklaşımda hesaplamalar ağacın yapraklarından başlanması gerekmektedir.

$$Benzerlik^{F}_{örnek}(F_1,F_2) = \frac{1}{N_1}\sum_{i=1}^{N} Benzerlik^{C_i}_{eleman}(F_1,F_2) \tag{7.8}$$

Eş. 7.8'de iki ağacın benzerliğinin bulunması için gerekli eşitlik verilmiştir. C ağaçta bulunan bir eleman, N elemanın alt elemanları, F_1 birinci ağaç, F_2 ikinci ağaç, N_1 birinci ağacın kök elemanının alt elemanları ve toplam içinde belirtilen eşitlik bir elemanın diğer eleman ile karşılaştırılmasında kullanılacak denklemdir. Her bir eleman için Eş. 7.5, Eş. 7.6 ve Eş. 7.7'da verilen denklemler kullanılabilir.

8. UYGULANAN YÖNTEM

EWIRDB veritabanın modern bir veritabanı ile yönetilmesi ve etkin bir şekilde kullanılması için yapılan bu çalışma, veritabanı hakkında teknik bilgi edinebilmek için kaynak araştırması ile başlamıştır. Bu konuda yapılan diğer çalışmalar incelenerek teknik olarak EWIRDB veritabanı yapısı öğrenilmiştir. Ardından, EWIRDB verilerin nasıl XML dokümanlarına dönüştürülebileceği araştırılmıştır. XML dokümanlarının sahip olması gereken şema belirlenmiştir. Bu işlemler gerçekleştirilirken örnek EWIRDB verileri üretilmiştir. Örnek veriler gerçekte kullanılan verilerle aynı karakteristiklere sahip olacak şekilde hazırlanmıştır. Örnek EWIRDB verilerinin yazılımla yönetilebilmesi için gerekli sınıf tasarımları yapılmış, verilerin XML formatına dönüştürülmesi ve XML dokümanları ile nesne oluşturulması için gerekli teknolojiler araştırılmış ve kullanılmıştır. EWIRDB verilerinin XML dokümanlarına dönüştürülmesinin ardından XML veritabanları araştırılmış ve çalışmada eXist [12] XML veritabanının kullanılmasına karar verilmiştir. Bu veritabanında bir kullanıcı rolü tanımlanmış ve EWIRDB verilerinden XML dokümanına dönüştürülmüş veriler, eXist veritabanı içinde tanımlanan bir *koleksiyon* içine eklenmiştir.

Veriler, veritabanında tanımlanan *koleksiyon* içinde ayrı ayrı dosyalarda saklanmıştır. Veritabanında bulunan XML dokümanlarında sorgulama yapmak için XQuery dilinin yapısı araştırılmış ve sorguların hazırlanabilmesi için gerekli alt yapı öğrenilmiştir. Bu işlemlerin ardından XML veritabanında bulunan verilerin yönetimi için gerekli servisler tanımlanmıştır. Tanımlanan servislerin gerçekleştirilmesi için gerekli tasarım yapılmış ve örnek uygulama geliştirilmiştir.

Örnek uygulamanın internet tarayıcısı üzerinden çalışabilmesi için gerekli internet geliştirme teknolojileri de araştırılmış ve bu teknolojilerden JSF (Java Server Faces) teknolojisi örnek uygulamada kullanılmıştır. Ayrıca hiyerarşik verilerin ağaç olarak internet tarayıcısında daha etkin çalışması amacıyla hiyerarşik verilerin gösteriminde, ağaç yapısı kullanılmış ve ağacın tüm elemanlarının bir anda yüklenmemesi, kullanıcının istekte bulunduğunda yüklenmesi için AJAX (Asynchronous JavaScript and XML) alt yapısı tasarıma dahil edilmiştir.

Verilerin örnek uygulama ile sunulumu sağlandıktan sonra iki hiyerarşik verinin benzerliği konusunda yapılan çalışmalar incelenmiş ve bu çalışmalar da göz önüne alınarak EWIRDB verileri arasındaki benzerlikleri tanımlayacak, verilerin yapısal ve içerik bilgilerini dikkate alan bir metot sunulmuştur. Bu metot kullanılarak verilerin benzerlikleri bulunmuş ve sonuçlar raporlanmıştır.

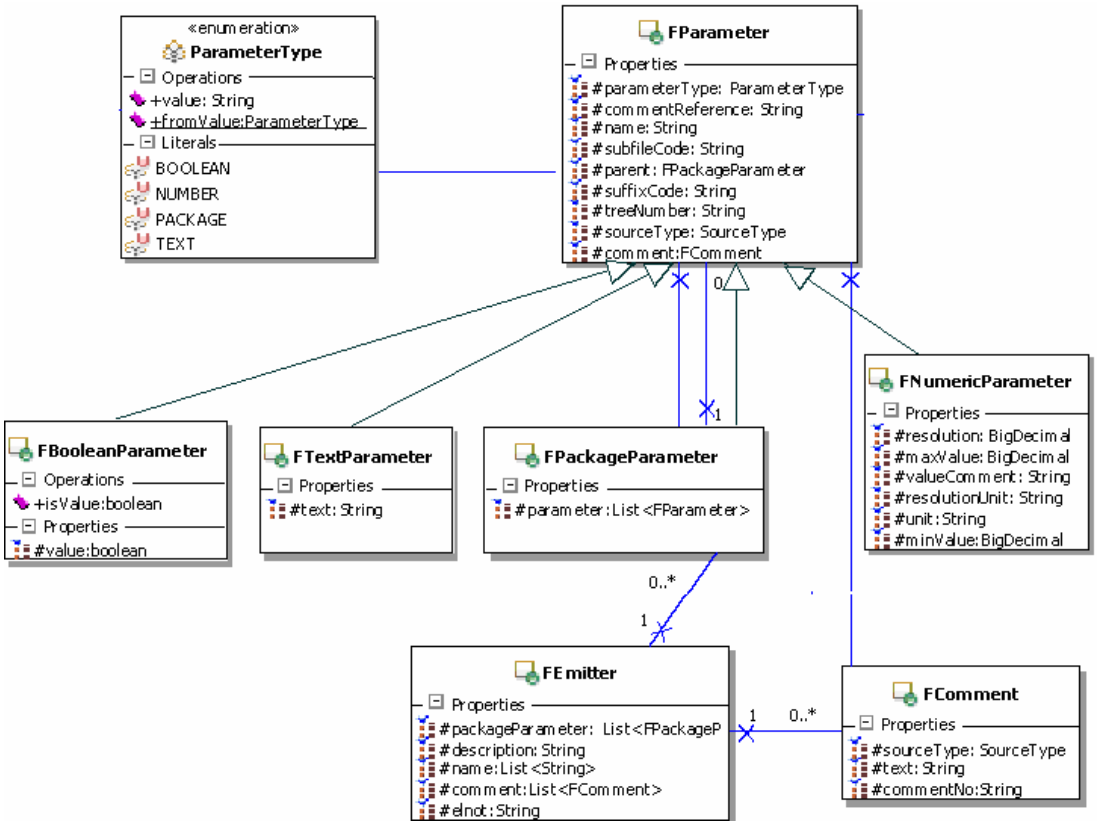
Hiyerarşik verilerin gruplandırılması üzerine yapılan çalışmalar incelenmiştir. Benzerlik metodunun tanımlanmasının ardından, elde edilen ve kaynağı belli olamayan bir verinin veritabanında bulunup bulunmadığını belirlemek için ağaçların yapısal benzerliklerine dayanan bir gruplama metodu tanımlanmıştır. Bu metot ile veritabanında bulunan ve birbirine benzer olan verilerin gruplandırılarak en az işlemle, hızlı ve doğru bir şekilde veritabanında arama yapılması sağlanmıştır.

Çalışma sonucunda, EWIRDB verileri XML dokümanları olarak XML veritabanında yönetilebilir hale getirilmiştir. İki hiyerarşik verinin benzerliğinin belirlenmesi için gerekli isterler ortaya çıkarılmış ve bu isterleri karşılayacak adaptif benzerlik bulma metodu tanımlanmıştır. Veritabanına hızlı erişim için hiyerarşik verilerin yapısal özellikleri kullanılarak gruplama metodu tanımlanmış ve veritabanında bulunan benzer verilerin bulunmasında benzerlik metodu ile birlikte kullanılmıştır.

Aşağıda, çalışmada gerçekleştirilen aktiviteler ve sonuçları detaylı olarak anlatılacaktır.

8.1. Hiyerarşik Verilerin XML Formatına Dönüştürülmesi

EWIRDB verilerinin parçalanıp XML formatına dönüştürülmesi için öncelikle yapının tanımlandığı dosya üzerinde incelemeler yapılmıştır. Bu dosyayı parçalayabilmek için gerekli sınıflar hazırlanmıştır. Bu sınıflar, dosyadan okunan verilerin nesneler olarak saklanmasına olanak vermiştir. Dosyanın işlenip verilerin saklanması için saha sınıfları dosyanın yapısına uygun olarak tasarlanmıştır. Bu sınıflar, verilerin hiyerarşik şekilde saklanmasına yardımcı olmuştur. Tanımlanan saha sınıflarının sınıf diyagramı aşağıda verilmiştir.



Şekil 8.1. EWIRDB verilerini parçalamada kullanılan saha sınıfları

EWIRDB dosya yapısı parçalanarak içinde bulunan verilerin saklanacağı saha sınıflarının isimleri *F* harfi ile başlatılmıştır. Tüm veriler dosyada benzer formatta bulunmaktadır. Dosyalarda, alanların bazı değerleri tanımlanmış bazı değerleri de tanımlanmamıştır. Tüm verilerde ortak olabilecek parametreler *FParameter* sınıfında saklanmış, verilere özel olan bilgiler için de *FTextParameter*, *FPackageParameter*, *FBooleanParameter* ve *FNumericParameter* sınıfları tanımlanmıştır. Bu sınıflar, *FParameter* sınıfından türetilerek oluşturuldukları için tüm ortak parametreler bu sınıflarda da bulunmaktadır. *FParameter* sınıfı, ayrıca parametre için tanımlanan açıklama bilgilerini saklamak amacıyla *FComment* sınıfını içermektedir. *FEmitter* sınıfı, parametrelerin hiyerarşik olarak saklanmasını sağlayan ve bu hiyerarşiyi kendi içinde saklayan sınıftır. Bu sınıf, kök elemanı saklamakta ve ek olarak açıklama bilgilerine referans tutmaktadır. Verilerin birbirleri arasında hiyerarşik olarak saklanması için *FParameter* sınıfı, *FPackageParameter* sınıfının bir referansını saklayarak bir elemanın çocuklarının olmasını sağlamaktadır. Veri saklayan tüm sınıflar bu sınıftan türedikleri için kendi hiyerarşilerini kendileri bilmektedir.

FParameter sınıfından türeyen tüm sınıflar ek parametrelere sahipse bunları kendi sınıfları içinde tanımlamaktadır. *FTextParameter* sınıfı yazı tipinde olan verilerin saklanması için, *FBooleanParameter* sınıfı doğru/yanlış bilgisi saklayan veriler için ve *FNumericParameter* sınıfı da sayısal değerlere sahip verilerin saklanması için kullanılmaktadır. *FPackageParameter* sınıfı hiyerarşide bulunan, bir değer saklamayan ve sadece verilerin organizasyonunda yer alan parametreler için kullanılmaktadır.

FParameter sınıfı, saklanacak olan parametrenin adını, tipini, açıklamasını, hiyerarşide kimin çocuğu olduğunu, veri kaynağını ve bu eleman için tanımlanan hiyerarşi numarasını saklamaktadır. Bunlara ek olarak bu sınıf bazı yardımcı alanlara (bölüm, öncelik) da sahiptir. Bu değerler dosyalarda bulunduğu için saklanmakta ve bu değerler ağaç oluşturulurken veya analiz işlemlerinde kullanılmaktadır.

EWIRDB yapı dosyası parçalandıktan sonra EWIRDB verilerinin bulunduğu dosyaların parçalanması işlemine geçilmiştir. Bu dosyalar aynı saha sınıfları ile parçalanarak ağaç hiyerarşisi oluşturulmuştur. Oluşturulan ağaç hiyerarşilerinin XML dokümanlarına yazılması için XSD şeması hazırlanmış ve bu şema kullanılarak dosyalar oluşturulmuştur.

XSD dokümanını kullanarak XML dokümanları oluşturmak için Java 6.0 sürümü ile birlikte gelen JAXB (Java API for XML Binding) altyapısı kullanılmıştır. Bu altyapıyı kullanabilmek için oluşturulan XSD dokümanını kullanabilecek Java sınıfları oluşturulmuştur. Oluşturulan bu sınıflar yardımı ile hiyerarşik veriler XML olarak veritabanına kaydedilebilir hale gelmiştir. Oluşturulan XSD dokümanında tanımlanan isim uzayı, bu çalışmayı ayırt edebilmek için *http://www.gazi.edu.tr/bm/24290921/thesis/v1.0* olarak tanımlanmıştır. Dosya içinde tanımlanan veri tipleri aşağıdaki gibidir:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
        targetNamespace="http://www.gazi.edu.tr/bm/24290291/thesis/v1.0"
```

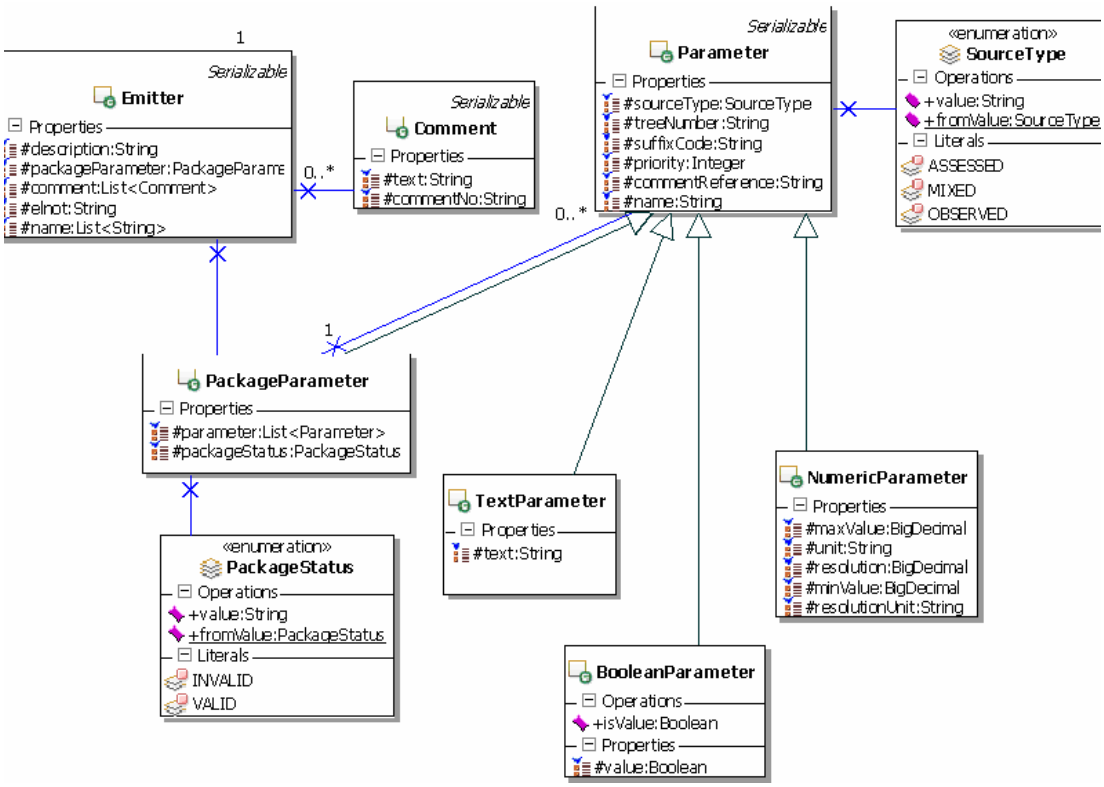
```
        xmlns="http://www.gazi.edu.tr/bm/24290291/thesis/v1.0"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
…
<xs:element name="emitter" type="emitter"/>
<xs:element name="booleanParameter" type="booleanParameter"/>
<xs:element name="parameter" type="parameter"/>
<xs:element name="comment" type="comment"/>
<xs:element name="coordinateParameter" type="coordinateParameter"/>
<xs:element name="packageParameter" type="packageParameter"/>
<xs:element name="numericParameter" type="numericParameter"/>
<xs:element name="textParameter" type="textParameter"/>
```

XSD dokümanları için kullanılacak sınıflar ile dosyaların parçalanmasında kullanılan saha sınıfları yapı olarak birbirine çok benzemektedir. Bu sınıflarda bazı alanlar daha detaylandırılmıştır. Dosya parçalanma işlemi tamamlandıktan sonra saha sınıflarının XSD dokümanının kullanacağı sınıflara dönüştürülmesi gerekmiştir. Bu işlemi gerçekleştirmek için gerekli sınıflar tasarlanmıştır. Bu sınıf, saha sınıflarını XSD dokümanlarının istediği sınıflara dönüştürmekte ve ardından elde edilen sınıflar JAXB altyapısı kullanılarak XML dokümanlarına yazmaktadır. Eleman olarak tanımlanan etiket isimlerinin açıklaması yine aynı doküman içindedir. Örnek eleman tanımları aşağıda verilmiştir.

```
<xs:complexType name="emitter">
        <xs:sequence>
                <xs:element ref="description" minOccurs="0"/>
                <xs:element ref="name" maxOccurs="unbounded"
                                minOccurs="0"/>
                <xs:element ref="packageParameter" minOccurs="0"/>
                <xs:element ref="comment" maxOccurs="unbounded"
                                minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="elnot" type="xs:string"/>
```

```
  </xs:complexType>
<xs:complexType name="parameter">
        <xs:sequence>
                <xs:element ref="commentReference" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="treeNumber" type="xs:string"/>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="suffixCode" type="xs:string"/>
        <xs:attribute name="sourceType" type="sourceType"/>
</xs:complexType>
….
<xs:complexType name="packageParameter">
        <xs:complexContent>
        <xs:extension base="parameter">
        <xs:sequence>
                <xs:element ref="parameter" maxOccurs="unbounded"
                        minOccurs="0"/>
        </xs:sequence>
                <xs:attribute name="packageStatus" type="packageStatus"/>
                </xs:extension>
        </xs:complexContent>
  </xs:complexType>
```

Bu dokümanın kullanılması için oluşturulan sınıflar aşağıda (Şekil 8.2) verilmiştir. Sınıflardan oluşturulan nesneler kullanılarak tekrar XML dokümanı oluşturmak veya XML formatındaki dosyalardan nesneler oluşturmak için tanımlanan sınıflar *Serializable* arayüzünü gerçeklemişlerdir.



Şekil 8.2. XSD dokümanlarını kullanmak için oluşturulan sınıflar

Gerekli tasarım ve geliştirme yapıldıktan sonra EWIRDB veritabanı dosyaları XML dosyalarına dönüştürülmüş ve bu dosyalar veritabanına eklenmiştir.

Oluşturulan XML dokümanlarında bazı bilgiler eleman özelliği, bazı bilgiler çocuk eleman olarak tanımlanmıştır. Hangi bilgilerin eleman özelliği olacağı değerin birden fazla değer alıp alamayacağına göre karar verilmiştir. Örneğin, hiyerarşi numaraları her bir eleman için tek olduğundan bunun özellik olmasına karar verilmiştir. XML dokümanının içindeki elemanların tipini belirlemek için XSD şemasında tanımlı olan etiketler kullanılmıştır.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<emitter xmlns="http://www.gazi.edu.tr/bm/24290291/thesis/v1.0"
        elnot="A555T">
   <description> bm24290291 thesis</description>
   <packageParameter treeNumber="000000000001.00"
        sourceType="MIXED" name="P/CW TREE">
```

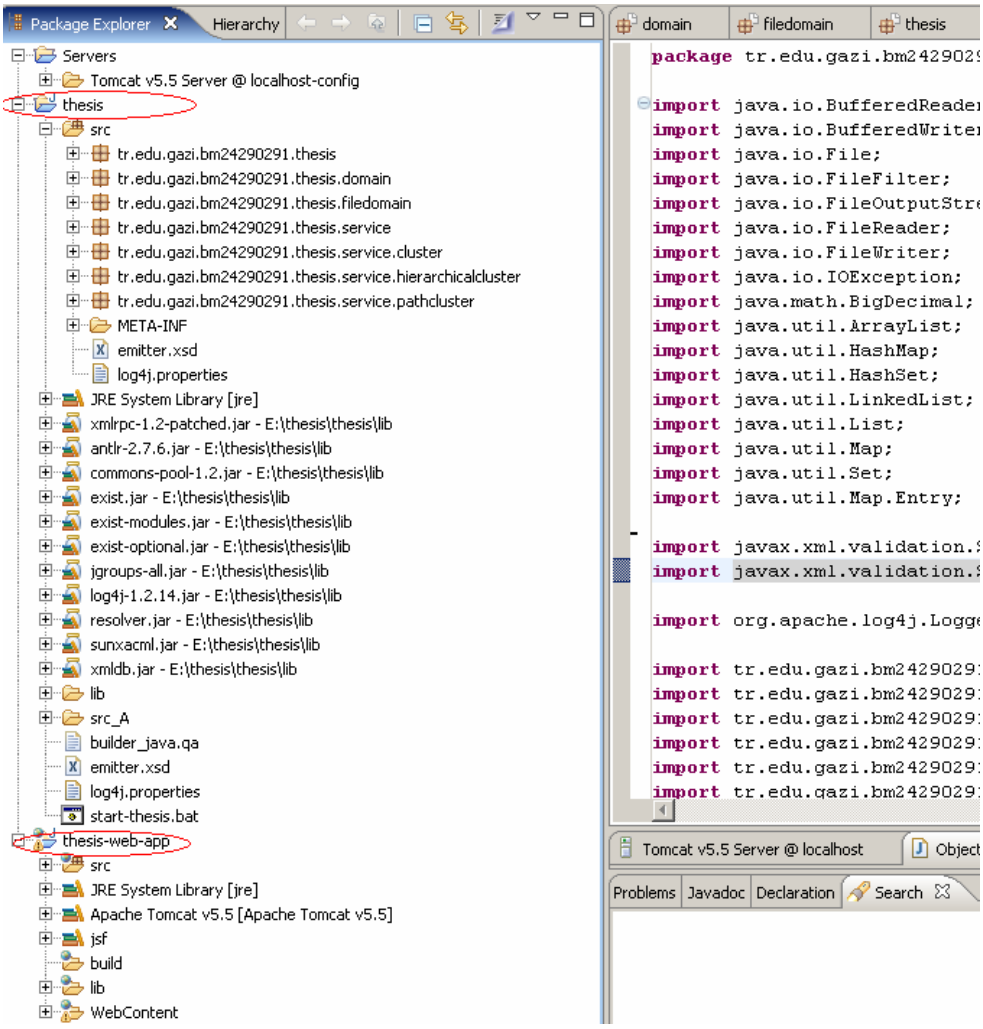
```
<parameter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="packageParameter" treeNumber="000000000011.00"
    sourceType="MIXED" name="SIGNAL POWER">
    parameter xsi:type="packageParameter" treeNumber="000000000111.00"
    suffixCode="++" sourceType="MIXED"
    name="CONSTANT POWER">
    <parameter xsi:type="numericParameter"
    treeNumber="000000000111.10" suffixCode="++"
    sourceType="OBSERVED"
    name="PEAK POWER (EFFECTIVE RADIATED)">
            <commentReference>C002</commentReference>
            <maxValue>1194.6</maxValue>
            <minValue>1294.6</minValue>
            <resolution>1.0</resolution>
            <resolutionUnit>DB</resolutionUnit>
            <unit>DBW</unit>
        </parameter>
...
</emitter >
```

8.2. Örnek Uygulamanın Hazırlanması

Örnek uygulamanın geliştirilmesinde, *Eclipse* geliştirme ortamı kullanılmıştır. Bu geliştirme ortamında EWIRDB veritabanı yapısını XML veritabanına taşımak ve XML veritabanını yönetmek için bir proje ve bu projenin çalıştırılmasından sonra sunulan servisi kullanacak örnek bir internet uygulaması projesi olmak üzere iki ayrı proje oluşturulmuştur. İnternet uygulaması projesinde gerekli kullanıcı arayüzleri tanımlanmıştır. Bu kullanıcı arayüzleri XML veritabanını yöneten servise bağlanarak işlemlerini gerçekleştirmiştir.

Bu projelerin geliştirilmesinde *Java* programlama dili kullanılmıştır. İnternet sunucusu olarak *Apache Tomcat 5.5* sunucusu kullanılmıştır. Projelerin birbirinden

bağımsız olarak ve aynı anda yönetilebilmesi konusunda kullanılan geliştirme ortamı çok yardımcı olmuştur. Aşağıda (Şekil 8.3) geliştirme ortamının yapısı gösterilmiştir.

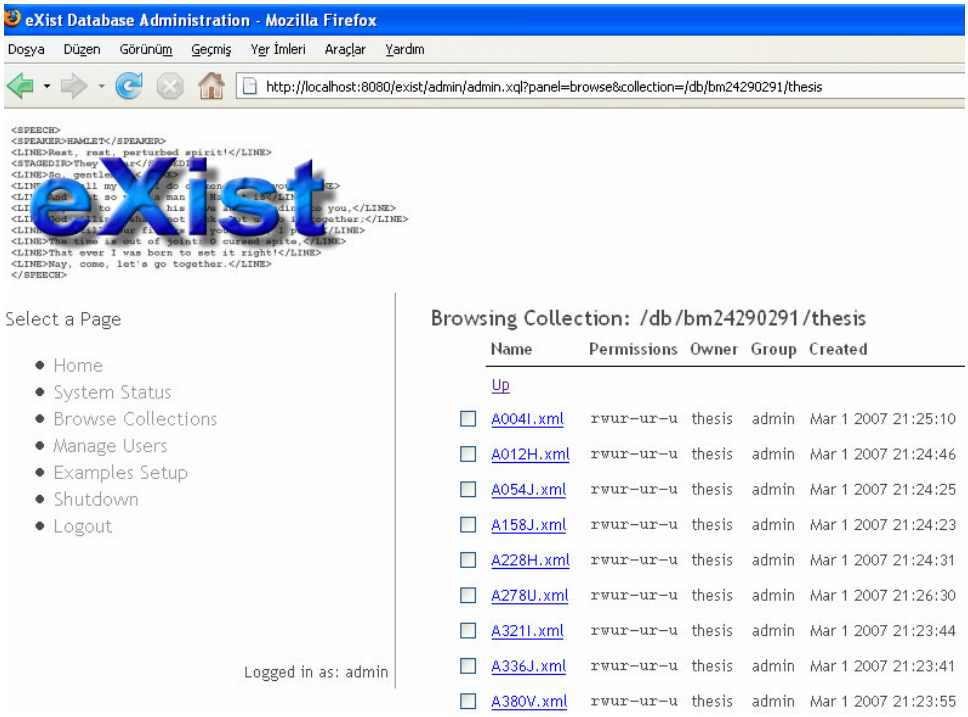


Şekil 8.3. Örnek uygulamanın geliştirilme ortamı

Geliştirme ortamında XML veritabanı olarak eXist Veritabanı [12] kullanılmıştır. Bu veritabanını yönetebilmek için gerekli kütüphaneler projeye eklenmiştir. Ayrıca, İnternet uygulaması için internet sunucusu kütüphaneleri ve geliştirme altyapısı olarak kullanılan JSF kütüphaneleri projelere eklenmiştir.

eXist Veritabanı [12], geliştirme ortamına gömülü olarak veya internet sunucusu üzerinden yönetilebilmektedir. Bu çalışmada internet sunucu üzerinden erişim ile veritabanının yönetilmesi tercih edilmiştir. Bu sayede veritabanı, bağımsız olarak çalışmış ve veritabanını kullanacak yazılımların veritabanına dışardan erişebilme

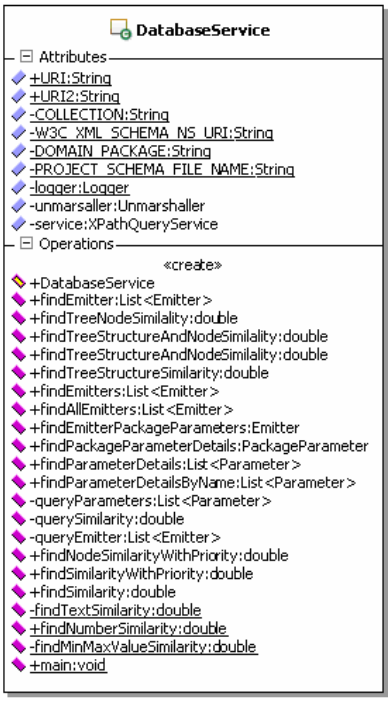
imkanı sağlanmıştır. Birden fazla yazılımın veritabanını kullanabilmesi şekilde sağlanmıştır. Veritabanında XML dokümanları */db/bm24290291/thesis* koleksiyonu altında toplanmıştır. Veritabanı, yazılımla yönetilebilmesinin yanında internet tarayıcısı ile de yönetilebilmektedir. Aşağıda (Şekil 8.4) veritabanının internet tarayıcısı üzerinden yönetilmesi için kullanılan arayüz verilmiştir.



Şekil 8.4. XML veritabanının internet tarayıcısı üzerinden yönetilmesi

Veritabanında saklanması için gerekli altyapı ve tasarım tamamlandıktan sonra, sırasıyla veritabanının kullanılması için gerekli olan servislerin geliştirilmesine ve bu servisleri kullanan örnek kullanıcı arayüzü oluşturulmasına geçilmiştir.

Veritabanında bulunan verilerin yönetilmesi için gerekli servislerin tanımlanmasında XQuery dili kullanılmıştır. İstemcilerden alınan istekler, XQuery diline dönüştürülerek veritabanında sorgulanmakta ve istek tekrar istemcilere gönderilmektedir. eXist veritabanında, XQuery sorgularını gerçekleştirebilmek için önceden tanımlanmış olan *XPathQueryService* adında bir sınıf kullanılmaktadır. Bu sınıf hem XPath hem de XQuery dili ile hazırlanan sorgulara cevap verebilmektedir.

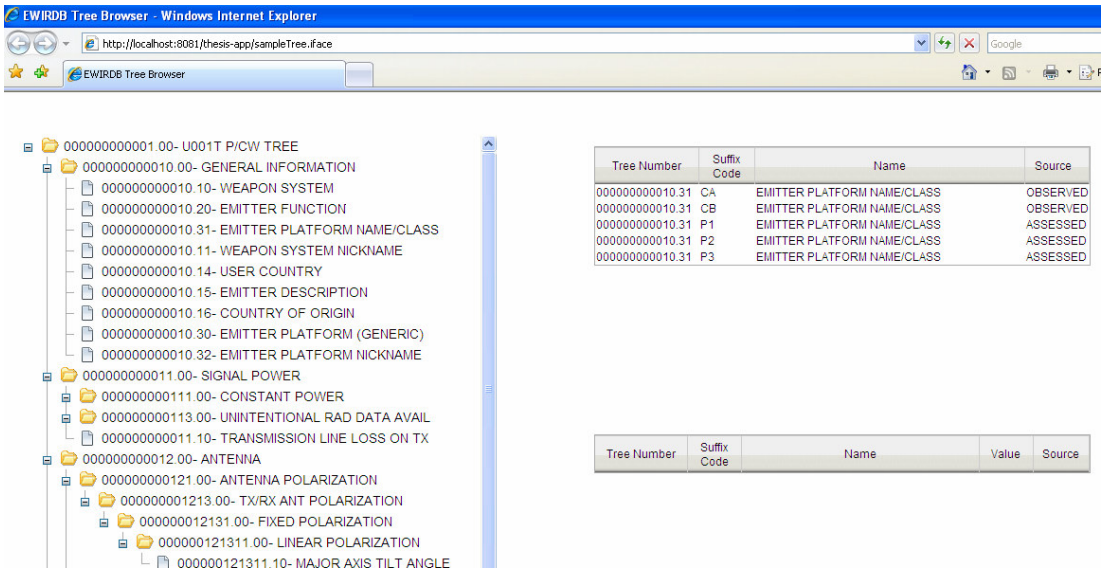


Şekil 8.5. EWIRDB veritabanını sorgulamakta kullanılan örnek servis sınıfı

Bu sınıf ile bir verinin tamamının istenebileceği gibi verinin belirli bir hiyerarşisi üzerinde de işlem yapılabilmektedir. Bu sınıf, yapılan isteklerin sonuçlarını XML formatına dönüştürebileceği gibi JAXB altyapısı ile XML dosyalarının içeriğini de nesnelere dönüştürülebilmektedir. Bu sayede veriler iki farklı formatta gösterilebilmektedir.

Geliştirilen örnek internet uygulaması, tanımlanan bu servisleri kullanarak verilerin sorgulanmasını ve görüntülenmesini sağlamaktadır. İnternet uygulamasının geliştirilmesinde JSF ve AJAX altyapısı birlikte kullanılmıştır. İnternet tarayıcısında görüntülenecek veriler kullanıcı isteği olduğunda sayfa yenilenmeden sunucuya bağlanarak yeni istekte bulunmaktadır. Bu sayede kullanıcı, verilerin sunucudan getirildiğini fark etmemekte ve tüm verinin yüklenebilmesi için beklemek zorunda kalmamaktadır. Aşağıda (Şekil 8.6) örnek uygulamanın ekran çıktısı verilmiştir. Seçilen bir verinin görüntülenmesi için kullanılan bu ekranda hiyerarşik yapı ağaç şeklinde solda görüntülenmektedir. Kullanıcı hiyerarşide bulunan bir elemanı seçtiğinde eğer bu elemanın alt hiyerarşisi yüklenmedi ise sunucudan bu veriler getirilmektedir. Sağ üst köşede bulunan tabloda seçilen elemandan birden fazla tekrar

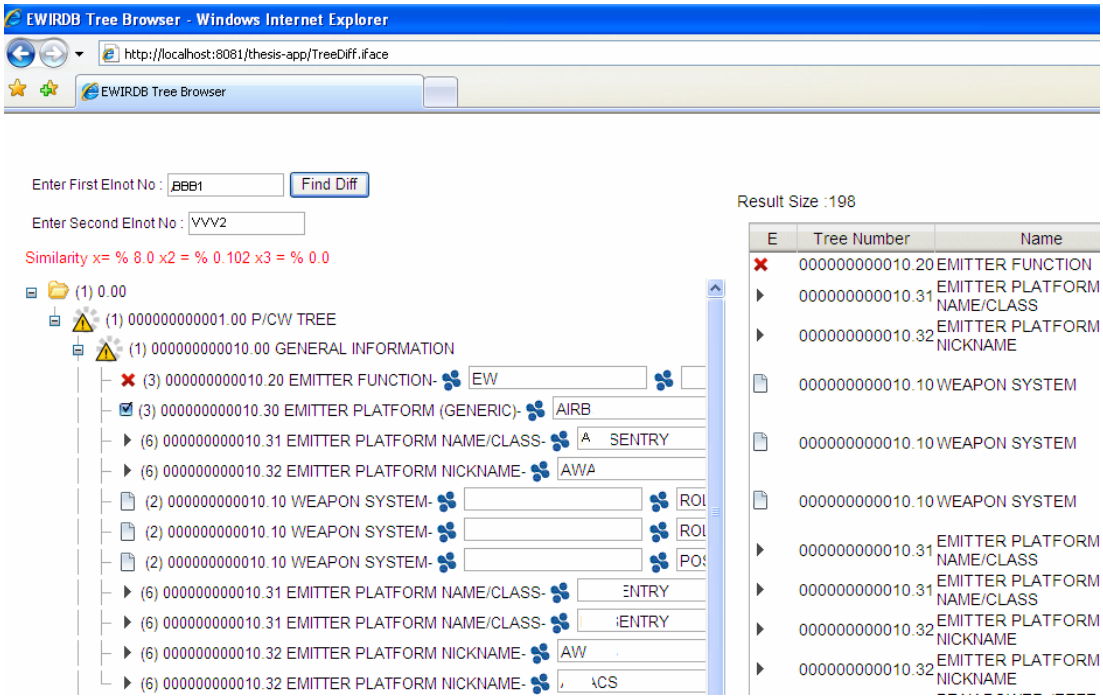
varsa tekrarlayan elemanların hepsini görüntülemektedir. Bu işlem için yine AJAX alt yapısı kullanılmıştır. Seçilen elemanın tekrarlı olan verileri, ekran yenilenmeden hemen gösterilmektedir.



Şekil 8.6. İnternet tarayıcısı üzerinden EWIRDB verilerinin sunulması

Tekrarlanan elemanların gösterimi için kullanılan tabloda elemanların ayırt edici özelliği *Suffix Code* sütunudur. Bu alan yardımı ile tekrarlı kayıtlar birbirinden ayırt edilebilmektedir. Ayrıca bu tablodan seçilen kayıtların detayları ekranın sağ alt köşesinde bulunan tabloda görüntülenmektedir. Kullanıcı bu arayüz sayesinde seçtiği bir veriyi ve verinin hiyerarşisini görebilmektedir.

Uygulama yazılımı ile ayrıca iki hiyerarşik verinin benzerliği yine ağaç yapısı olarak görüntülenmektedir. Seçilen iki veri arasındaki farklara göre ağacın elemanlarının simgeleri değişik olarak gösterilerek farkların ve benzerliklerin daha kolay anlaşılması sağlanmaktadır. Benzerliklerin gösterimi için kullanılan örnek ekran aşağıda (Şekil 8.7) verilmiştir.



Şekil 8.7. İki hiyerarşik verinin benzerliklerini gösteren kullanıcı arayüzü

Bu ekranın sol tarafında iki hiyerarşik verinin gösterimi ve farklı olarak değerleri görüntülenirken sağ tarafta ise farklar tablosal olarak gösterilmektedir. Kullanıcı arayüzü ile iki ağaç arasındaki fark detaylı olarak gösterilmekte ve kullanıcının farkları anlamasında yardımcı olmaktadır. Farkların belirtilmesinde aşağıda belirtilen gösterimler kullanılmaktadır:

– Bir eleman karşılaştırılan ilk veride var diğerinde yoksa ✖ işareti

– Bir eleman karşılaştırılan ilk veride yok diğerinde varsa 🗋 işareti

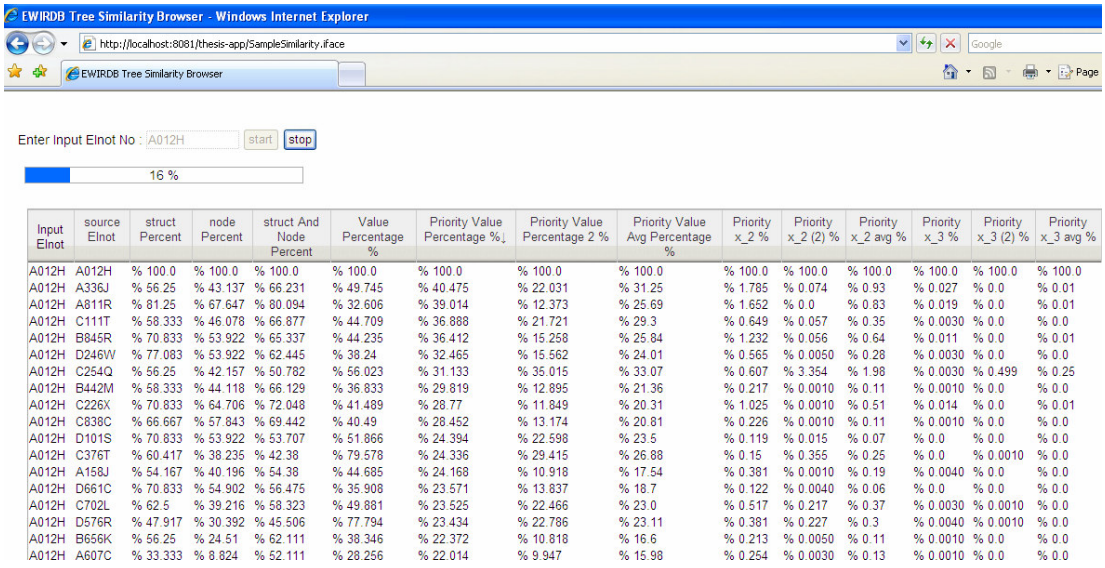
– Bir eleman her iki veride var fakat değerleri farklı ise ▸ işareti

– Bir eleman her iki veride de var ve değerleri de aynı ise ☑ işareti

– Hiyerarşik veriler içerisinde bilgi saklamayan sadece verilerin organizasyonunda rol alan elemanların altındaki bütün veriler aynı yapıda ve aynı değerlere sahip değilse ⚠ işareti kullanılır.

Uygulama yazılımı ayrıca bir hiyerarşik verinin veritabanında bulunup bulunmadığını veya veritabanında bu veriye benzer verilerin bulunup bulunmadığını

görüntüleyen bir kullanıcı arayüzüne (Şekil 8.8) sahiptir. Bu arayüz metotlarla elde edilen benzerlikleri yüzde olarak görüntülemektedir.



Input Elnot	source Elnot	struct Percent	node Percent	struct And Node Percent	Value Percentage %	Priority Value Percentage %↓	Priority Value Percentage 2 %	Priority Value Avg Percentage %	Priority x_2 %	Priority x_2 (2) %	Priority x_2 avg %	Priority x_3 %	Priority x_3 (2) %	Priority x_3 avg %
A012H	A012H	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0	% 100.0
A012H	A336J	% 56.25	% 43.137	% 66.231	% 49.745	% 40.475	% 22.031	% 31.25	% 1.785	% 0.074	% 0.93	% 0.027	% 0.0	% 0.01
A012H	A811R	% 81.25	% 67.647	% 80.094	% 32.606	% 39.014	% 12.373	% 25.69	% 1.652	% 0.0	% 0.83	% 0.019	% 0.0	% 0.01
A012H	C111T	% 58.333	% 46.078	% 66.877	% 44.709	% 36.888	% 21.721	% 29.3	% 0.649	% 0.057	% 0.35	% 0.0030	% 0.0	% 0.0
A012H	B845R	% 70.833	% 53.922	% 65.337	% 44.235	% 36.412	% 15.258	% 25.84	% 1.232	% 0.056	% 0.64	% 0.011	% 0.0	% 0.01
A012H	D246W	% 77.083	% 53.922	% 62.445	% 38.24	% 32.465	% 15.562	% 24.01	% 0.565	% 0.0050	% 0.28	% 0.0030	% 0.0	% 0.0
A012H	C254Q	% 56.25	% 42.157	% 50.782	% 56.023	% 31.133	% 35.015	% 33.07	% 0.607	% 3.354	% 1.98	% 0.0030	% 0.499	% 0.25
A012H	B442M	% 58.333	% 44.118	% 66.129	% 36.833	% 29.819	% 12.895	% 21.36	% 0.217	% 0.0010	% 0.11	% 0.0010	% 0.0	% 0.0
A012H	C226X	% 70.833	% 64.706	% 72.048	% 41.489	% 28.77	% 11.849	% 20.31	% 1.025	% 0.0010	% 0.51	% 0.014	% 0.0	% 0.01
A012H	C838C	% 66.667	% 57.843	% 69.442	% 40.49	% 28.452	% 13.174	% 20.81	% 0.226	% 0.0010	% 0.11	% 0.0010	% 0.0	% 0.0
A012H	D101S	% 70.833	% 53.922	% 53.707	% 51.866	% 24.394	% 22.598	% 23.5	% 0.119	% 0.015	% 0.07	% 0.0	% 0.0	% 0.0
A012H	C376T	% 60.417	% 38.235	% 42.38	% 79.578	% 24.336	% 29.415	% 26.88	% 0.15	% 0.355	% 0.25	% 0.0	% 0.0010	% 0.0
A012H	A158J	% 54.167	% 40.196	% 54.38	% 44.685	% 24.168	% 10.918	% 17.54	% 0.381	% 0.0010	% 0.19	% 0.0040	% 0.0	% 0.0
A012H	D661C	% 70.833	% 54.902	% 56.475	% 35.908	% 23.571	% 13.837	% 18.7	% 0.122	% 0.0040	% 0.06	% 0.0	% 0.0	% 0.0
A012H	C702L	% 62.5	% 39.216	% 58.323	% 49.881	% 23.525	% 22.466	% 23.0	% 0.517	% 0.217	% 0.37	% 0.0030	% 0.0010	% 0.0
A012H	D576R	% 47.917	% 30.392	% 45.506	% 77.794	% 23.434	% 22.786	% 23.11	% 0.381	% 0.227	% 0.3	% 0.0040	% 0.0010	% 0.0
A012H	B656K	% 56.25	% 24.51	% 62.111	% 38.346	% 22.372	% 10.818	% 16.6	% 0.213	% 0.0050	% 0.11	% 0.0010	% 0.0	% 0.0
A012H	A607C	% 33.333	% 8.824	% 52.111	% 28.256	% 22.014	% 9.947	% 15.98	% 0.254	% 0.0030	% 0.13	% 0.0010	% 0.0	% 0.0

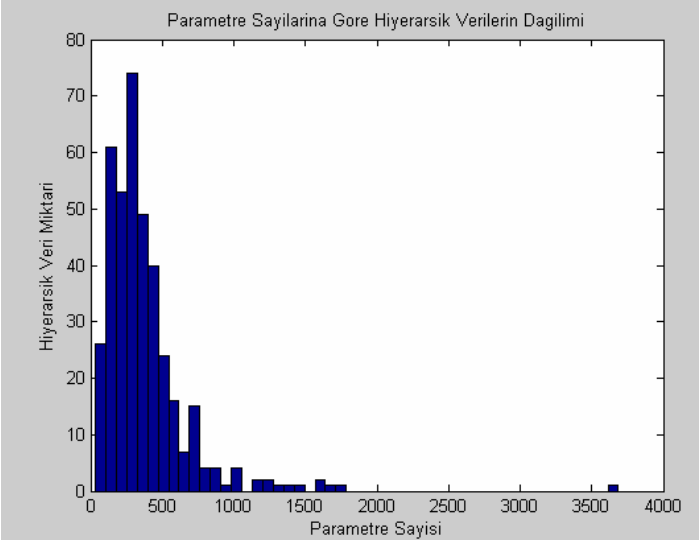
Şekil 8.8. Bir hiyerarşik verinin veritabanında bulunan hiyerarşik veriler ile benzerliklerinin gösterimi

Bu ekranda bulunan tablo sütunları seçilerek, seçilen sütuna göre benzerlikler küçükten-büyüğe veya büyükten-küçüğe sıralanabilmektedir. Bu arayüz ile kullanıcının bir verinin benzerliğini, değişik metotlarla karşılaştırmalı olarak görebilmesine ve benzerlikleri yorumlayarak sonuç çıkarmasına yardımcı olunmaktadır. Ekrandaki birinci sütunda benzerliği aranan verinin adı, ikinci sütunda veritabanında bulunan verinin adı gösterilmektedir. Benzerlik bulunurken sadece ağacın dallarını dikkate alındığında elde edilen sonuç üçüncü sütunda, ağacın yapraklarını dikkate alan yöntem kullanıldığında elde edilen sonuç dördüncü sütunda, ağacın dalları ve yapraklarını dikkate alan yöntem kullanıldığında elde edilen sonuç beşinci sütunda verilmiştir. Yapısal ve içerik olarak benzerliği dikkate alan yöntem uygulandığında elde edilen sonuç altıncı sütunda, yapısal ve içerik bilgilerini ve önceliklerini dikkate alınması sonucu elde edilen sonuç (birinci sütundaki ağacın ikinci sütundaki ağaca benzerliği) yedinci sütunda verilmiştir. İkinci sütunda verilen ağacın ilk sütundakine benzerliği sekizinci sütunda verilmiştir. Simetrik benzerlik kullanmak istendiğinde yedi ve sekizinci sütunlardaki benzerliklerin ortalaması bir sonraki sütunda verilmiştir. Diğer sütunlarda benzerlik

bulurken ağaçların elemanlarının karşılaştırılmasında elde edilen benzerliklerin kareleri ve küpü alınarak elde edilen sonuçlar verilmiştir.

8.3. Hiyerarşik Verilerin Gruplandırılması

Çalışmada kullanılan EWIRDB verilerinde en az 33 ve en çok 3386 parametre bulunmaktadır. Verilerin parametre sayılarına göre dağılımı Şekil 8.9'de verilmiştir. 390 örnek veride ortalama 377 parametre yer almaktadır. EWIRDB verilerinin veritabanında XML formatında saklanmasının ardından verilerin benzerliklerine göre gruplandırılması için çalışma yapılmıştır. Bu çalışmada veriler sırası ile seviyelerine göre, yapısal / içerik bilgilerine göre ve güncelleme mesafesi bilgilerine göre gruplandırılmıştır.



Şekil 8.9. Parametre sayılarına göre hiyerarşik verilerin dağılımı

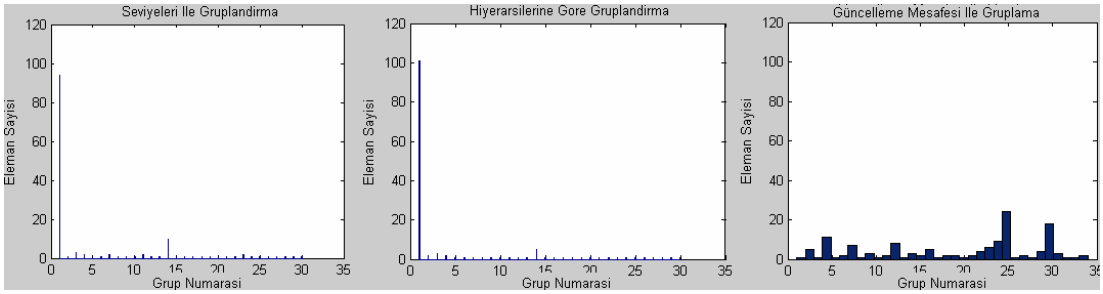
Gruplandırma yapılabilmesi için iki ağaç arasındaki farkın nasıl hesaplanacağı ve her bir ağacın birbirlerine olan mesafenin belirlenmesi gerekmiştir. İki ağaç arasındaki farkın hesaplanmasında yapısal benzerlik hesaplanması kullanılmıştır. Benzerlik hesaplamaları 0-1 arasında değerlerden oluşmaktadır. İki ağaç arasındaki benzerlik bulunduktan sonra 1 değerinden çıkarılarak birbirine benzememe (mesafe) değeri elde edilmiştir. Bu bilgi kullanılarak her bir ağacın diğerlerine olan mesafesi hesaplanmış ve bu mesafe bilgileri kullanılarak gruplandırma işlemi yapılmıştır. İki ağacın yapısal benzerliğinin bulunmasında:

– İki ağacın elemanlarının seviyelerine göre benzerlik hesaplaması [44],

– İki ağacın hiyerarşi bilgileri kullanılarak benzerlik hesaplaması [32],

– Bir ağacı diğer ağaca benzetebilmek için güncelleme mesafesinin bulunması ile benzerliğin hesaplanması [26] yöntemleri kullanılmıştır.

İki ağacın elemanlarının seviyelerine göre ve iki ağacın hiyerarşi bilgilerine göre elde edilen benzerlik bilgileri hiyerarşik gruplandırma algoritması ile gruplandırılmıştır. Elde edilen gruplarda elemanların dağılımı dengeli olmamıştır. Güncelleme mesafesi kullanılarak elde edilen bilgiler, [44]'de belirtilen gruplama metodundan adapte edilen metotla sağlanmıştır. Gruplarda bulunan elemanların dağılımı, diğer iki gruplama metodu ile elde edilen sonuçlardan daha iyi olmuştur. Aşağıda her üç yöntemle oluşturulan grupların elemanlarının gruplara göre dağılım grafiği verilmiştir.



Şekil 8.10. Seviye, Hiyerarşi ve Güncelleme mesafesi benzerlikleri kullanılarak elde edilen grupların elemanlarının dağılımı

Seviye ve hiyerarşi bilgileri kullanılarak, hiyerarşik gruplama algoritması ile gruplandırılmasında elemanların dağılımı düzenli olmamıştır. Seviyelerine göre benzerlik bilgileri kullanıldığında 90'nın üzerindeki ağaç, hiyerarşik bilgilerine göre benzerlik kullanıldığında 100 ağaç aynı grubun içinde bulunmuştur. Diğer gruplarda bulunan eleman sayıları da 10'nun altında kalmıştır. [44]'ten adapte edilen metot kullanıldığında, eleman dağılımları daha dengeli olmuştur. Bu çalışmada sunulan gruplama metodu aşağıda verilmiştir:

1. Tüm veriler arasından benzerliği en çok olan iki veri bulunur.

2. İki verinin benzerliği eşik değerinden fazla ise bu iki veri [44]'de belirtilen şekilde birleştirilerek bir grup oluşturulur. İki veri kümeden çıkarılarak yerine yeni oluşturulan grup eklenir.
3. Hiçbir gruba dahil olmayan eleman varsa 1. adıma dönülür.
4. Tüm veriler gruplandırıldıktan sonra gruplar içindeki veriler diğer gruplarla karşılaştırılır.
5. Eğer kendi grubundan daha yüksek benzerlik elde edilen bir grup var ise veri kendi grubundan çıkarılarak diğer gruba dahil edilir.
6. Tüm veriler için 5. adım gerçekleştirildi ve grup değişikliği yapıldı ise 4. adıma dönülür.
7. Tüm veriler için 5. adım gerçekleştirildi ve grup değişikliği yapılmadı ise çıkılır.

Bu gruplama metoduyla oluşturulan her bir grup iki ağacın birleşiminden oluşan ağaçla ifade edilmiştir. Bu sayede, bir veri bir grupla karşılaştırılırken aynı benzerlik metodu kullanılmıştır.

8.4. Hiyerarşik Verilerin Benzerliklerinin Bulunması

Oluşturulan gruplar kullanılarak benzerliklerin bulunmasında aşağıda belirtilen yöntem kullanılmıştır:

1. Her bir grup üzerinde gezilerek girdi ağaç ile grup ağacı arasındaki benzerlik güncelleme mesafesi kullanılarak bulunur.
2. Benzerlik belirli eşik değeri üzerinde olan gruplarda bulunan bütün verilerle girdi ağacı karşılaştırılır.
3. Benzerlik belirli bir eşik değeri üzerinde ise bu ağaç benzerlik değeri ile birlikte sorgu sonucuna eklenir.

Kullanılan yöntemde, her bir grup ile girdi ağacı arasındaki yapısal benzerlik bulunur. Bu yapısal benzerlik belirli eşik değeri üzerinde ise grupta girdiye benzeyen ağaçların bulunduğu kabul edilir ve grubun her bir elemanı ile içerik ve yapısal olarak benzerlikler hesaplanır. Yapısal ve içerik olarak benzerliklerin bulunmasında [15]'ten adapte edilen benzerlik bulma yöntemi kullanılmıştır. Some ve arkadaşlarının çalışmasında belirtilen yöntemde, ağacın elemanlarının ağırlıkları

dikkate alınmamıştır. Bu çalışmada ise, elemanların benzerlikleri ile elde edilen sonuçlarda elemanların ağırlıkları dikkate alınmıştır. EWIRDB veritabanında bulunan her bir verinin 1-6 arasında ağırlıkları bulunmaktadır. 1 en ağırlıklı, 6 en az ağırlıklı elemanlar için kullanılmıştır. Ayrıca [15]'de belirtilen yöntemde, benzerliklerin aritmetik ortalamaları alınarak benzerlik hesaplaması yapılmıştır. Bu çalışmada ise, benzer verilerin öne çıkması ve benzerliği az olan verilerin daha arkada kalması için her bir elemanın ağırlığı hesaplanırken kullanılan fonksiyon değiştirilmiştir. [15]'de belirtilen eleman benzerliği *benzerlik(x)* (Bkz. Eş. 7.7) olarak tanımlanmışken bu çalışmada (*1 – cos(benzerlik(x)))* olarak tanımlanmıştır. Bu çalışmada, tanımlanan gruplama metodu kullanılarak benzerlik bulunurken iki değişken tanımlanmıştır. Bu değişkenler:

- Grupları gezerken elde edilen yapısal benzerliğin eşik değeri,
- Sorgu sonucu olarak dönebilmesi için ağaçların yapısal ve içerik olarak benzerliğinin eşik değeridir.

EWIRDB veritabanında bulunan verilerin ağaç yapılarından beşer kopya oluşturulmuştur. Oluşturulan her bir kopyanın adı gerçek ağacın adına benzer şekilde tanımlanmıştır. Kopya ağaçların isimlendirilmesinde gerçek ağacın adına "_X" şeklinde bir ekleme yapılmıştır (X 1 ile 5 arasında bir sayıdır. Örneğin A322B ağacı için A322B_1, A322B_2, A322B_3, A322B_4 ve A322B_5 kopya ağaçlar üretilmiştir). ve her bir kopya veri üzerinde değişiklikler yapılarak sorguda kullanılacak örnek ağaçlar elde edilmiştir. Bu eşik değerlerinin aldığı değerlere göre yapısal [44], yapısal/içerik [15] ve bu çalışmada önerilen benzerlik metotlarının etkinliği incelenmiştir. Bu ağaçların benzerleri, yukarıda anlatılan yöntemlerle sorgulanmış ve sonuçların doğruluğu kontrol edilmiştir. Bu üç yöntem için ağaçların büyüklüklerine göre performans ölçümleri yapılmıştır (Çizelge 8.1 ve Şekil 8.11).

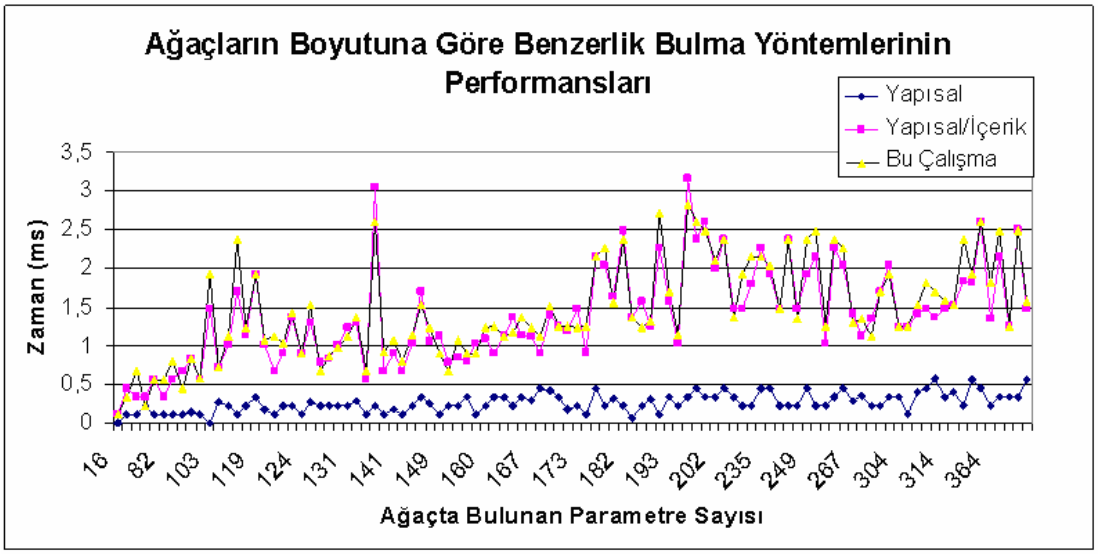
Çizelge 8.1. Benzerlik bulma metotlarının ortalama performansları

Benzerlik Bulma Yöntemi	Ortalama Zaman (ms)
Yapısal	0,262966184
Yapısal/İçerik	1,380664251
Tez	1,464652174

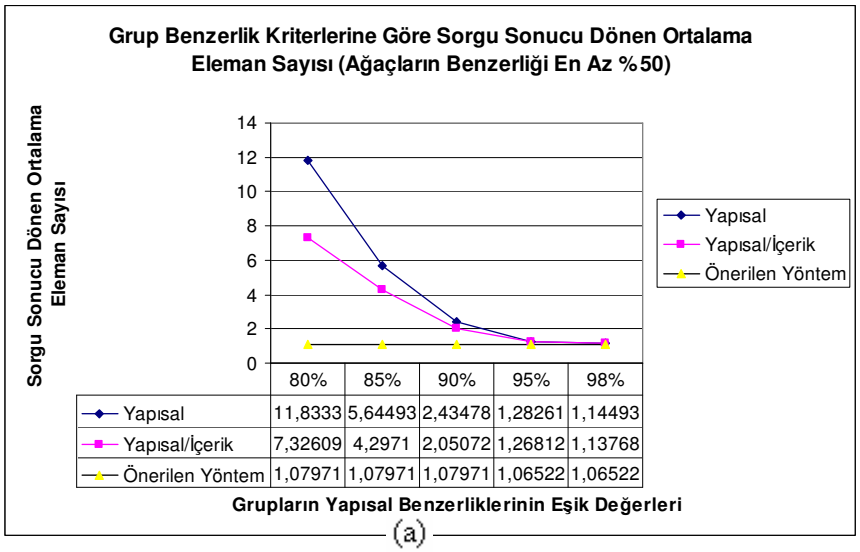
Yapısal özellik kullanılarak benzerlik bulma yöntemi daha performanslıdır (Şekil 8.11). Fakat bu yöntemde ağacın değerleri göz ardı edildiği için sonuçlarda hatalı verilerin bulunma olasılığı artmaktadır.

Yapısal ve içerik kontrolü yapan yöntemde [15], performans diğerine göre beş kat küçüktür. Bu yöntemde, verilerin içeriklerinin kontrolünün yapılması performans kaybına sebep olmaktadır.

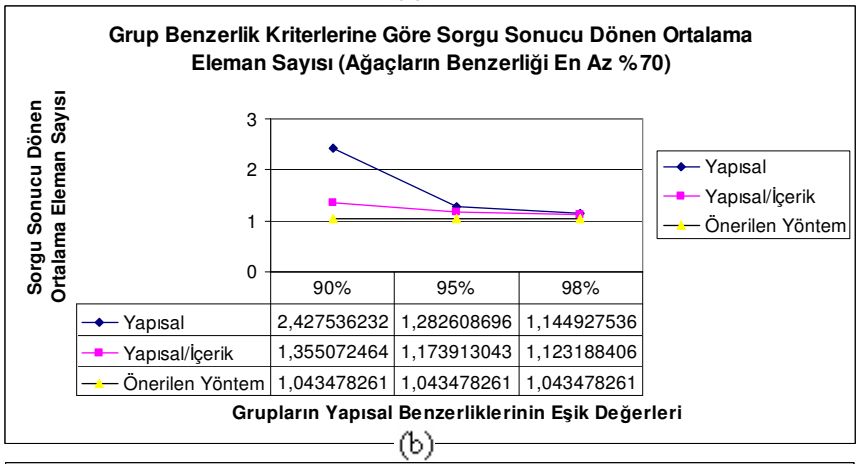
Bu çalışmada önerilen yöntem ile [15]'de önerilen yöntemin performansında çok büyük değişiklik bulunmamaktadır. Ortalamalar göz önüne alındığında, ortalama performans her üç yöntem için 2 mili saniyenin altındadır. Bu çalışmada kullanılan yöntem ek performans kaybına sebep olmamıştır.



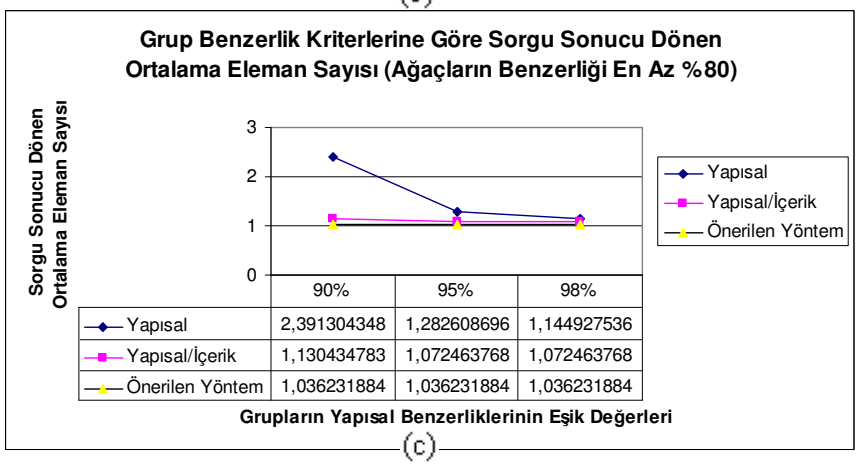
Şekil 8.11. Ağacın boyutlarına göre benzerlik bulma yöntemlerinin performansları



(a)



(b)



(c)

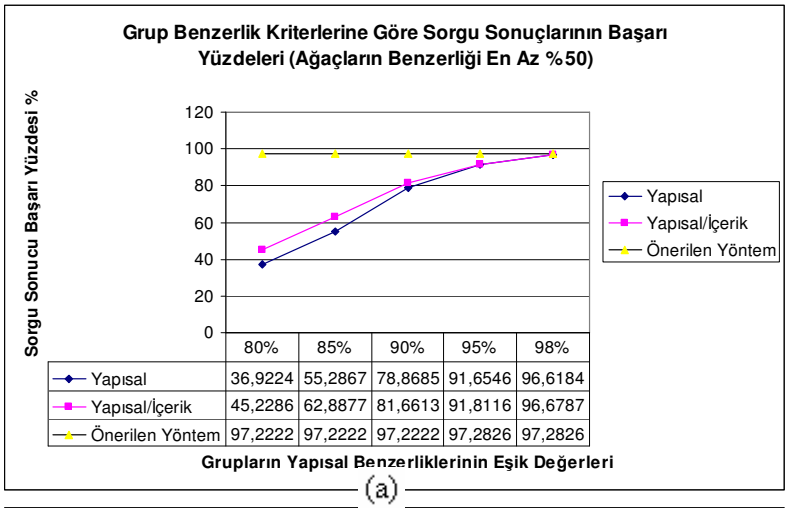
Şekil 8.12. Grup yapısal benzerlik kriterlerine göre sorgu sonucu dönen eleman sayıları

a. Benzerlik değeri en az %50 iken sorgu sonucu dönen eleman sayıları
b. Benzerlik değeri en az %70 iken sorgu sonucu dönen eleman sayıları
c. Benzerlik değeri en az %80 iken sorgu sonucu dönen eleman sayıları

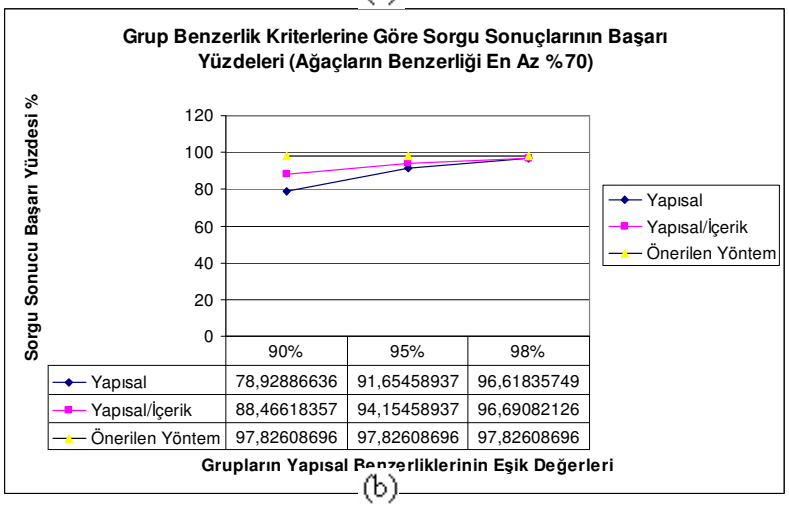
Sorgu sonuçlarında kaç değerin döndüğü ve bu dönen kayıtların ortalama doğruluk yüzdeleri bulunmuştur. Şekil 8.12'de değişik eşik değerlerine göre sorgu sonucunda dönen kayıtların ortalama sayıları verilmiştir. Bu işlemler bir ağaç ile bir grubun yapısal benzerliğinde kullanılan eşik değerleri ve bir ağacın diğer ağaçla benzerliğinde kullanılan eşik değerleri için ayrı ayrı yapılmıştır. Öncelikle veri benzerliğinin eşik değeri %50 iken grup benzerlik eşik değerlerinin %80, %85, %90, %95, %98 olduğu durumlar incelenmiştir. Daha sonra, veri benzerlik eşik değerlerinin %70 ve %80 olduğu durumlar incelenmiştir.

Her üç sonuçta da en az sorgu sonucu bu çalışmada tanımlanan yöntem kullanılarak elde edilmiştir. Yapısal benzerliklerle elde edilen elemanlar grupların yapısal benzerliklerinde kullanılan eşik değer azaldıkça artmakta ve yanlış sonuçlar oluşturmaya başlamaktadır. Yapısal ve içerik benzerliği kullanılarak elde edilen eleman sayısı yapısal benzerlik yöntemine göre daha iyi sonuçlar vermektedir. Bu çalışmada tanımlanan yöntemle elde edilen eleman sayısı, yapısal benzerlik eşik değeri küçülse de bundan etkilenmemektedir.

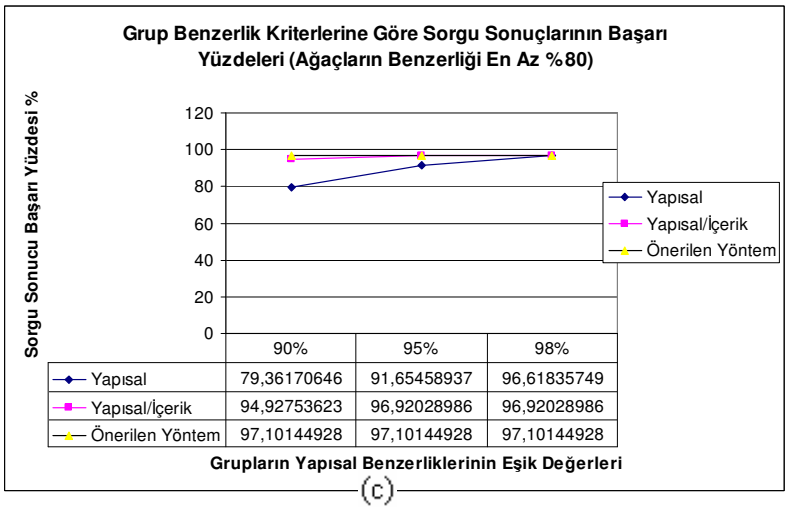
Elde edilen sonuçların başarılı olup olmadığı incelendiğinde bu çalışmada tanımlanan yöntemle elde edilen sonuçların başarısının %97'nin altına düşmediği görülmüştür (Şekil 8.13). Diğer yöntemlerde, grup yapısal benzerlik eşik değeri düştükçe doğruluk oranı düşmekte ve eşik değeri yükseldikçe başarı yüzdesi %97'e yaklaşmaktadır. Eşik değeri %90'nın altına düştüğünde yapısal benzerlikle elde edilen sonuçlar ancak %80'nin altında başarı sağlamaktadır. İki ağacın benzerlik kriteri %50 olarak belirlendiğinde diğer iki yöntemle ile elde edilen başarı %50'nin altına düşerken, bu çalışmada belirtilen yöntemle elde edilen sonuçlar hala %97'nin üstündeki başarısını korumaktadır.



(a)



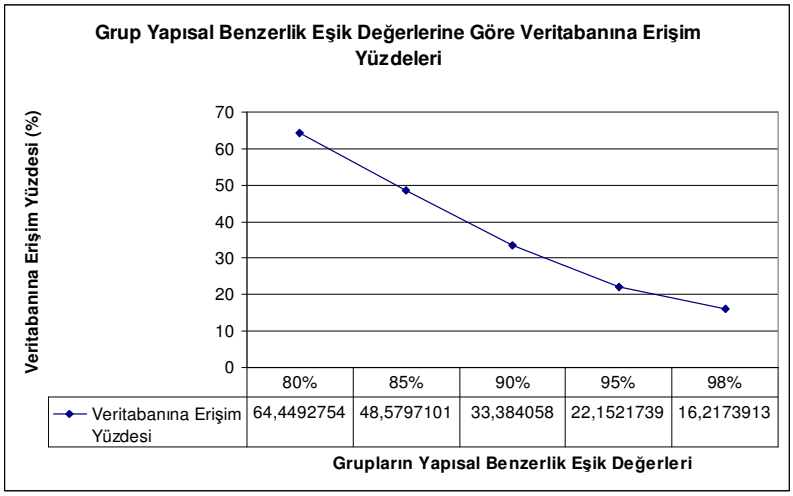
(b)



(c)

Şekil 8.13. Grup yapısal benzerlik kriterlerine göre sorgu sonucu başarı yüzdeleri
a. Ağaçların benzerliği en az %50 iken sorgu sonucu başarı yüzdeleri
b. Ağaçların benzerliği en az %70 iken sorgu sonucu başarı yüzdeleri
c. Ağaçların benzerliği en az %80 iken sorgu sonucu başarı yüzdeleri

Bu çalışmada belirtilen yöntemle elde edilen sonuçların gruplar üzerinde tanımlanan kısıtlara bağlı olmadığı görülmüştür. Grup yapısal benzerlik eşik değerinin %50 seçilmesi ile %98 seçilmesi arasında başarı açısından büyük bir fark bulunmamaktadır. Bu eşik değerinin küçültülüp büyütülmesi karşılaştırılacak olan ağaç sayısını etkilemektedir. Aşağıda (Şekil 8.14), eşik değerinin değişimine göre veritabanında bulunan verilerin yüzde kaçının kullanılacağı gösterilmiştir.



Şekil 8.14. Grup yapısal benzerlik kriterlerinin değerlerine göre veritabanına erişim yüzdeleri

Eşik değeri %80 seçildiğinde veritabanından yaklaşık %64 verinin benzerlikte kullanılması gerekirken, %98 seçildiğinde veritabanından %16 veri kullanılmaktadır. Veritabanına en az erişim ile doğru sonuçların elde edilme başarısı bu çalışmada %97'nin üzerinde olmuştur.

9. SONUÇLAR

EWIRDB, elektronik harp verilerinin saklandığı en büyük veri kaynağıdır. Bu veri kaynağının verileri dosyalar şeklinde saklanmakta ve yönetilmektedir. Daha önce yapılan çalışmalarda EWIRDB veritabanı ilişkisel ve nesne yönelimli veritabanı modelleri denenmiş, fakat veritabanının en çok %20'si modellenebilmiştir. Mevcut çalışmalarla elde edilen veritabanları üzerinde kısıtlı sorgular gerçekleştirilebilmiştir. EWRIDB veritabanının etkin yönetimi için veritabanı yönetim sistemleri alternatif olarak değerlendirilmiş ve veriler hiyerarşik yapısına uygun olan XML veritabanına taşınmıştır. Bu çalışma sonucunda, verilerin modellenmesinde ilişkisel veritabanı modelinin ve nesne yönelimli veritabanı modelinin etkin bir çözüm olmadığı görülmüştür. Bu çalışmada veritabanının tamamı modellenmiş, veritabanı üzerinde istenilen sorgunun çalıştırılabilmesi ve sorgu sonuçlarının istenilen formatta sunulması sağlanmıştır. Tez çalışmasında kullanılan veri yapıları tasnif dışıdır. Çalışma bu veri yapısı göz önünde bulundurularak oluşturulmuş veriler üzerinde yapılmıştır.

Veritabanında bulunan hiyerarşik verilerinin benzerliklerinin bulunmasında, verilerin hiyerarşik yapı ve içerik bilgileri ile birlikte elemanlarının öncelik değerleri kullanılmıştır. Ayrıca, benzer verilerin öne çıkması ve benzerliği da az olan verilerin daha arkada kalması için her bir elemanın ağırlığı hesaplanırken elde edilen sonucun kosinüsü alınarak 1 değerinden çıkarılmıştır *(1 – cos(benzerlik(x)))*.

Sunulan benzerlik yöntemi ile çok amaçlı kullanılabilecek bir alt yapı oluşturulmuştur. Bu yöntemle:

- İki radar verisi arasındaki benzerliğinin bulunması,
- Adı veya kaynağı bilinmeyen bir elektronik harp verisinin veritabanından sorgulanarak adının bulunması,
- Birbiri ile yakın parametrelere sahip olan verilerin veritabanına eklenmeden önce tespit edilmesi ve böylece veritabanında bulunan veriler arasında belirsizlik durumunun önlenmesi,

– Farklı zamanlarda elde edilen ve aynı kaynağa ait olan EWIRDB verilerinin incelenerek değişikliklerin tespit edilmesi sağlanmıştır.

Bu çalışmada ağaçların yapısal benzerlikleri dikkate alınarak gruplandırma yapılmıştır. Bu yöntem ayrıca:

– Ağaç güncelleme mesafesi sonucu elde edilen benzerlikleri kullanan,
– Benzerlik değerleri yüksek olanların aynı grupta yer almasını sağlayan,
– Grupları ağaç şeklinde ifade edebilen,
– Grupların da benzerliklerini hesaplayarak hiyerarşik gruplar oluşturabilen bir yöntemdir.

Bu tez çalışmasında sunulan gruplama yöntemi:

– İki ağacın elemanlarının seviyelerine göre benzerlik hesaplaması sonuçlarına göre gruplandırma,
– İki ağacın hiyerarşi bilgileri kullanılarak benzerlik hesaplaması sonuçlarına göre gruplandırma yöntemleri ile karşılaştırılmıştır.

Karşılaştırılan diğer gruplama yöntemleri, EWRIDB verilerinin gruplandırılmasında başarısız olmuşlardır. Karşılaştırılan her iki yöntemin sonuçlarında; bir grupta bulunan eleman sayısı 90'nın üzerinde olmuş ve diğer gruplarda çok az eleman yer almıştır. Bu tez çalışmasında kullanılan yöntemle, elemanlar gruplara daha dengeli olarak ve bir grupta en çok 25 eleman olacak şekilde dağılmıştır. Bu çalışmada sunulan gruplama metodu ve benzerlik bulma yöntemi ile veritabanında sorgulamalar yapılmıştır. Benzerlikleri bulunurken sorgu sonuçlarını iyileştirebilmek için iki değişken tanımlanmıştır. Bu değişkenler:

– Grupları gezerken elde edilen yapısal benzerliğin eşik değeri,
– Sorgu sonucu olarak dönebilmesi için ağaçların yapısal ve içerik olarak benzerliğinin eşik değeridir.

Sunulan benzerlik bulma yöntemi :

– Ağaçların yapısal özelliklerine göre,
– Ağaçların hem yapısal hem de içeriğine göre benzerlik oranı bulan yöntemler ile karşılaştırılmıştır.

Bu karşılaştırmalarda, EWIRDB verileri ile aynı karakteristik özelliklere sahip olan ve ortalama 377 elemanı bulunan hiyerarşik veriler kullanılmıştır. Bu veriler üzerinde değişik sorgulamalar ve performans testleri yapılmıştır. Yapısal özellikleri kullanan benzerlik bulma yönteminin daha performanslı olduğu görülmüştür. Fakat bu yöntemde ağacın parametrelerinin içeriği göz ardı edildiği için sonuçlarda hatalı verilerin bulunma olasılığı artmıştır. Yapısal ve içerik kontrolü yapan yöntemin performans değeri diğerine göre beş kat kötüdür. Verilerin içeriklerinin kontrolünün yapılması performans kaybına sebep olduğu görülmüştür.

Bu çalışmada önerilen yöntem ile yapısal ve içerikleri kullanan yöntemin performansında çok büyük farklar bulunmamıştır. Ortalamalar göz önüne alındığında ortalama performanslar her üç yöntem için 2 milisaniyenin altında olmuştur. Bu çalışmada kullanılan yöntem ek performans kaybına sebep olmamış, fakat daha doğru sonuçların elde edilmesini sağlamıştır. Bu çalışmada sunulan yöntemle elde edilen başarı %97'nin altına düşmediği görülmüştür. Diğer yöntemlerde, grup yapısal benzerlik eşik değeri düştükçe doğruluk oranı düşmüş ve bu eşik değeri yükseldikçe başarı yüzdesi %97'e çıkmamıştır. Bu eşik değeri %90'nın altına düştüğünde yapısal benzerlikle elde edilen başarı %80'nin altında olmuştur. İki ağacın benzerlik kriteri %50 olarak belirlendiğinde diğer iki yöntem ile elde edilen başarı %50'nin altına düşerken, bu çalışmada sunulan yöntemle elde edilen sonuçlarda başarı hala %97'nin üstünde olmuştur.

Çalışma sonunda, EWIRDB verilerinin etkin yönetimi ve kullanımı sağlanmıştır. Veritabanında hızlı şekilde ve doğru benzerlik sorgulamaları yapılabilir bir alt yapı hazırlanmış ve bu alt yapı kullanılarak örnek bir uygulama geliştirilmiştir.

KAYNAKLAR

1. Adamy, D., "EW 101: a first course in electronic warfare", ***Artech House***, Norwood, Massachusetts, USA, 1-6 (2001).

2. Schleher, D. C., "Electronic warfare in the information age", ***Artech House***, Norwood, Massachusetts, USA, 1-15 (1999).

3. Coyne, K. M., "Design and analysis of an object-oriented database of electronic warfare data", Yüksek Lisans Tezi, ***Naval Postgraduate School***, Monterey, California, USA, 1-17, 37-64, 83-85 (1996).

4. Lee, J. J., McKenna, T. D., "The object-oriented database and processing of electronic warfare data", Yüksek Lisans Tezi, ***Naval Postgraduate School***, Monterey, California, USA, 1-4, 15-19, 28-33,46-73 (1996).

5. Scrivener, D. N., Edwards, R. D., "Reactivation of relational interface in M2DBMS and implementation of EWIR database", Yüksek Lisans Tezi, ***Naval Postgraduate School***, Monterey, California, USA, 1-20, 39-51, 61-62 (1996).

6. Werre, T. J., Diehl, B. A., "The activation and testing the network CODASYL-DML interface of the M2DBMS using the EWIR database", Yüksek Lisans Tezi, ***Naval Postgraduate School***, Monterey, California, USA,1-7, 36-39, 89-105, 117-119 (1996).

7. Barnes, G. B., "A conceptual approach to object-oriented data modeling", Yüksek Lisans Tezi, ***Naval Postgraduate School***, Monterey, California, USA, 1-4, 62-73, 85-88 (1994).

8. Bisbal, J., Lawless, D., Bing W., Grimson, J., Wade, V., Richardson, R., O'Sullivan, D., "An overview of legacy information system migration", ***Proceedings of the 4th Asia-Pacific Software Engineering and International Computer Science Conference***, Hong Kong, 529-530 (1997).

9. Tümay, A., Dinçer, K., Yürekten, Ö., Sungur M., Dikici, A, Tambağ, Y., "A metadata repository model to integrate heterogeneous databases of hardware dependent legacy systems", ***The 4th Biennial International Conference on Advances in Information Systems***, İzmir, Türkiye, LNCS 4243: 149-157 (2006).

10. Bisbal, J., Lawless, D., Wu, B., Grimson, J., "Legacy information systems: issues and directions", ***IEEE Software Journal***, 16(5): 103-111 (1999).

11. Brodie, M. L., Stonebraker, M., "Migrating legacy systems: gateways, interfaces and the incremental approach", ***Morgan Kaufmann Publishers Inc.,*** San Francisco, USA, 210-210 (1995).

12. Meier, W., "eXist: an open source native XML database", ***Web, Web Services, and Database Systems, Web and Database Related Workshops***, Erfurt, Germany, LNCS 2593: 169-183 (2003).

13. Chamberlin, D., "XQuery: a query language for XML", ***Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data***, San Diego, California, USA, 682-682 (2003).

14. Brundage, M., "XQuery: the XML query language", ***Addison-Wesley Professional***, Boston, Massachusetts, USA, 5-14, 19-21, 63-86, 227-240, 463-472 (2004).

15. Some, R.R., A. Czikmantory, A., Neff, J., Marshall, M., "XML hierarchical database for missions and technologies", ***IEEE Aerospace Conference Proceedings***, Big Sky, Montano, USA, 1: 303-313 (2004).

16. Some, R., Neff, J., "XML technology planning database: lessons learned", ***IEEE Aerospace Conference Proceedings***, Big Sky, Montano, USA, 1-15 (2005).

17. Ida, M., Nozawa, T., Yoshikane, F., Miyazaki, K., Kita, H.: "Syllabus database and web service on higher education", ***The 7th International Conference on Advanced Communication Technology***, Seoul, Korea, 1: 415-418 (2005).

18. Philippi, S., Köhler, J., "Using XML technology for the ontology-based semantic integration of life science databases", ***IEEE Transactions on Information Technology in Biomedicine***, USA, 8(2): 154-160 (2004).

19. Sokic, M., Matic, V., Bazent, A., "Web content management system based on XML native database", ***Proceedings of the 25th International Conference on Information Technology Interfaces,*** Dubrovnik, Croatia, 457-462 (2003).

20. Molitoris, J. J., "Use of COTS XML and web technology for current and future C2 systems", ***Military Communications Conference***, Monterey, California, USA, 1: 221-226 (2003).

21. Yürekten, Ö., Dinçer, K., Akar, B., Sungur, M., Özbudak, E. K., "Migrating a hierarchical legacy database application onto an XML-based service-oriented web platform", ***The 21st International Symposium on Computer and Information Systems***, İstanbul, Türkiye, LNCS 4263: 645-654 (2006).

22. Myers, G., "A four russians algorithm for regular expression pattern matching", ***Journal of the ACM***, 39(2): 432-448 (1992).

23. Chawathe, S. S., Rajaraman, A., Garcia-Molina, H., Widom, J., "Change detection in hierarchically structured information", ***Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data***, New York, USA, 25(2): 493-504 (1996).

24. Chawathe, S. S., Garcia-Molina, H., "Meaningful change detection in structured data", ***Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data***, New York, USA, 26-37 (1997).

25. Lee, J. W., Lee, K., Kim, W., "Preparations for semantic-based XML mining", ***Proceedings of the 2001 IEEE International Conference on Data Mining***, San Jose, California, USA, 345-352 (2001).

26. Nierman, A., Jagadish, H. V., "Evaluating structural similarity in XML documents", ***Proceedings of the 5th International Workshop on the Web and Databases***, Madison, Wisconsin, USA, 61-66 (2002).

27. Ma, Y., Chbeir, R., "Content and structure based approach for XML similarity", ***IEEE the 5th International Conference on Computer and Information Technology***, Shanghai, China, 136-140 (2005).

28. Chang, C. H., Lui, S. C., Wu, Y. C., "Applying pattern mining to web information to web information extraction", ***Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining***, Taipei, Taiwan, LNAI 2035: 4-15 (2001).

29. Miyahara, T., Suzuki, Y., Shoudai, T., Uchida, T., Takahashi, K., Ueda, H., "Discovery of frequent tag tree patterns in semistructured web documents", ***Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining***, Taipei, Taiwan, LNCS 2336: 341-355 (2002).

30. Lee, J. S., Lee, K. H., "XML schema matching based on incremental ontology update", ***ACM Workshop on Wireless Security***, Philadelphia, USA., LNCS 3306: 608-618 (2004).

31. Ceravolo, P., Nocerino, M. C., Viviani, M., "Knowledge extraction from semi-structured data based on fuzzy techniques", ***The 9th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems***, Wellington, New Zealand, LNAI 3215: 328-334 (2004).

32. Leung, H., Chung F., Chan, S. C., "A new sequential mining approach to xml document similarity computation", ***The 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining***, Seoul, Korea, LNAI 2637: 356-362 (2003).

33. Leung, H., Chung, F., Chan, S. C., "On the use of hierarchical information in sequential mining-based XML document similarity computation", ***Knowledge and Information Systems***, 7 (4): 476-498 (2005).

34. Zhang, Z., Li, R., Cao, S., Zhu, Y., "Similarity metric for XML documents", ***The Workshop on Knowledge and Experience Management***, Karlsruhe, Germany, 1-7 (2003).

35. Canfora, G., Cerulo L., Scognamiglio, R., "Measuring XML document similarity: a case study for evaluating information extraction systems", ***IEEE Proceedings of the Software Metrics, 10th International Symposium on Software Metrics***, Chicago, Illinois, USA, 36-45 (2004).

36. Xie, T., Sha, C., Wang, X., Zhou, A., "Approximate top-k structural similarity search over XML documents", ***The 8th Asia-Pacific Web Conference***, LNCS 2841: 319-330 (2006).

37. Patrick, K. L. N., Vincent, T. Y. N., "Structural similarity between XML documents and DTDs", ***Lecture Notes in Computer Science***, 2659: 412-421 (2003).

38. Bertino, E., Guerrini, G., Mesiti, M., "Matching an XML document againts a set of DTDs***", International Syposium on Methodologies for Intelligent Systems***, Lyon, France, LNAI 2366: 412-422 (2002).

39. Bertino, E., Guerrini, G., Mesiti, M., "A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications", ***Information Systems***, 29(1): 23 - 46 (2004).

40. Kriegel, H. P., Schönauer, S., "Similarity search in structured data", ***Data Warehousing and Knowledge Discovery Workshop***, Prague, Czech Republic, LNCS 2737: 309-319 (2003).

41. Lee, J. W., Park, S. S., "Computing similarity between XML documents for XML mining", ***The 14th International Conference on Knowledge Engineering and Knowledge Management***, Whittlebury Hall, Northamptonshire, UK, LNAI 3257: 492-493 (2004).

42. Nayak, R., Iryadi, W., "XMine: a methodology for mining XML structure", ***The 8th Asia-Pacific Web Conference***, Harbin, China, LNCS 3841: 786-792 (2006).

43. Shen, Y., Wang, B., "Clustering schemaless XML documents", ***International Conference on Ontologies, Databases and Applications of Semantics***, Catania, Sicily, Italy, LNCS 2888: 767-784 (2003).

44. Nayak, R., Xu, S., "XML documents clustering by structures", ***Initiative for the Evaluation of XML Retrieval Workshop***, Dagstuhl Castle, Germany, LNCS 3977: 432–442 (2006).

45. Nayak, R., Xu, S., "XCLS: a fast and effective clustering algorithm for heterogonous XML documents", ***Proceedings of the 10th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining,*** Singapore, LNAI 3918: 292–302 (2006).

46. Nayak, R., Brisbane, A., "Investigating semantic measures in XML clustering", ***IEEE/WIC/ACM International Conference on Web Intelligence***, Hong Kong, 1042-1045 (2006).

47. Popovici, E., Marteau, P. F., Menier, G., "Information retrieval of sequential data in heterogeneous XML databases", ***The 4th International Workshop on Adaptive Multimedia Retriaval***, Geneva, Switzerland, LNCS 3877: 236-250 (2006).

48. Lee, J. M., Hwang, B. Y., "Path bitmap indexing for retrieval of XML documents", ***Modeling Decisions for Artificial Intelligence Workshop***, Tarragona, Catalonia, Spain, LNAI 3885: 329-339 (2006).

49. Lee, J. M., Hwang, B. Y., "Two-phase path retrieval method for similar XML document retrieval", ***The 9th International Conference on Knowledge-Based & Intelligent Information and Engineering Systems***, Melbourne, Australia, LNAI 3681: 967-971 (2005).

50. Hwang, J. H., Ryu, K. H., "Clustering and retrieval of XML documents by structure", ***International Conference on Computational Science and its Applications***, Singapore, LNCS 3481: 925-935 (2005).

51. Hwang, J. H., Ryu, K. H., "A new sequential mining approach to XML document clustering", ***The 7th Asia Pacific Web Conference***, Shanghai, China, LNCS 3399: 266-276 (2005).

52. Hwang, J. H., Ryu, K. H., "A new XML clustering for structural retrieval", ***International Conference on Conceptual Modeling***, Shanghai, China, LNCS 3288: 377-387 (2004).

53. Wang, T., Liu, D. X., Lin, X. Z., Sun, W., Ahmad, G., "Clustering large scale of XML documents", ***International Conference on Grid and Pervasive Computing***, Tunghai University, Taiwan, LNCS 3947: 447-455 (2006).

54. Dalamagas, T., Cheng, T., Winkel, K. J., Sellis, T., "Clustering XML documents by structure", ***The 3rd Hellenic Conference on Artificial Intelligence***, Samos, Greece, LNAI 3025: 112-121 (2004).

55. Dalamagas, T., Cheng, T., Winkel, K. J., Sellis, T., "Clustering XML documents using structural summaries", ***The 9th International Conference on Extending Database Technology***. Heraklion, Greece, LNCS 3268: 547-556 (2004).

56. Dalamagas, T., Cheng, T., Winkel, K. J., Sellis, T., "A methodology for clustering XML documents by structure", ***Information Systems***, 31(3): 187-228 (2006).

57. Hatano, K., Kinutani, H., Yoshikawa, M., Uemura, S., "Information retrieval system for XML documents", ***The 13th International Workshop on Database and Expert Systems Applications***, Aix-en, France, LNCS 2453: 758-767 (2002).

58. Sanz, I., Pezer, J. M., Berlanga, R., Aramburu, M. J., "XML schemata inference and evolution", ***The 14th International Workshop on Database and Expert Systems Applications***, Prague, Czech Republic, LNCS 2736: 109-118 (2003).

59. Costa, G., Manco, G., Ortale, R., Tagarelli, A., "A tree-based approach to clustering XML documents by structure", ***The 8th European Conference on Principles and Practice of Knowledge Discovery in Databases***, Pisa, Italy, LNAI 3202: 137-148 (2004).

60. Apshankar, K., Waterhouse, M., Zhang, L. J., O'Riordan, D., Sadhwani, D., "Web services business strategies and architectures", ***Expert Press***, Chicago, Illinois, USA, 136-137 (2002).

61. Sevdik, A. B., "Elektronik devlette genişletilebilir biçimleme dili (XML): bazı örnek senaryolar", Yüksek Lisans Tezi, ***Bilkent Üniversitesi Mühendislik ve Fen Bilimleri Enstitüsü***, Ankara, 1-5, 16-20, 25-26 (2004).

62. Kuru, İ., "E-öğrenme projelerinde XML web servislerinin kullanımı ve örnek uygulama", Yüksek Lisans Tezi, ***İstanbul Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 1-2, 4-7 (2006).

63. Monson-Haefel, R., "The ultimate guide J2EE web services", ***Addison-Wesley***, Boston, Massachusetts, USA, 17-77, 603-651 (2006).

64. Erl, T., "Service-oriented architecture a field guide to integrating XML and web services", ***Prentice Hall***, USA, 2-4, 18-44 (2004).

65. Ramakrishnan, R., Gehrke, J., "Database management systems", ***McGraw-Hill Companies Inc.,*** New York, USA, 605-639, 772-810 (2002).

66. Elmasri, R., Navathe, S. B., "Fundamentals of database systems", ***Addison-Wesley***, USA, 147-161, 229-248 (2000).

67. Yarımağan, Ü., "Veri tabanı sistemleri", ***Akademi Yayın Hizmetleri San. Tic. Ltd. Şti.***, Ankara, 65-99, 201-223 (2000).

68. Wang, F., Zaniolo, C., "Publishing and querying the histories of archived relational databases in XML", ***Proceedings of the 4th International Conference on Web Information Systems Engineering***, Roma, Italy, 93-102 (2003).

69. Jiang, H., Lu, H., Wang, W., Yu, J. X., "XParent: an efficient RDBMS-based XML database system", ***IEEE Computer Society Proceedings of the 18th International Conference on Data Engineering***, San Jose, California, USA, 335-336 (2002).

70. Jagadish, H. V., Al-Khalifa, S., Chapman, A., Lakshmanan, L. V. S., Nierman, A., Paparizos, S., Patel, J. M., Srivastava, D., Wiwatwattana, N., Wu, Y., Yu, C., "TIMBER: a native XML database", ***The International Conference on Very Large Databases***, Hong Kong, China, 11(4): 274-291 (2002).

71. Lu, H., Yu, J. X., Wang, G., Zheng, S., Jiang, H., Yu, G., Zhou, A., "What makes the differences: benchmarking XML database implementations", ***IEEE Proceedings of the 19th International Conference on Data Engineering***, Bangalore, India, 1: 720-722 (2003).

72. Lawrence, R., "The space efficiency of XML", ***Information and Software Technology***, 46(11): 753-759 (2004).

73. Salminen, A., Tompa, F. W., "Requirements for XML document database systems", ***Proceedings of the 2001 ACM Symposium on Document Engineering***, Atlanta, Georgia, USA, 85 - 94 (2001).

74. Wan, X., Yang, J., "Using proportional transportation similarity with learned element semantics for XML document clustering", ***Proceedings of the 15th International Conference on World Wide Web***, Edinburg, Scotland, 961-962 (2006).

75. Yang, J., Cheung, W. K., Chen, X., "Integrating element and term semantics for similarity-based XML document clustering", ***IEEE/WIC/ACM International Conference on Web Intelligence***, France, 222-228 (2005).

76. Yoon, J., Raghavan, V., Chakilam, V., Kerschberg, L., "BitCube: a three dimensional bitmap indexing for XML documents", ***Journal of Intelligent Information Systems***, 17(2-3) 241-254 (2001).

77. Ciaccia, P., Penzo, W., "Adding flexibility to structure similarity queries on XML data", ***The 5th International Conference on Flexible Query Answering Systems***, Copenhagen, Denmark, LNAI 2522: 124-139 (2002).

78. Kailing, K. Kriegel, H.-P. Schonauer, S. Seidl, T., "Efficient similarity search in large databases of tree structured objects", ***The 20th International Conference on Data Engineering***, Boston, Massachusetts, USA, 835-836 (2004).

79. Flesca, S., Manco, G., Masciari, E., Pontieri, L., Pugliese, A., "Fast detection of XML structural similarity", ***IEEE Transactions on Knowledge and Data Engineering***, 17(2): 160-175 (2005).

80. Schlieder, T., "Fast similarity search in XML data", Doktora Tezi, ***Freie University Institute of Mathematic and Informatics***, Berlin, Germany, 1-6, 9-10, 22-23, 175-178 (2003).

81. Carrasco, R. C., Rico-Juan, J. R., "A similarity between probabilistic tree languages: application to XML document families", ***Pattern Recognition***, 36(9): 2197-2199 (2003).

82. Bordawekar, R., Shmueli, O., "Flexible workload-aware clustering of XML documents", ***Database and XML Technologies, Second International XML Database Symposium***, Toronto, Canada, 204-218 (2004).

83. Guillaume, D., Murtagh, F., "Clustering of XML documents", ***Computer Physics Communications***, 127(2-3): 215-227 (2000).

84. Doucet, A., Ahonen-Myka, H., "Naive clustering of a large XML document collection", ***Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval***, Dagstuhl, Germany, 81-87 (2002).

85. Francesca, D. F., Gordano, G., Ortale, R., Tagarelli, A., "Distance-based clustering of XML documents", ***The 7th European Conference on Principles and Practice of Knowledge Discovery in Databases***, Cavtat, Dubrovnik, Croatia, 75–78 (2003).

86. Tagarelli, A., Greco, S., "Clustering transactional XML data with semantically-enriched content and structural features", ***The 5th International Conference on Web Information Systems Engineering***, Brisbane, Australia, LNCS 3306: 266-278 (2004).

87. Candillier, L, Tellier, I., Torre, F., "Transforming XML trees for efficient classification and clustering", ***Workshop on Mining XML Documents***, Dagstuhl, Germany, 469-480 (2005).

88. Despeyroux, T., Lechevallier, Y., Trousse, B., Vercoustre, A. M., "Experiments in clustering homogeneous XML documents to validate an existing typology", ***Proceedings of the 5th International Conference on Knowledge Management,*** Vienne, Autriche, 1-8 (2005).

89. Zhang, X. Z., Lv, T. Y., Wang, Z., Zuo, W., "Clustering XML documents by structure based on common neighbor", ***International Conference on Computational Intelligence and Security***, Xi'an, China, LNCS 3801: 771-776 (2005).

90. Choi, I., Moon, B., Kim, H. J., "A clustering method based on path similarities of XML data", ***Data & Knowledge Engineering***, 60(2): 361-376 (2007).

91. Lian, W., Cheung, D. W., Mamoulis, N., Yiu, S. M., "An efficient and scalable algorithm for clustering XML documents by structure", ***IEEE Transactions on Knowledge and Data Engineering***, 16(1): 82-96 (2004).

EKLER

EK-1. Çalışmada kullanılan terimler ve ingilizce karşılıkları

Bu çalışmada kullanılmış olan bazı Türkçe kelimelerin ve terimlerin İngilizce karşılıkları aşağıda sunulmuştur.

Türkçe Kelime / Terim	İngilizce Karşılığı
ABD Haberleşme için Olamayan Sistemler Veritabanı	United States of America Noncommunications Systems Database
Ağ Veritabanı Modeli	Network Database Model
Ağaç Güncelleme Mesafesi	Tree Edit Distance
Ardışık Madencilik Yaklaşımı	Sequential Mining Approach
Ardışık Örüntüsü Madenciliği	Sequential Pattern Mining
Betik	Script
Bilimsel ve Teknik İstihbarat Kurumu	Scientific and Technical Intelligence
Birincil Anahtar	Primary Key
Birleşik Kaynak Tanımlayıcı	Uniform Resource Identifier (URI)
Birleştirilmiş Modelleme Dili	Unified Modeling Language (UML)
Birleştirme	Aggregation
Büyük Yazı İşaretleme Dili	Hyper Text Markup Language
Çok Dilli ve Çok Modelli Veritabanı Modelleme Sistemi	Multi-lingual, Multi-model Database Modeling System
Çoklu Soya Çekim	Multiple Inheritance
Dağıtık	Distributed
Doğal XML Veritabanı	Native XML Database
Doküman Nesne Modeli	Document Object Model (DOM)
Doküman Tipi Tanımı	Document Type Definition (DTD)
Düz Dosya	Plain Text File
Elektronik Harp	Electronic Warfare
Elektronik Harp Birleştirilmiş Yeniden Programlanabilir Veritabanı	Electronic Warfare Integrated Reprogramming Database (EWIRDB)
Elektronik İstihbarat	Electronic Intelligence (ELINT)

EK-1. (Devam) Çalışmada Kullanılan Türkçe Terimlerin İngilizce Karşılıkları

Türkçe Kelime / Terim	İngilizce Karşılığı
Eleman	Node, Element
Emiter	Emitter
Etiket	Tag, Label
Etiketlenmiş Ağaç Örüntüsü	Tag Tree Pattern
Evrensel Kod Birliği	Unicode
Genel İşaretleme Dili	Generalize Markup Language (GML)
Genelleştirme	Generalization
Genişleyebilir Biçimlendirme Dili Dönüşümleri	Extensible Style Sheet Language Transformations (XSLT)
Genişleyebilir İşaretleme Dili	Extensible Markup Language (XML)
Geri Alma	Roll back
Gruplama	Clustering
Hiyerarşik	Hierarchical
İkili Dosya	Binary File
İlişkili Emiter	Associated Emitter
İndeks Yöneticisi	Index Manager
İsim Uzayı	Namespace
İş Tamamlama	Transaction
İşaretleme Dili	Markup Language
İşaretleyici	Marker
İşlenmiş Karakter Verisi	Parsed Character Data (PCDATA)
İyi Organize Edilmiş	Well-Formed
Java Sunucu Yüzleri	Java Server Faces (JSF)
Karakter Verisi	Character Data (CDATA)
Koleksiyon	Collection
Kullanım Durumu Diyagramı	Use Case Diagram
Makine Tarafından Anlaşılabilir	Machine Understandable

EK-1. (Devam) Çalışmada Kullanılan Türkçe Terimlerin İngilizce Karşılıkları

Türkçe Kelime / Terim	İngilizce Karşılığı
Miras	Legacy
Nesne Sorgulama Dili	Object Query Language (OQL)
Nesne Yönelimli	Object Oriented (OO)
Nesne Yönelimli Programlama	Object Oriented Programming (OOP)
Olay	Event
Örnek	Instance
Özelleştirme	Specialization
Özellik	Attribute
Pratik Sayı ve Harflerle Kodlanmış Bilgi Getirme Algoritması	Practical Algorithm to Retrieve Information Coded in Alphanumeric (PAT)
Rafta Hazır Ticari Ürün	Commercial Off- The Shelf (COTS)
Saha Sınıfları	Domain Classes
Son Ek	Suffix
Sorgu Değerlendiricisi	Query Evaluator
Sorgu İşleyicisi	Query Parser
Sorgu Optimizasyoncusu	Query Optimizer
Soya Çekim	Inheritance
Standart Genel İşaretleme Dili	Standard Generalized Markup Language
Tek Başına Çalışabilen	Standalone
Tetikleyici	Trigger
Ulusal Güvenlik Ajansı	National Security Agency (NSA)
Ulusal Havacılık ve Uzay Dairesi	National Aeronautics and Space Administration (NASA)
Veri Depolama Yöneticisi	Data Storage Manager
Veri Madenciliği	Data Mining
Veri Parçalayıcı	Data Parser

EK-1. (Devam) Çalışmada Kullanılan Türkçe Terimlerin İngilizce Karşılıkları

Türkçe Kelime / Terim	İngilizce Karşılığı
Veri Sistemleri Dili Konferansı ve Veri Güncelleme Dili	Conference on Data Systems Language, Data Manipulation Language
Veri Tanımlama Dili	Data Definition Language
Veri Yöneticisi	Data Manager
XML Dokümanı için Basit Altyapı	Simple API for XML (SAX)
XML Dokümanı İşlemek İçin Java Altyapısı	Java API for XML Processing (JAXP)
XML Dokümanlarını Bağlamak İçin Java Altyapısı	Java API for XML Binding (JAXB)
XML Dokümanlarını Dolaylı Gruplama	Indirect Clustering for XML Documents (ICLS)
XML Formatını Destekleyen Veritabanı	XML Enabled Database
XML Şema Tanımlama	XML Schema Definition (XSD)
Yabancı Anahtar	Foreign Key
Yapı Yöneticisi	Metadata Manager
Yol	Path

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: YÜREKTEN, Özgür
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 17.08.1979 Tavas / Denizli
Medeni hali	: Evli
Telefon	: 0 (312) 426 67 89 - 2269
Faks	: 0 (312) 426 48 80
e-mail	: ozgur.yurekten@tubitak.gov.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Başkent Üniversitesi / Bilgisayar Mühendisliği	2001
Lise	Yunusemre Anadolu Öğretmen Lisesi	1997

İş Deneyimi

Yıl	Yer	Görev
2003-	TÜBİTAK-UEKAE / ANKARA	Proje Takım Lideri
2002-2003	Hv.K.K. Karargahı / ANKARA	Bilgisayar Subayı
2001-2002	TÜBİTAK-UEKAE / ANKARA	Araştırmacı
2000-2001	TÜBİTAK-UEKAE / ANKARA	Yarı Zamanlı Programcı

Yabancı Dil

İngilizce

Yayınlar

1. Yürekten, Ö., Dinçer, K., Akar, B., Sungur, M., Özbudak, E. K., “Migrating a Hierarchical Legacy Database Application onto an XML-Based Service-Oriented

Web Platform", ***The 21st International Symposium on Computer and Information Systems***, İstanbul, Türkiye, LNCS 4263: 645-654 (2006).

2. Tümay, A., Dinçer, K., Yürekten, Ö., Sungur M., Dikici, A, Tambağ, Y., "A Metadata Repository Model to Integrate Heterogeneous Databases of Hardware Dependent Legacy Systems", ***The 4th Biennial International Conference on Advances in Information Systems***, İzmir, Türkiye, LNCS 4243: 149-157 (2006).

3. Yürekten, Ö., Dinçer, K., Tanyer, S. G., İngeç, T. S., "İşaret Karakteristikleri Kütüphanesi Uygulama Yazılımı", ***10. Sinyal İşleme ve İletişim Uygulamaları Kurultayı***, Denizli, 676-680 (2002).

4. Tümay, A., Dinçer, K., Avcı, O. O., Demirsoy, M., Erdem, A., Yürekten, Ö., Sungur, M., Erdem, A., Özbudak, E. K., "Information System Infrastructure for a National Crisis Management Center", ***The 9th International Conference of International Emergency Management Society***, Ontario, Canada (2002).

5. Dinçer, K., Yürekten, Ö., Oğuz, B., Durgar, İ., "Üniversite Ortamında İnternet Erişimli Bir Öğrenci Yönetim ve Karar Destek Sistemi Uygulaması", ***5. Türkiye'de İnternet Konferansı***, Ankara (1999).

Hobiler

Bilgisayar, Futbol, Gezi, Pul koleksiyonu, Motosiklet