دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

گزارش مقدماتی درس پایگاه داده‌های پیشرفته

مدیریت داده‌های XML

عادل اردلان (۸۱۰۱۸۴۰۹۱)

a.ardalan@ece.ut.ac.ir

دکتر رهگذر

پاییز ۸۴

تکنولوژی XML

XML

زبان XML یک زبان نشانه‌گذاری برای توصیف یک مستند است که در درون خود اطلاعات ساختاری دارد. این اطلاعات ساختاری در واقع ترکیبی از داده‌های موجود در مستند و نقش آن قسمت از داده در درون مستند است. برای مثال قسمتی از یک مقاله عنوان آن است که می‌توان آن را با یک برچسب[1] مشخص کرد. در واقع یک زبان علامت‌گذاری یک روش برای بیان ساختارهای موجود در یک مستند است. در این میان ویژگیهای XML به نحوی است که یک راه استاندارد برای افزودن این چنین ساختارهایی را در مستندات امکان پذیر می‌سازد.

باید به این نکته توجه کرد که استاندارد XML [1] به هیچ وجه برای تعیین نوع نمایش داده به کار نمی‌رود و در این رابطه کاملا از HTML متمایز می‌باشد.

به کمک DTD می‌توانیم گرامری از نوع EBNF را برای مستندات XML مورد نظر خود تعریف کنیم.

برای هر مستند XML بخش‌ها و مفاهیمی مطرح می‌باشد:

۱. برچسب‌ها و متن‌ها

به طور کلی یک مستند XML تشکیل شده‌است از تعدادی برچسب و متن. برچسب‌ها به صورت زوج آغاز و خاتمه می‌یابند.

برچسب شروع: <course>

برچسب خاتمه: </course>

۲. المان‌ها

قسمتی که بین یک برچسب شروع و برچسب خاتمه معادل آن مشخص شده‌است یک المان نامیده می‌شود.

به المان‌هایی که در داخل یک المان قرار می‌گیرند و در واقع تشکیل دهنده‌های واقعی آن می‌باشند زیر المانهای آن المان می‌گویند.

شکل 1: المان و زیرالمان

[1] Tag

۳. ساختار برچسب‌های تودرتو[2]

برچسب‌های تودرتو برای بیان ساختارهای مختلف (به عنوان مثال برای بیان یک رکورد اطلاعاتی) به کار می‌روند. برای بیان لیستی از رکوردهای اطلاعاتی تعدادی برچسب‌ها مشابه به تعداد مورد نظر تکرار می‌شوند.

```
<person>… </person>
<person>… </person>
…
```

شکل 2 : نمایش لیستی از رکوردهای اطلاعاتی

۴. ساختار ترتیبی

در XML بیان فیلدهایی که به صورت مجموعه هستند به راحتی امکان‌پذیر می‌باشد، و دیگر محدودیتهای حاکم بر مدلهایی مانند مدل‌های رابطه‌ای بر آن حاکم نیست.

۵. المان‌های ویژه

هر مستند XML یک المان یکتا دارد که کل المان‌ها را در بر می‌گیرد که به آن المان ریشه می‌گویند. در مثال زیر اگر کل مستند مه به این صورت باشد، db یک المان ریشه است.

```
<db>
        <person> … </person>
        <person> … </person> ...
</db>
```

شکل 3 : المان ریشه

المانهایی که هیچ محتوای از نوع متن یا زیر المان نداشته باشد می‌توان به صورت یک المان تهی بیان کرد.

```
<giggle></giggle> → <giggle/>
<image img="picture.gif" />
```

شکل 4 :المان تهی

۶. خصیصه‌ها

برچسب شروع خود می‌تواند شامل خصیصه‌هایی باشد که مشخصه‌های قطعی آن را توضیح می‌دهد. این خصیصه‌ها برای ارجاع به یک برچسب نیز کاربرد دارد که برای این منظور باید در DTD معادل آن این موضوع مشخص شده باشد.

```
<person id = "011" pal="012">
        <name> George Bush</name>
</person>
<person id = "012" pal="011">
        <name> Saddam Hussein </name>
```

[2] Nested Tags

</person>

شکل 5 : خصیصه های المان

خصیصه‌های یک المان نمی‌تواند تکراری باشد و ترتیب آنها برای یک المان مهم نیست.

XML خوش‌فرم[3]

یک شیء متنی[4] را XML خوش‌فرم [1] می‌گویند اگر:

تمام متن مورد نظر درون یک عنصر که با برچسبی به نام برچسب ریشه[5] آغاز می‌شود می‌باشند و آن عنصر شامل صفر یا تعداد زیادی عنصر دیگر باشد.

"یعنی اگر یک عنصر الف داشته باشیم، حتما یک عنصر ب وجود دارد که آن شامل الف می‌شود که الف را فرزند ب و یا ب را والد الف می‌نامیم. در غیر این صورت الف نشانه ریشه است."

همچنین متن مورد نظر باید تمامی قواعد آن مستند که در فایل DTD مربوطه آمده را رعایت کند.

تمامی عنصرهای پارس‌شده در این مستند که به صورت مستقیم یا غیر مستقیم ارجاع داده می‌شوند نیز باید خوش‌فرم باشند.

مدل داده‌ای درختی

یک مدل داده‌ای درختی برای داده‌های XML [2] وجود دارد که زبانهای درخواست XML بر اساس این مدل کار می‌کنند. در این مدل متن یک مستند XML را با قواعد زیر به صورت یک درخت فرض می‌کنند:

۱. هر گره المان[6] چندین گره فرزند دارد که این فرزندان می‌توانند یا زیر المان و یا خصیصه[7] آن گره باشند.

۲. یک متن موجود در یک المان به صورت یک گره فرزند برای گره آن المان مدل می‌شود.

۳. فرزندان یک المان به همان ترتیبی که در فایل XML قرار دارند در درخت قرار می‌گیرند.

۴. هر گره المان و یا خصیصه به استثناء گره ریشه یک گره پدر دارد که از نوع المان است.

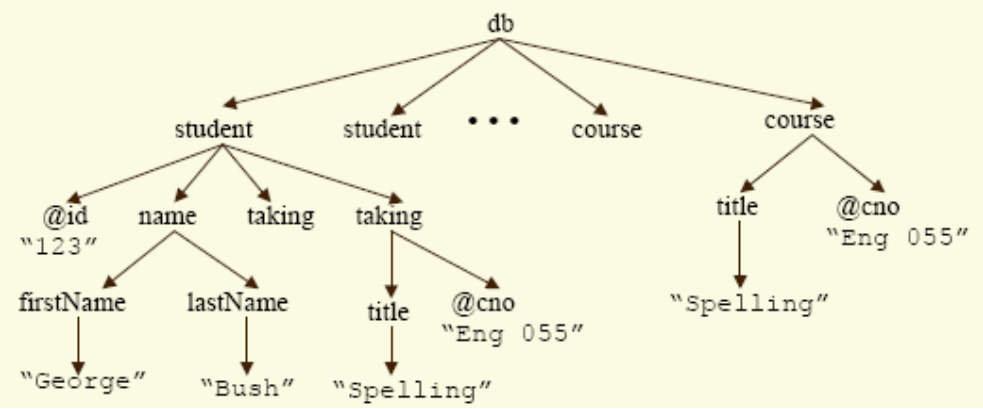
برای مثال مدل درختی مستند XML شکل(۶) در شکل(۷) نشان داده شده‌است.

```
<?xml version= ‘1.0’?>
<!DOCTYPE book PUBLIC “~wenfei/school.dtd”>
<?xml:stylesheet href=“school.xsl” type=“text/xsl”?>
<db> <!-- school database -->
  <student id = “123”>
    <name>
      <firstName>George</firstName> <lastName>Bush</lastName>
    </name>
    <taking> Eng 055 </taking>
    <GPA> 1.5 </GPA>
  </student>
```

[3] Well-Formed XML
[4] Textual Object
[5] Root Tag
[6] Element
[7] Attribute

```
  <course cno = “Eng 055”>
    <title> Spelling </title>
  </course>
</db>
```

شکل 6 : مستند XML مربوط به یک مدرسه با یک رکورد درس و یک رکورد دانش آموز



شکل 7 : مدل درختی برای XML مثال(۶)

در شکل(۷) برای مثال، گره course یک گره المان است. در این مثال گره @id یک گره خصیصه است که مقدار آن نیز در همان گره می‌آید. گره George نیز در اینجا یک گره برای مقادیر متنی است.

تعریف نوع مستند[8]

برای تعریف نوع مستندات XML دو روش وجود دارد.

۱. Document Type Definition (DTD)

هر مستند XML می‌تواند یک تعریف نوع مستند (DTD) داشته باشد، که در واقع بیان کننده شمای داده‌ای یک مجموعه از مستندات XML است.

```
<!ELEMENT db (book*)>
<!ELEMENT book (title, authors*, chapter*, ref*)>
<!ELEMENT chapter (text | section)*>
<!ELEMENT ref book>
<!ELEMENT title #PCDATA>
<!ELEMENT author #PCDATA>
<!ELEMENT section #PCDATA>
<!ELEMENT text #PCDATA>
```

شکل 8 : تعریف نوع مستند (DTD)

۱. تعریف نوع المان

برای هر نوع المان ما یک اعلام به صورت <!ELEMENT E P> داریم که P در اینجا یک عبارت منظم[9] است.

P ::= EMPTY | ANY | #PCDATA | E’ |

[8] DTD (Document Type Definition)
[9] Regular Expression

P1, P2 | P1 | P2 | P? | P+ | P*
E: element type
P1 , P2: concatenation
P1 | P2: disjunction
P?: optional
P+: one or more occurrences
P*: the Kleene closure

۲. اعلام[10] خصیصه

```
<!ATTLIST element_name  attribute-name attribute-type default-declaration>
```

۳. مشخص کردن ارجاعات

در DTD ما معادلهایی برای کلید اصلی و کلید خارجی داریم. در مثال زیر کلید اصلی(ID) و کلید خارجی (IDREF) دیده می‌شود.

```
<!ATTLIST book
        isbn ID #required>
<!ATTLIST ref
        to IDREFS #implied>
```

۲. XML Schema

این روش در جهت رفع مشکلات و نواقص روش قبل ایجاد شده است. از مزایای این روش نسبت به روش DTD می‌توان به موارد زیر اشاره کرد:

۱. قابلیت تعریف انواع داده‌ای توسط کاربر
۲. امکان محدود کردن یک field به یک نوع داده‌ای (تعریف نوع داده‌ای برای آن)
۳. امکان تعریف انواع داده‌ای جدید با محدود کردن انواع داده‌ای موجود
۴. امکان گسترش انواع داده‌ای با استفاده از نوعی *وراثت*
۵. فوق مجموعه‌ای از DTD
۶. قابلیت تعریف محدودیت‌های یکتایی و کلید خارجی
۷. امکان استفاده از namespace برای مقید کردن بخش‌های مختلف یک مستند به شماهای مختلف
۸. تعریف از طریق XML

مثالی از این روش در زیر آمده است.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:r="http://recipes.org"
      targetNamespace="http://recipes.org"
      elementFormDefault="qualified">

  <element name="collection">
    <complexType>
```

[10] Deceleration

```
    <sequence>
      <element name="description" type="r:anycontent"/>
      <element ref="r:recipe" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

<complexType name="anycontent" mixed="true">
  <sequence>
    <any minOccurs="0" maxOccurs="unbounded"
        processContents="skip" namespace="##other"/>
  </sequence>
</complexType>

<element name="recipe">
  <complexType>
    <sequence>
      <element name="title" type="string"/>
      <element ref="r:ingredient" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="r:preparation"/>
      <element name="comment" minOccurs="0" type="string"/>
      <element name="nutrition">
        <complexType>
          <attribute name="protein" type="r:nonNegativeDecimal" use="required"/>
          <attribute name="carbohydrates" type="r:nonNegativeDecimal" use="required"/>
          <attribute name="fat" type="r:nonNegativeDecimal" use="required"/>
          <attribute name="calories" type="r:nonNegativeDecimal" use="required"/>
          <attribute name="alcohol" type="r:nonNegativeDecimal" use="optional"/>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>

<element name="preparation">
  <complexType>
    <sequence>
      <element name="step" type="string" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

<element name="ingredient">
  <complexType>
    <sequence>
      <element ref="r:ingredient" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="r:preparation" minOccurs="0"/>
    </sequence>
    <attribute name="name" use="required"/>
    <attribute name="amount" use="optional">
      <simpleType>
        <union>
          <simpleType>
            <restriction base="string">
              <enumeration value="*"/>
            </restriction>
          </simpleType>
          <simpleType>
            <restriction base="r:nonNegativeDecimal"/>
          </simpleType>
        </union>
      </simpleType>
    </attribute>
    <attribute name="unit" use="optional"/>
  </complexType>
</element>

<simpleType name="nonNegativeDecimal">
  <restriction base="decimal">
    <minInclusive value="0"/>
```

```
  </restriction>
 </simpleType>
</schema>
```

استانداردهای جستجوی مستندات

[2] XPath

یک زبان برای آدرس‌دهی قسمتی از مستندات XML است. این زبان در سال ۱۹۹۹ توسط W3C پیشنهاد شده است که در سال ۲۰۰۲ استاندارد شد. این زبان تا حدودی نقش SQL را برای مستندات XML ایفا می‌کند و در حقیقت این عبارتها مسیر دسترسی به گره‌های خاصی را مشخص می‌کنند. زبان XPath مستند XML را به کمک DOM[11] مدل می‌کند. از کاربردهای مهم این زبان این است که به عنوان اجزای اصلی برخی ابزارهای دیگر (مانند: XPointer، XSLT، XQuery و ...) می‌باشد.

اجزای هر درخواست XPath را می‌توان به سه بخش زیر تقسیم کرد:

۱. پیمایش درخت

الف- روبه‌پایین

عباراتی هستند که برای حرکت رو به پایین در درخت و انتخاب گره‌های مورد نظر به کار می‌روند.

Syntax:
Q ::= . | l | @l | Q/Q | Q | Q | //Q | /Q | Q[q]
q ::= Q | Q op c | q and q | q or q | not(q)
.: self, the current node
l: either a tag (label) or *: wildcard that matches any label
@l: attribute
/, |: concatenation (child), union
//: descendants or self, “recursion”
[q]: qualifier (filter, predicate)
op: =, !=, <=, <, >, >=, >
c: constant
and, or, not(): conjunction, disjunction, negation

مثال:

/collection/dvd/cert

خروجی معادل درخواست داده شده در این مستند به صورت زیرخط‌دار مشخص شده‌است.

```
<?xml version=“1.0” encoding=“ISO-8859-1”?>
<collection>
        <dvd title=“Fawlty Towers”>
                <genre>comedy</genre>
                <year>1975</year>
                <cert>PG</cert>
        </dvd>
```

[11] Document Object Model

```
        <dvd title="Quadrophenia">
                <genre>cult</genre>
                <year>1979</year>
                <length>114</length>
        </dvd>
        <dvd title="Goldfinger">
                <year>1964</year>
                <cert>PG</cert>
                <length>105</length>
        </dvd>
</collection>
```

شکل 9 : مثال XPath

ب- روبه‌بالا

عباراتی هستند که برای حرکت رو به بالا و دسترسی به گره‌های اجداد به کار می‌روند.

Syntax:
Q ::= . . . | ../Q | ancestor ::Q | ancestor-or-self::Q
../: parent
ancestor, ancestor-or-self: recursion

ج- به پهلو

برای دسترسی به گره‌های هم‌سطح که به نوعی برادر به حساب می‌آیند.

Syntax:
Q ::= . . . | following-sibling ::Q | preceding-sibling::Q
following-sibling: the next sibling
preceding-sibling: the previous sibling
position function: e.g., //author[position( ) < 2]

۲. عبارت‌های رابطه‌ای و بولی

این عبارات همانند مسایل حاکم بر درخواستهای موجود در محیط رابطه‌ای برای مشخص کردن چندین شرط به صورت یک عبارت بولی نمایش داده می‌شوند.

۳. توابع

از این توابع برای کارهای پردازشی بر روی اطلاعات مانند پردازش رشته‌ها و انجام توابع اجتماعی به کار برده می‌شوند.

[2] XQuery

یک زبان قوی برای درخواستهای XML می‌باشد که در حال حاضر در W3C در مرحله بررسی می‌باشد گرچه به نوبه خود در مکان‌های متفاوتی استفاده شده و پیاده‌سازی شده است. زبان XQuery در واقع مرکب از XPath و یک سری امکانات دیگری -که به آن افزوده شده- است. یک نمونه نسبتا خوب پارسر XQuery به نام Galax در آزمایشگاه بل پیاده‌سازی شده است. این زبان بر مبنای زبان Quilt تعریف شده است.

```
for loop; $x: variable
where: condition test; selection
return: evaluate an expression and return its value
```

شکل 10: شکل کلی یک درخواست XQuery

امکانات عمومی این زبان شامل موارد زیر می‌شود:

۱. عبارات FLWR
۲. امکان مشاهده ترتیب عناصر
۳. گروه‌بندی و ایجاد مجموعه مقادیر

در اینجا برای تعریف نوع مستندی که در مثال(۸) آمده یک مثال جستجوی مستندات در شکل(۱۱) آمده است.

درخواست ما این است که: "می‌خواهیم عنوان و نویسنده‌های تمام کتابهای انتشارات Addison Wesley را که بعد از سال ۹۴ به چاپ رسیده‌اند را داشته باشیم."

```
<answer>{
        for $book in /bib/book
        where $book/@year > 1991 and $book/publisher='Addison-Wesley'
        return    <book>
                        <title> {$book/title } </title>,
                        for $author in $book/author return
                        <author> {$author } </author>
                </book>
}</answer>
```

مثال(): یک نمونه XQuery

```
        define function total (element part $part)
                returns element part {
                let $subparts :=
                for $s in $part/part return total($s)
                        return {
                                <part name="$part/@name"
                                cost="$part/@cost + sum($subparts/@cost)">
                        } </part>
                }
```

شکل 11: یک نمونه از XQuery برای تعریف تابع

زبان تعریف قالب XML (XML Stylesheet Language)

هدف از ایجاد این زبان نیل به یکی از اهداف ابتدایی XML یعنی جدا نمودن داده از قالب نمایش آن است. این زبان شامل مکانیزمی بنام **تبدیل تعریف قالب XML** (**XSL Transformation (XSLT)**) است که برای تبدیل داده‌های XML به قالب‌های دیگر مانند HTML مورد استفاده قرار می‌گیرد.

در زیر مثالی از چنین مستندی دیده می‌شود.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
            xmlns="http://www.w3.org/1999/xhtml">

  <xsl:template match="card[@type='simple']">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <title>business card</title><body>
        <xsl:apply-templates select="name"/>
        <xsl:apply-templates select="title"/>
        <xsl:apply-templates select="email"/>
        <xsl:apply-templates select="phone"/>
    </body></html>
  </xsl:template>

  <xsl:template match="card/name">
    <h1><xsl:value-of select="text()"/></h1>
  </xsl:template>

  <xsl:template match="email">
    <p>email: <a href="mailto:{text()}"><tt>
      <xsl:value-of select="text()"/>
    </tt></a></p>
  </xsl:template>

  ...
 </xsl:stylesheet>
```

در مورد مستند XML زیر:

```
<card type="simple">
   <name>John Doe</name>
   <title>CEO, Widget Inc.</title>
   <email>john.doe@widget.com</email>
   <phone>(202) 456-1414</phone>
 </card>
```

نتیجه بصورت ذیل خواهد بود:

```
<html xmlns="http://www.w3.org/1999/xhtml"><title>business card</title>
 <body><h1>John Doe</h1><h3><i>CEO, Widget Inc.</i></h3>
 <p>email: <a href="mailto:john.doe@widget.com">
                <tt>john.doe@widget.com</tt></a></p>
 <p>phone: (202) 456-1414</p>
 </body></html>
```

جبر درختی [7]

در پایگاه داده رابطه ای هشت نوع عملگر[۱۲]، اجتماع، اشتراک، تفاضل، ضرب کارتزین[۱۳]، گزینش[۱۴]، پرتـو[۱۵]، پیوند و تقسیم برروی جبر رابطه ای تعریف می شد که این عملگرها یک یا چنـد رابطـه را در ورودی مـی

[12] Operator

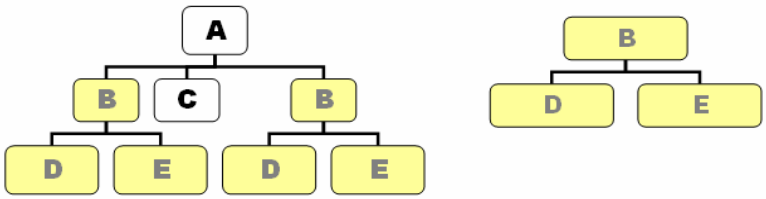
گرفتند و یک یا چند رابطه در خروجی ایجاد می کردند. در جبر در ختی عملگر های مشابه با جبـر رابطـه ای تعریف می شوند با این تفاوت که ورودی ها و خروجی های عملگرهـای جبـر درختـی مجموعـه ای از درخت های XML می باشند. علاوه بر این ،یک تفاوت عمده بین جبر رابطه ای و جبر درختی وجود دارد و آن مسئله وجود Heterogeneity در جبر درختی می باشد .

Heterogeneity

در جبر رابطه ای در هر رابطه بدلیل یکتا بودن نام صفات رابطه، با شماره صفت و یا نام صفت می توان به مقادیر آن صفت دسترسی پیدا کرد. اما در جبر درختی بدلیل ماهیت سندهای XML، زیر عناصر می توانند نامهای یکسان داشته باشند. به این خاصیت سندهای XML، Heterogeneity می گویند.

درخت الگو

تطبیق الگو به این معنی است که سیستم مدیریت پرس و جو در درخت های اصلی پایگـاه داده بـه دنبـال زیر درخت هایی می گردد که با درخت الگو[16] داده شده منطبق باشد. شکل زیر تطبیق الگو یـک درخـت الگو ورودی (الف) را با درخت اصلی (ب) نشان می دهند. زیر درخت هایی که گره های آنهـا بـا گـره تیـره مشخص شده اند با درخت الگو تطبیق دارند.



پایگاه داده XML

با افزایش کاربرد XML در محیط وب و زیاد شدن مستندات XML نیاز به یک سیستم بهینه و کامل برای ذخیره و بازیابی اطلاعات برای آن بیش از پیش احساس شد. محققان پایگاه داده بر آن شدند تا این نیاز را به نوعی پاسخ دهند و به دنبال آن، تحقیقاتی در این زمینه با پیش‌قدمی دانشگاه استانفورد آغاز شد.
برای این منظور دو دیدگاه مطرح شد. یک دیدگاه بر استفاده از سیستم‌های مدیریت پایگاه‌داده موجود تاکید می‌کرد و اعتقاد داشت که با توجه به بلوغ امکانات در این سیستم‌ها بهتر است از آن‌ها برای ذخیره و

[13] Product
[14] Select
[15] Project
[16] Pattern Tree

بازیابی بهینه مستندات XML استفاده شود.به این نوع از سیستم‌ها پایگاه داده با توانایی XML[17] می‌گویند. دیدگاه دوم این بود که با توجه به خاص بودن نیازها برای XML بهتر است یک سیستم کاملا بهینه شده برای XML طراحی و پیاده‌سازی شود که به آن پایگاه‌های بومی[18] XML می‌گویند.

سیستم‌ها پایگاه داده با توانایی XML

در این سیستم‌ها تلاش بر این است که تنها واسطهایی برای تبدیل مستندات XML به مدل داده‌ای آن پایگاه داده موجود تبدیل شود. همچنین درخواستهای مرتبط با آن (XQuery و ...) نیز به خوبی به درخواستهای آن زبان (SQL و ...) تبدیل شود و بر روی آن اجرا گردد و حاصل به صورت یک سند XML بازگردانده شود. در این روش دیگر نیازی به پیاده‌سازی سیستم‌های ذخیره‌سازی، کنترل هم روندی، بازسازی اطلاعات نیست و تلاش بر این است که از امکانات موجود حداکثر استفاده صورت پذیرد.

در این مورد ۳ استراتژی کلی مورد استفاده قرار می‌گیرد:

۱. ذخیره بصورت رشته
۲. نمایش درختی
۳. نگاشت بر روی رابطه‌ها

از مهمترین مسائلی که در این زمینه مطرح می‌شوند، می‌توان به موارد زیر اشاره کرد:

۱. انتخاب شمای رابطه‌ای
۲. پاسخگویی به پرسش‌ها
۳. بازسازی (مستندات از رابطه‌ها)

در این میان بهینه‌سازی پرسش‌ها از اهمیت خاصی برخوردار می‌شود. در این زمینه روش‌های زیر مورد توجه‌اند:

- توسعه یک جبر برای XQuery
- نگهداری داده‌های آماری برای پرسش‌های بر مبنای مسیر
- اعمال محدودیت بر روی داده‌ها

[17] XML Enabled Database
[18] Native

پایگاه‌های بومی XML

در این نوع از پایگاه داده هدف این است که یک سیستم کاملا منطبق بر نیازهای XML طراحی و تولید شود. البته در اکثر مواقع برای پیاده‌سازی بسیاری از اجزای این سیستم از ایده‌های به کار رفته در پایگاه-های موجود استفاده می‌شود.

در این زمینه ۲ روش عمومی مورد توجه قرار گیرد:

۱. ذخیره‌سازی در فایل مسطح
۲. ذخیره‌سازی در پایگاه داده‌های XML

مباحث مرتبط روز

۱. زبانهای بروزرسانی

از زبان‌های بروزرسانی XML می‌توان به XUpdate [8] اشاره کرد. این زبان از قالب XSLT برای تعریف خود استفاده می‌کند.

در این زبان یک تغییر بوسیله برچسب xupdate:modifications مشخص می‌شود. این برچسب می-تواند شامل برچسب‌های زیر باشد:

- xupdate:insert-before
- xupdate:insert-after
- xupdate:append
- xupdate:update
- xupdate:remove
- xupdate:rename
- xupdate:variable
- xupdate:value-of
- xupdate:if

بعنوان مثال دستورات زیر:

```
<xupdate:element name="address">
  <town>San Francisco</town>
</xupdate:element>
```

باعث ایجاد عنصر زیر می‌گردند:

```
<address>
  <town>San Francisco</town>
</address>
```

همچنین XQuery در استاندارد خود قابلیت بروزرسانی را پیش‌بینی کرده است که هنوز در مرحله پیشنهاد [9] قرار دارد. XEditor نیز از جمله تلاش‌هایی است که در این زمینه انجام شده است.

۲. ذخیره‌سازی داده‌ها در پایگاه رابطه‌ای

این روش‌ها به دو دسته کلی تقسیم می‌شوند:

۱. وابسته به DTD
   a. Basic Inlining
   b. Shared Inlining
   c. Hybrid Inlining

۲. مستقل از DTD
   a. Edge
   b. Edge-Value
   c. XRel
   d. Monet
   e. XParent
   f. OrdPath
   g. DLN
   h. …

منابع

[1] W3C XML Recommendation: http://www.w3.org/TR/2004/REC-xml-20040204/, Visited on 18 November 2005.

[2] W3C XQuery 1.0 and XPath 2.0 Data Model Candidate Recommendation: http://www.w3.org/TR/xpath-datamodel/, Visited on 18 November 2005.

[3] A. Silberschatz, H. F. Korth, S. Sudarshan, *Database System Concepts*. McGrow-Hill Higher Education, 4th edition, 2002.

[4] R. Ramakrishnan, J. Gehrke, *Database Management Systems*, McGraw-Hill Higher Education, 3rd Edition, 2003.

[۵] م. عمادی، *بستری کارآ برای مدیریت داده‌های XML*، گزارش سمینار کارشناسی ارشد، دانشگاه تهران، اسفند ۱۳۸۳.

[۶] ص. سلطان، *پردازش و بهینه سازی پرس و جو در زبان XML*، گزارش پیشرفت پروژه کارشناسی ارشد، دانشگاه تهران، شهریور ۱۳۸۴.

[7] H. V. Jagadish, Laks V. S. Lakshmanan, Divesh Srivastava, Keith Thompson, *TAX: A Tree Algebra for XML*, 8th International Workshop on Database Programming Languages, DBLP 2001.

[8] XUpdate Working Draft: http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html, Visited on 16 November 2005.

[9] XQuery Update Facility Requirements: http://www.w3.org/TR/2005/WD-xquery-update-requirements-20050603/, Visited on 16 November 2005.

[10] XEditor: http://www.openhealth.org/editor/, Visited on 16 November 2005.