Chapter 23: XML

Database System Concepts, 6th Ed.

XML

XML 데이터 구조

XML 문서 스키마

질의 및 변환

Introduction

XML: Extensible Markup Language

WWW Consortium (W3C) 에서 정의함

SGML (Standard Generalized Markup Language)에서 파생되었으나, SGML 보다 간단함.

문서에서 각 섹션에 대한 정보를 기술하기 위하여 태그(tags) 구조를 사용

E.g. <title> XML </title> <slide> Introduction …</slide>

HTML과 달리 확장 가능함.

새로운 태그를 부가할 수 있으며, 태그의 출력 형태를 별도로 기술할 수 있음.

XML Introduction (Cont.)

XML의 태그 정의, 태그의 네스팅 구조 등은 데이터 교환에 매우 유용한 기능을 제공한다.

따라서 XML은 HTML의 대용으로 사용되기 보다는 데이터 교환 응용 분야에 주로 사용된다.

태그는 자체 문서화 (self-documenting) 기능을 제공한다.

E.g.

```
<university>
    <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <course>
        <course_id> CS-101 </course_id>
        <title> Intro. to Computer Science </title>
        <dept_name> Comp. Sci </dept_name>
        <credits> 4 </credits>
    </course>
</university>
```

XML:동기

데이터 교환은 네트워크화된 오늘날의 세계에서 매우 중요한 기능을 담당한다.

Examples:

- Banking: funds transfer
- Order processing (especially inter-company orders)
- Scientific data
  - Chemistry: ChemML, …
  - Genetics: BSML (Bio-Sequence Markup Language), …

과거의 종이 문서 교환에 의한 정보 흐름은 전자 문서 교환으로 바뀌었다.

각 응용 분야는 정보 표현을 위한 자체적 표준안을 가지고 있다.

XML은 차세대 데이터 교환 표준으로 자리잡고 있다.

XML Motivation (Cont.)

XML 기반의 정보 표현/전달 방법

XML 구문(syntax) 기술언어

- DTD (Document Type Descriptors)
- XML Schema

추가적인 의미론(semantics)을 위한 추가적 텍스트 기술

XML에서는 필요에 따라 새로운 태그 정의가 가능하다

그러나 태그 정의는 DTD 기술과 일치하여야 한다.

XML 문서/데이터 처리를 위한 많은 파싱/브라우징/질의 도구가 존재한다.

릴레이션 데이터와의 비교

비효율성: 실제로 스키마 정보를 표현하고 있는 태그 정보가 반복되어 나타난다

데이터 교환 포맷으로서 릴레이션 튜플 구조에 비하여 우수한다.

- XML 데이터는 태그를 이용한 자체 문서화 (self-documenting) 기능을 제공한다.
- 유연성 있는 포맷 구조: 새로운 태구 추가가 가능하다.
- 네스팅 구조가 허용된다.
- 데이터베이스 시스템 분야뿐 아니라 다양한 응용 분야에 사용된다.

XML 데이터 구조

태그 **(Tag)**: 섹션 데이터를 위한 레이블 정보

엘리먼트 **(Element)**: 스타트 태그 *<tagname>* 에서 엔딩 태그 *</tagname>*까지의 섹션 데이터

엘리먼트는 다음과 같은 적법한 네스팅 구조를 가져야 한다.

Proper nesting

- <course> … <title> …. </title> </course>

Improper nesting

- <course> … <title> …. </course> </title>

스타트 태그와 엔딩 태그가 구조적으로 맞아야 한다.

모든 문서는 단 하나의 최상위 엘리먼트를 갖는다.

Example of Nested Elements

```
<purchase_order>
    <identifier> P-101 </identifier>
    <purchaser>  …. </purchaser>
    <itemlist>
       <item>
          <identifier> RS1 </identifier>
          <description> Atom powered rocket sled </description>
          <quantity> 2 </quantity>
          <price> 199.95 </price>
       </item>
       <item>
          <identifier> SG2 </identifier>
          <description> Superb glue </description>
          <quantity> 1 </quantity>
          <unit-of-measure> liter </unit-of-measure>
          <price> 29.95 </price>
       </item>
     </itemlist>
  </purchase_order>
```

XML 데이터 구조 (계속)

XML 에서는 다음과 같이 텍스트와 서브 엘리먼트를 같이 기술할 수 있다.

Example:

```
<course>
    This course is being offered for the first time in 2009.
    <course id> BIO-399 </course id>
    <title> Computational Biology </title>
    <dept name> Biology </dept name>
    <credits> 3 </credits>
</course>
```

이와 같은 기능은 문서 마크업을 위한 기능으로는 유용하지만, 단순 데이터 기술 도구로 사용되는 경우 사용하지 않는 것이 좋다.

애트리뷰트 (Attributes)

엘리먼트는 애트리뷰트를 가질 수 있다

```
<course course_id= “CS-101”>
    <title> Intro. to Computer Science</title>
    <dept name> Comp. Sci. </dept name>
    <credits> 4 </credits>
</course>
```

애트리뷰트는 엘리먼트의 스타트 태그 안에 *name=value* 의 형태로 기술된다.

각 애트리뷰트는 여러 개의 애트리뷰트를 가질 수 있으나, 각 애트리뷰트 이름은 유일하여야 한다.

```
<course  course_id = “CS-101”  credits=“4”>
```

애트리뷰트 vs. 서브 엘리먼트

서브 엘리먼트와 애트리뷰트의 차이점

하나의 문서에서 애트리뷰트는 일종의 마크업 기능을 가지나, 서브 엘리먼트는 기본 문서의 일부분을 나타낸다.

그러나, 그 차이점은 불명확하며 혼동되어 사용된다.

- 임의의 정보는 다음과 같이 두 가지 방식으로 기술될 수 있다.
  - <course course_id= “CS-101”> … </course>
  - <course>
    <course_id>CS-101</course_id> …
    </course>

권고사항: 엘리먼트의 식별자 (ID)를 나타내고자 하는 경우에는 애트리뷰트를 사용하고 내용을 나타내고자 하는 경우에는 서브 엘리먼트르 사용하라.

네임스페이스 (Namespaces)

여러 기관 사이에 XML 데이터를 교환하는 경우를 가정하자

각 기관들은 동일한 테그 명을 각각 다른 의미로 사용하고 있을 수 있으므로, 데이터 교환 시 혼란이 야기될 수 있다

각 엘리멘트 명을 유니크한 스트링으로 표현하여 혼란의 야기를 방지할 수 있다

Better solution: unique-name:element-name의 표현법을 사용하라

XML 네임스페이스를 이용하여 길이가 긴 unique name 사용하는 것을 피할 수 있다.

```
<university xmlns:yale="http://www.yale.edu">
        …
        <yale:course>
            <yale:course_id> CS-101 </yale:course_id>
            <yale:title> Intro. to Computer Science</yale:title>
            <yale:dept_name> Comp. Sci. </yale:dept_name>
            <yale:credits> 4 </yale:credits>
        </yale:course>
        …
</university>
```

More on XML Syntax

서브 엘리먼트 혹은 텍스트를 내부에 포함하지 않는 엘리먼트를 기술하는 경우, 다음과 같이 앤드 테그를 생략하고 스타트 테그 마지막에 /> 를 붙여 약식 기술할 수 있다

```
<course  course_id="CS-101" Title="Intro. To Computer Science"
         dept_name = "Comp. Sci." credits="4"  />
```

태그를 서브 엘리먼트로 해석하지 않고 단순한 태그를 포함하는 스트링 데이터로 저장하기 위하여 다음과 같이 CDATA를 사용한다.

```
<![CDATA[<course> … </course>]]>
```

여기에서 <course> 와 </course> 는 단순 스트링으로 취급된다.

CDATA 는 "character data"를 의미한다

XML 문서 스키마

데이터베이스 스키마는 어떤 정보가 저장될 수 있는지, 그리고 저장되는 값의 데이터 타입이 무엇인지의 정보를 포함한다

XML 문서는 관련 스키마를 반드시 가질 것을 요구하지는 않는다

그러나 XML 데이터 교환에 있어, 스키마 정보는 매우 중요한 역할을 담당한다

스키마 정보가 없다면 데이터를 전송 받은 사이트에서는 정보를 자동으로 해석할 수 없다

XML schema를 기술하기 위한 두 가지 기법

Document Type Definition (DTD)

- 널리 사용됨

XML Schema

- 새로운 형태로 점차적으로 사용이 증가하고 있음

Document Type Definition (DTD)

XML 문서의 타입 정보는 DTD를 사용하여 기술할 수 있다

DTD 는 XML 데이터의 구조에 대하여 기술한다

- 어떤 엘리먼트가 올 수 있는지를 기술
- 엘리먼트는 어떤 애트리뷰트를 가져야 하는 지 혹은 가질 수 있는지를 기술
- 각 엘리먼트 내부에 어떤 서브 엘리먼트가 올 수 있는지 혹은 와야 하는지, 그리고 몇 번 나와야 하는지의 정보를 기술

DTD는 구체적 데이터 타입을 기술하지 않는다

- XML에서 모든 값은 스트링으로 취급된다

DTD syntax

- <!ELEMENT element (subelements-specification) >
- <!ATTLIST element (attributes) >

DTD에서의 엘리먼트 기술

- <!ELEMENT element (subelements-specification) >

서브 엘리먼트는 다음과 같이 기술된다

엘리먼트 이름 (names of elements) 혹은

#PCDATA (parsed character data), i.e., character strings

EMPTY (no subelements) or ANY (anything can be a subelement)

Example

<! ELEMENT department (dept_name building, budget)>
<! ELEMENT dept_name (#PCDATA)>
<! ELEMENT budget (#PCDATA)>

서브 엘리먼트는 다음과 같은 정규화 표현식 (regular expressions)으로 기술될 수 있다

<!ELEMENT university ((department | course | instructor | teaches)+)>

- Notation:
  - “|” - alternatives
  - “+” - 1 or more occurrences
  - “*” - 0 or more occurrences

University DTD

```
<!DOCTYPE  university [
    <!ELEMENT university (
    (department|course|instructor|teaches)+)>
    <!ELEMENT department ( dept name, building, budget)>
    <!ELEMENT course ( course id, title, dept name, credits)>
    <!ELEMENT instructor (IID, name, dept name, salary)>
    <!ELEMENT teaches (IID, course id)>
    <!ELEMENT dept name( #PCDATA )>
    <!ELEMENT building( #PCDATA )>
    <!ELEMENT budget( #PCDATA )>
    <!ELEMENT course id ( #PCDATA )>
    <!ELEMENT title ( #PCDATA )>
    <!ELEMENT credits( #PCDATA )>
    <!ELEMENT IID( #PCDATA )>
    <!ELEMENT name( #PCDATA )>
    <!ELEMENT salary( #PCDATA )>
]>
```

```
<university>
    <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department>
        <dept_name> Biology </dept_name>
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course>
        <course_id> CS-101 </course_id>
        <title> Intro. to Computer Science </title>
        <dept_name> Comp. Sci. </dept_name>
        <credits> 4 </credits>
    </course>
    <course>
        <course_id> BIO-301 </course_id>
        <title> Genetics </title>
        <dept_name> Biology </dept_name>
        <credits> 4 </credits>
    </course>
    <instructor>
        <IID> 10101 </IID>
        <name> Srinivasan </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 65000 </salary>
    </instructor>
    <instructor>
        <IID> 83821 </IID>
        <name> Brandt </name>
        <dept_name> Comp. Sci. </dept_name>
        <salary> 92000 </salary>
    </instructor>
    <teaches>
        <IID> 10101 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 83821 </IID>
        <course_id> CS-101 </course_id>
    </teaches>
    <teaches>
        <IID> 76766 </IID>
        <course_id> BIO-301 </course_id>
    </teaches>
</university>
```

```
<!DOCTYPE university [
    <!ELEMENT university (
    (department|course|instructor|teaches)+)>
    <!ELEMENT department ( dept name, building, budget)>
    <!ELEMENT course ( course id, title, dept name, credits)>
    <!ELEMENT instructor (IID, name, dept name, salary)>
    <!ELEMENT teaches (IID, course id)>
    <!ELEMENT dept name( #PCDATA )>
    <!ELEMENT building( #PCDATA )>
    <!ELEMENT budget( #PCDATA )>
    <!ELEMENT course id ( #PCDATA )>
    <!ELEMENT title ( #PCDATA )>
    <!ELEMENT credits( #PCDATA )>
    <!ELEMENT IID( #PCDATA )>
    <!ELEMENT name( #PCDATA )>
    <!ELEMENT salary( #PCDATA )>
]>
```

DTD에서의 애트리뷰트 기술

- <!ATTLIST element (attributes) >

애트리뷰트 기술:

Name

Type of attribute

- CDATA
- ID (identifier) 혹은 IDREF (ID reference) 혹은 IDREFS (multiple IDREFs)
  - more on this later

다음 중 하나의 조건을 가짐

- mandatory (#REQUIRED)
- has a default value (value),
- or neither (#IMPLIED): the attribute does not have to be included.

Examples

<!ATTLIST course course_id CDATA #REQUIRED>, or

```
<!ATTLIST course
    course_id     ID         #REQUIRED
    dept_name   IDREF    #REQUIRED
    instructors    IDREFS #IMPLIED   >
```

IDs 와 IDREFs

엘리먼트는 ID 타입을 갖는 애트리뷰트를 최대 하나까지만 가질 수 있다

하나의 XML 문서에서 각 엘리먼트의 ID 애트리뷰트 값은 서로 달라야 한다

즉, ID 애트리뷰트 값은 object 식별자의 역할을 담당한다

IDREF 타입의 애트리뷰트는 같은 문서 내에 있는 임의의 엘리먼트의 ID 값을 해당 값으로 갖는다.

IDREFS 타입의 애트리뷰트는 임의의 개수 (0 혹은 그 이상)의 ID 값을 갖는다. 이 때 각 ID 값은 동일 문서 내에 있는 엘리먼트의 ID 값이어야 한다.

University DTD with Attributes

University DTD with ID and IDREF attribute types.

```
<!DOCTYPE university-3 [
    <!ELEMENT university-3 ( (department|course|instructor)+)>
    <!ELEMENT department ( building, budget )>
    <!ATTLIST department
        dept_name ID #REQUIRED >
    <!ELEMENT course (title, credits )>
    <!ATTLIST course
        course_id ID #REQUIRED
        dept_name IDREF #REQUIRED
        instructors IDREFS #IMPLIED >
    <!ELEMENT instructor ( name, salary )>
    <!ATTLIST instructor
        IID ID #REQUIRED
        dept_name IDREF #REQUIRED >
    · · · declarations for title, credits, building,
        budget, name and salary · · ·
]>
```

```
<university-3>
    <department dept_name="Comp. Sci.">
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department dept_name="Biogoly">
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course course_id="CS-101" dept_name="Comp. Sci."
                               instructors="10101 83821">
        <title> Intro. to Computer Science </title>
        <credits> 4 </credits>
    </course>
    <course course_id="BIO-301" dept_name="Biology"
                               instructors="76766">
        <title> Genetics </title>
        <credits> 4 </credits>
    </course>
    <instructor IID="10101" dept_name="Comp. Sci.">
        <name> Srinivasan </name>
        <salary> 65000 </salary>
    </Instructor>
    <instructor IID="83821" dept_name="Comp. Sci.">
        <name> Brandt </name>
        <salary> 72000 </salary>
    </Instructor>
    <instructor IID="76766" dept_name="Biology">
        <name> Crick </name>
        <salary> 72000 </salary>
    </Instructor>
</university-3>
```

University DTD with ID and IDREF attribute types.

```
<!DOCTYPE university-3 [
    <!ELEMENT university-3 ( (department|course|instructor)+)>
    <!ELEMENT department ( building, budget )>
    <!ATTLIST department
        dept_name ID #REQUIRED >
    <!ELEMENT course (title, credits )>
    <!ATTLIST course
        course_id ID #REQUIRED
        dept_name IDREF #REQUIRED
        instructors IDREFS #IMPLIED >
    <!ELEMENT instructor ( name, salary )>
    <!ATTLIST instructor
        IID ID #REQUIRED
        dept_name IDREF #REQUIRED >
    . . . declarations for title, credits, building,
        budget, name and salary . . .
]>
```

DTD의 제약점

엘리먼트 혹은 애트리뷰트의 구체적 타입이 기술되지 못한다

모든 값은 string으로 취급되어 integer, real 등의 값을 가지지 못한다

서브 엘리먼트의 순서가 없는 집합 (unordered set) 개념을 표현하기 어렵다

순서의 개념은 데이터베이스에서는 일반적으로 사용하지 않는다 (즉, XML이 사용되는 문서 레이아웃 환경과 다르다)

(A | B)* 표현이 순서없는 집합을 표현한다

- 그러나 A 와 B가 한번만 나타나는 것을 보장할 수 없다

IDs 와 IDREFs 역시 구체적 타입 정보를 가지지 못한다

앞의 예에서 course의 *instructors* 애트리뷰트는 다른 course 엘리먼트를 참조할 수도 있으며, 이 것은 의미없는 잘못된 사용법이 될 것이다.

- *instructors* 애트리뷰트는 원래 instructor 엘리먼트만을 참조할 수 있도록 제한되어야 한다

```
<university-3>
    <department dept_name="Comp. Sci.">
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <department dept_name="Biogoly">
        <building> Watson </building>
        <budget> 90000 </budget>
    </department>
    <course course_id="CS-101" dept_name="Comp. Sci."
                               instructors="10101 83821">
        <title> Intro. to Computer Science </title>
        <credits> 4 </credits>
    </course>
    <course course_id="BIO-301" dept_name="Biology"
                               instructors="76766">
        <title> Genetics </title>
        <credits> 4 </credits>
    </course>
```

```
    <instructor IID="10101" dept_name="Comp. Sci.">
        <name> Srinivasan </name>
        <salary> 65000 </salary>
    </Instructor>
    <instructor IID="83821" dept_name="Comp. Sci.">
        <name> Brandt </name>
        <salary> 72000 </salary>
    </Instructor>
    <instructor IID="76766" dept_name="Biology">
        <name> Crick </name>
        <salary> 72000 </salary>
    </Instructor>
</university-3>
```

XML Schema

XML Schema는 DTD의 단점을 보완하여 다음과 같은 기능을 제공하는
의 스키마 언어이다.

타입 정의

- E.g. integer, string, etc
- 또한, min/max 값의 제한이 가능함

사용자 정의, 복합 타입 정의

그 외의 다양한 기능 지원

- Uniqueness, foreign key constraints, inheritance

XML Schema는 XML 문서와 같은 구문을 갖는다.

보다 표준화된 표현 형식을 따른다.

XML Scheme에서 namespaces 개념을 이용 가능하다.

그러나: XML Schema는 DTD와 비교하여 훨씬 복잡하다.

XML Schema Version of Univ. DTD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="university" type="universityType" />
<xs:element name="department">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="dept name" type="xs:string"/>
            <xs:element name="building" type="xs:string"/>
            <xs:element name="budget" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
....
<xs:element name="instructor">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="IID" type="xs:string"/>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="dept name" type="xs:string"/>
            <xs:element name="salary" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
... Contd.
```

XML Schema Version of Univ. DTD (Cont.)

```
....
<xs:complexType name="UniversityType">
   <xs:sequence>
      <xs:element ref="department" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="course" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="instructor" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="teaches" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
</xs:schema>
```

여기에서 사용된 "xs:" 는 이 예제에 국한된 것이고, 임의의 다른 네임 스페이스 선택이 가능하다.

"university" 엘리먼트는 "universityType"의 별도의 타입으로 정의되어 있다.

xs:complexType 정의를 이용하여 name이 "UniversityType"인 복합 구조 타입을 정의하고 있음.

More features of XML Schema

애트리뷰트는 xs:attribute 의 태그 형태로 기술된다. 예를 들어 department 엘리먼트의 정의부에 다음과 같이 추가하여 dept_name을 애트리뷰트로 지정할 수 있다.

```
<xs:attribute name = “dept_name”/>
```

여기에 use = “required” 라는 형태의 use 애트리뷰트가 추가되어 기술되면, 애트리뷰트 값이 반드시 와야 한다는 의미를 나타낸다.

Key 제약조건: 루트가 되는 university 엘리먼트 아래의 department 엘리먼트의 dept_name 필드를 키로 지정하는 예제:

```
<xs:key name = “deptKey”>
        <xs:selector xpath = “/university/department”/>
        <xs:field xpath = “dept_name”/>
</xs:key>
```

Foreign key 제약조건: course 엘리먼트에서 department 엘리먼트로 외래키를 지정하는 예제. 여기에서 refer 애트리뷰트는 참조되는 키의 이름이고, field는 참조하는 애트리뷰트의 식별자를 나타낸다 :

```
<xs:keyref name = “courseDeptFKey” refer=“deptKey”>
        <xs:selector xpath = “/university/course”/>
        <xs:field xpath = “dept_name”/>
</xs:keyref>
```

Thank You