XML'in ZAMAN ve YER ETKİNLİĞİ AÇISINDAN İNCELENMESİ

Erdinç Uzun
Trakya Üniversitesi
Bilgisayar Müh. Bölümü
erdinc@trakya.edu.tr

Erdem Uçar
Trakya Üniversitesi
Bilgisayar Müh. Bölümü
erdemu@trakya.edu.tr

Yılmaz Kılıçaslan
Trakya Üniversitesi
Bilgisayar Müh. Bölümü
yilmazk@trakya.edu.tr

ÖZET

XML, hızla yaygınlaşan geleceğin web alt yapısı için düşünülmüş bir biçimleme dilidir. XML; okunabilirlik ve tanımlanabilirlik açısından uygun bir dil olsa da aslında etiket tekrarlarının fazla olması yer etkinliği sorununu beraberinde getirmektedir. Bu çalışmamızda yer etkinliği sorununun nelere yol açtığını ve bu sorundan kurtulma yöntemleri hakkındaki görüşlerimizi bildireceğiz.

ABSTRACT

XML is a widely spreading markup language designed for the web base of tomorrow. Even though Xml is an efficient language in terms of readibility and describility, it has ostacles on space efficiency because of the redundant use of tags. In this studythe cost of space effiency problem and its probable solutions are discussed.

Anahtar Kelimeler: XML, Yer Etkinliği, Zaman Etkinliği, JSON, SMEL, YAML

1. GİRİŞ

XML özellikle web ortamında veri taşımacılığında CSV (Comma Seperated Values – Virgül ile ayrılmış değerler) ve FSV (Fixed Seperated Files – Sabit ayrılmış dosyalar) gibi veri taşımakta kullanılan standartların yerini almaya başlamıştır.

XML ise oluşan her veri için onu açıklayıcı başlangıç ve bitiş etiketleri kullanır. XML sayesinde etiketler içindeki verinin ne anlama geldiğini kolay bir şekilde anlayabiliriz. Kullanılan etiketler verinin okunabilirliğini ve anlaşabilirliğini arttırmasına rağmen CSV ve FSV dosya yapılarına göre XML dosyaları dosya boyutu açısından daha büyük dosyalar oluşturmaktadır. Dosya boyutu büyüdüğü içinde bu dosyanın oluşturulması ve erişim süreleri de doğru orantılı bir şekilde artmaktadır. [8]

Dosya boyutunun büyüklüğünü göz önüne alan araştırmacılar çeşitli XML alternatifler ve sıkıştırma teknikleri ortaya koymuşlardır. JSON[2], YAML[3], SMEL[1] gibi alternatif yöntemler amacı XML'deki etiket tekrarlarını azaltarak yine okunabilirliği ve anlaşılabilirliği olan yeni biçimleme dilleri yaratmaya çalışmışlardır. Dosya boyutunu azaltmak için çeşitli sıkıştırma teknikleri üzerinde durulmuştur. [5,6,7,9] Özellikle etiket tekrarı fazla olduğu için XML dosyaları yüksek oranda sıkışan dosyalardır.

Bundan sonraki bölümde CSV, FSV, XML, JSON, YAML ve SMEL gibi yöntemleri sözdizimsel olarak inceleceğiz. Ardından XML, JSON, YAML, SMEL gibi biçimleme dilleri üzerine yaptığımız performans ölçümleri hakkında bilgi verdikten sonra son bölümde ise bu konu üzerine çeşitli önerilerde bulunacağız.

2. SÖZDİZİMSEL OLARAK CSV, FSV, XML, JSON, YAML ve SMEL'in İNCELENMESİ

2.1. FSV ve CSV

CSV'de metin dosyada veriler virgülle birbirinden ayrılırken FSV'de ise sabit uzunluklarla birbirinde ayrılmaktadır.[4]

```
Definition:
Authors: nr, firstname, lastname, email
Data:
0,Erdinç,Uzun,erdinc@trakya.edu.tr
1,Yılmaz,Kılıçaslan,yilmazk@trakya.edu.tr
2,Erdem,Uçar,erdemu@trakya.edu.tr
```

Örnek 1. CSV Örneği

```
Definition:
Authors:
        nr(5)
        firstname(25)
        lastname(25)
        email(25)
Data:
0  Erdinç          Uzun
erdinc@trakya.edu.tr
1  Yılmaz          Kılıçaslan
yilmazk@trakya.edu.tr
2  Erdem           Uçar
erdemu@trakya.edu.tr
```

Örnek 2. FSV Örneği

CSV ve FSV her ne kadar veri taşımacılığında kullanılabilse de hiyerarşik yapılar karşısında yetersiz

kalan yöntemlerdir. Özellikle ilişkisel verileri taşımada eksiklikleri vardır. Bu gibi eksiklikleri ortadan kaldırmak için XML gibi bir biçimleme dili gerekir.

2.2. XML

XML, CSV ve FSV de taşınamayan hiyerarşik verilerin taşınması için uygun bir biçimleme dilidir. Microsoft, IBM, Oracle gibi büyük geliştiriciler tarafından da desteklenmekte olan bir dildir. XML hiyerarşik bilimsel verileri (DNA dizisi, linguistic veri kümeleri gibi) tutmada, internet ortamında ilişkisel verilerin taşınması için kullanılmaktadır. (12)

Her XML elementi başlangıç ve bitiş etiketleri arasına yazılır. Her etiketin bir anlamı vardır. En içteki etikette veri bulunur. Etiket uzunluğu değişken uzunluktadır.

```
<authors>
        <nr>0</nr>
        <firstname>Erdinç</firstname>
        <lastname>Uzun</lastname>
        <email>erdinc@trakya.edu.tr</email>
</authors>
<authors>
        <nr>1</nr>
        <firstname>Yılmaz</firstname>
        <lastname>Kılıçaslan</lastname>
        <email>yilmazk@trakya.edu.tr</email>
</authors>
<authors>
        <nr>2</nr>
        <firstname>Erdem</firstname>
        <lastname>Uçar</lastname>
        <email>erdemu@trakya.edu.tr</email>
</authors>
```

Örnek 3. XML Örneği

XML'deki etiket tekrarını göz önüne alan araştırmacılar çeşitli alternatiflere yönelmişlerdir.

2.3. JSON

JSON (JavaScript Object Notation) [6] etiket tekrarın daha az yapıldığı bir veri taşıma ortamı sunar. JSON'da etiket, veri yada iç etiket etiketler içerebilir. Eğer iç etiket varsa ":" karakterinden sonra "{" karakteri kullanılır. "{" etiketi C'deki gibi başlangıç anlamına gelir ta ki "}" karakterine kadar olan bölüm başlangıçtan önceki etiketin içindedir. Eğer veri varsa ":" karakterinden sonra tırnak karakterleri içine veri yazılır.

```
"authors": {
                "nr": "0",
                "firstname": "Erdinç",
                "lastname": "Uzun",
                "email": "erdinc@trakya.edu.tr"
        }
"authors": {
                "nr": "1",
                "firstname": "Yılmaz",
                "lastname": "Kılıçaslan",
                "email": "yilmazk@trakya.edu.tr"
        }
"authors": {
                "nr": "2",
                "firstname": "Erdem",
                "lastname": "Uçar",
                "email": "erdemu@trakya.edu.tr"
        }
```

Örnek 4. JSON Örneği

2.4. YAML

YAML, JSON daki tekrarlarına ortadan kaldıran bir yaklaşımdır. "{,}" ve tırnak gibi karakterler ortadan kaldırılmış sadece ":" karakteri kullanılmaktadır.

```
authors:
        - nr        : 0
          firstname : Erdinç
          lastname  : Uzun
          email     : erdinc@trakya.edu.tr
        - nr        : 2
          firstname : Yılmaz
          lastname  : Kılıçaslan
          email     : yilmazk@trakya.edu.tr
        - nr        : 3
          firstname : Erdem
          lastname  : Uçar
          email     : erdemu@trakya.edu.tr
```

Örnek 5. YAML Örneği

2.5. SMEL

SMEL yaklaşımı CSV yaklaşımının geliştirilmiş şeklidir. Fakat okunabilirlik ve anlaşılabilirlik açısından XML, JSON ve YAML tekniklerine göre farklı bir yaklaşımdır. Veri tanımı ve veri kısımları ikiye ayrılmıştır.

```
Table(id="data: authors")
{
 fields(pk=!nr)
 {
  field(id=!nr type="number");
  field(id=!firstname type="text");
  field(id=!lastname type="text");
  field(id=!email type="text");
 }
```

```
data
 {
  row {0, “Erdinç”, “Uzun”,
“erdinc@trakya.edu.tr”}
  row {1, “Yılmaz”, “Kılıçaslan”,
“yilmazk@trakya.edu.tr”}
  row {2, “Erdem”, “Uçar”,
“erdemu@trakya.edu.tr”}
 }
}
```

Örnek 6. SMEL Örneği

3. YÖNTEMLERİN KARŞILAŞTIRILMASI

CSV ve FSV yöntemlerinin hiyerarşik yapılarda yetersiz kaldığını daha önce söylemiştik. XML hiyerarşik yapılar içinde yeterli bir biçimleme dilidir. XML en büyük eksiklik olarak ilk göze çarpan noktada gereksiz etiket tekrarlarıdır. JSON ve YAML gibi yöntemler bu tekrarları azaltmaya çalışmışlardır. SMEL yöntemi ise tekrarları tamamen ortadan kaldırmasına rağmen okunabilirlik ve anlaşılabilirlik açısından daha yetersiz bir yaklaşımdır.

Yöntemlerin karşılaştırmasını yaparken XML, JSON, YAML ve SMEL gibi hiyerarşik yapıların gösterilebileceği yöntemleri göz önüne aldık. Bu yöntemleri bir tablo üzerindeki kayıtları kullanarak dosya boyutu, veri ve etiket kullanım yüzdeleri, dosya oluşturma süreleri, dosya erişim süreleri ve sıkıştırılabilirlik gibi beş kıstası göz önüne alarak inceledik.

Dosya boyutu açısından beklenildiği gibi en büyük dosya XML dosyası olmaktadır. Sırasıyla JSON %25’lik, YAML %41’lik, SMEL %56’lük kazanç sağlayarak daha küçük dosya boyutları elde etmişlerdir.

Etiket kullanımı açısından ise %73 oranında etiket kullanan bir XML dosyasında JSON %64, YAML %54, SMEL ise %27 oranında etiket kullanmaktadır.

Dosya boyutunun artması beklenildiği gibi dosya oluşum ve erişim sürelerini de etkilemiştir. Dosya oluşun süreleri XML göre JSON ile %15, YAML ile %30 ve SMEL ile %63 oranında daha kısa sürelerde oluşmaktadır. Dosyaya erişim süreleri (verilerin ayrıştırıp geri alınmasında) ise süre yaklaşık iki kat artmasına rağmen biçimleme dilleri arasında benzer oranlarla karşılaşılmaktadır.

Dosyaların sıkıştırılması ile oluşan yeni dosya boyutları arasında sırasıyla JSON %17’lik, YAML %28’lik, SMEL %54’lük kazanç sağlayarak daha küçük dosya boyutları elde etmişlerdir.

Not: Yapılan test ölçümleri SQL Server ile birlikte gelen Nortwind veritabanı içindeki orders tablosu kullanılarak yapılmıştır. Orders tablosu 10000 kayıt eklenmiş bu kayıtlar farklı biçimleme dillerine çevrilmiştir. Bu sonuçlara göre kıyaslar yapılmıştır.

4. TARTIŞMA ve SONUÇ

Yukarıdaki sonuçları göz önüne aldığımızda aslında XML’in zaman ve yer açısından etkin bir yöntem olmadığı göz önüne çıkmaktadır. Yine de XML şu an tüm dünyada kabul edilen bir standart halini almıştır. XML üzerine çıkarılan yeni teknikler sayesinde web dünyasında hızlı bir şekilde yayılmaya devam etmektedir. XML’in en büyük avantajı sıkıştırılabilirlik açısından uygun bir yapıya sahip olmasıdır. Fakat sıkıştırmanın da bir zaman kaybı olduğu unutulmamalıdır. Zaman ve yer etkinliğinin önemli olduğu uygulamalarda özellikle veri taşımak için XML yerine farklı bir alternatif tasarlanabilir ya da kullanılan etiket uzunluklarına dikkat edilmelidir.

KAYNAKLAR

[1] T. Carlier, (Some Modest Extensible Language) SMEL, http://users.pandora.be/tommycarlier/smel/index.htm

[2] D. Crockford, JavaScript Object Notation (JSON), http://www.crockford.com/JSON/index.html

[3] C. Evans, B. Ingerson, O. Ben-Kiki, Yet Another Markup Language (YAML), http://www.yaml.org/

[4] Glossary of Scientific Terms, http://www.harmsy.freeuk.com/glossary.html

[5] Gzip Web Site, www.gzip.org

[6] H. Liefke, D. Suciu, Xmill: an efficient compressor for XML data. In" Proc. of the ACM SIGMOD Int l Conf. on Management of Data, Dallas, Texas, United States, 2000, 153-164

[7] B. Iyer and D. Wilhite. Data compression support in databases. Proceedings of the 20th International Conference on Very Large Databases. Santiago, Chile, 1994, 695-704

[8] R. Lawrence, The Space Efficiency of XML, Information and Software Technology, 2004, 46 (11):753-759

[9] S. Radhakrishnan, Speed Web delivery with HTTP compression, Senior Architect, eBusiness, TATA Consultancy, 2003
Available http://www-128.ibm.com/developerworks/web/library/wa-httpcomp/