# Web Technologies and Data Storage 1

## ⚠Structure of XML Data

Definition: XML: Extensible Markup Language [可扩展的标记语言]


Documents have tags giving extra information about sections of the document

e.g. <title> XML </title> <slide> Introduction ···</slide>

Extensible, unlike HTML. Users can define and add new tags, and separately specify how the tag should be handled for display

XML 简介 – 什么是 XML? | 菜鸟教程 (runoob.com)

Tag structures make XML a great way to exchange data, and make data (relatively) self-documenting

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>
```

Each application area has its own set of standards for representing information and XML has been the basis for all new generation data interchange formats

XML compared with Relational Data are inefficient (tags which represent schema information are repeated) but better as a data exchange format.

## ⚠XML Document Schema [XML 文档架构]

Tag: label for a section of data

Element: section of data beginning with <tagname> and ending with matching </tagname>

Elements must be properly nested

Proper nesting:

<course> ··· <title> ···. </title> </course> √

Improper nesting:

<course> ··· <title> ···. </course> </title> ×

Formally: every start tag must have a unique matching end tag, that is in the context of the same parent Element and every document must have a single top-level element

Elements can have attributes

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

Attributes are specified by name=value pairs inside the starting tag of an element

An element may have several attributes, but each attribute name can only occur once

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

**▲Distinction between subelement and attribute:**

In the context of documents, attributes are part of markup, while subelement contents are part of the basic document contents

Same information can be represented in both ways:

```
<course course_id= “CS-101”> ··· </course>
<course> <course_id>CS-101</course_id> ···</course>
```

Suggestion: use attributes for identifiers of elements, and use subelements for contents

[使用属性作为元素的标识符，使用子元素作为内容]

**▲Namespace**

XML data has to be exchanged between organisations. Same tag name may have different meaning in different organisations, causing confusion on exchanged documents. Specifying a unique string as an element name avoids confusion.

Using XML Namespaces:

```
<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>
```

**▲XML Document Schema**

Database schemas constrain what information can be stored and the data types of stored values.

Although XML documents are not required to have an associated schema but schemas are important for XML data exchange. Otherwise, a site cannot automatically interpret data received from another site.

[尽管 XML 文档不需要有关联的架构，但对于数据交换非常重要，否则一个站点无法自动解释从另一个站点接收到的数据]

Two mechanisms for specifying XML schema:

**Document Type Definition (DTD)**

①DTD constraints structure of XML data

- What elements can occur
- What attributes can/must an element have
- What subelements can/must occur inside each element, and how many times

②DTD does not constrain data types

- All values represented as strings in XML

③DTD syntax

<!ELEMENT element (subelements-specification) >

<!ATTLIST element (attributes) >

④Subelements can be specified as:

names of elements followed by #PCDATA (parsed character data) or EMPTY (no subelements) or ANY (anything can be a subelement)

Example:

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

<! ELEMENT dept_name (#PCDATA)>

<! ELEMENT budget (#PCDATA)>

Subelement specification may have regular expressions

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

Notation:

"|" - alternatives

"+" - 1 or more occurrences

"*" - 0 or more occurrences

⑤Limitations of DTDs:

- No typing of text elements and attributes so all values are strings, no integers, reals…
- Difficult to specify unordered sets of subelements [难以指定无序的子元素集]
  - Order is usually irrelevant in databases
  - (A | B)* allows specification of an unordered set, but cannot ensure that each of A and B occurs only once
- IDs and IDREFs are untyped
- instructors attribute should ideally be constrained to refer to instructor elements

**XML Schema**

Typing of values: integer, string, …, constraints on min/max values

User-defined, complex types

Many more features, including uniqueness and foreign key constraints, inheritance [增加了主外键约束]

XML Scheme is integrated with namespaces [XML Scheme 与名称空间集成]

XML Schema is itself specified in XML syntax, unlike DTDs

更标准的表示，但冗长

### ⚠Querying and Transformation

Query and transformation of XML data are closely related, and handled by the same set of tools. Standard XML querying/transformation languages:

XPath

Simple language consisting of path expressions

XSLT

Simple language designed for translation from XML to XML and XML to HTML

XQuery

An XML query language with a rich set of features

### ⚠Application Program Interfaces to XML

There are two standard application programming interfaces to XML data:

SAX (Simple API for XML)

Based on parser model, user provides event handlers for parsing events

E.g. start of element, end of element

DOM (Document Object Model)

XML data is parsed into a tree representation

Variety of functions provided for traversing the DOM tree

E.g. Java DOM API provides Node class with methods

getParentNode( ), getFirstChild( ), getNextSibling( )

getAttribute( ), getData( ) (for text node)

getElementsByTagName( ), ···

Also provides functions for updating DOM tree

### ⚠Storage of XML Data

Non-relational data stores:

Flat files [平面文件]

Natural for storing XML

But has all problems, e.g. no concurrency, no recovery

XML database [XML 数据库]

Database built specifically for storing XML data, supporting DOM model and declarative querying

Relational databases:

String Representation [字符串表示]

•Store each top-level element as a string field of a tuple in a relational database

•Store values of subelements/attributes to be indexed as extra fields of the relation and build Indices on these fields (Some database systems support function indices, which use the result of a function as the key value)

Benefits:

Can store any XML data even without DTD

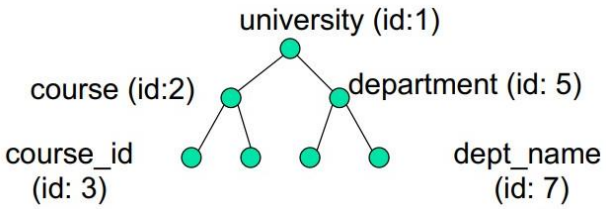
Allows fast access to individual elements.

Drawbacks:

Need to parse strings to access values inside the elements

Parsing is slow

Tree Representation [树表示]




nodes(id, parent_id, type, label, value)

Benefit:

Can store any XML data, even without DTD

Drawbacks:

Data is broken up into too many pieces, increasing space overheads

Even simple queries require a large number of joins, which can be slow

Map to relations [映射到关系]

Id attribute, relation attribute, parent_id attribute

All subelements that occur only once can become relation attributes

Subelements that can occur multiple times represented in a separate table

Data must be translated into relational form

Advantage: uses mature database systems

Disadvantages: overhead of translating data and queries

### ⚠XML Applications

Storing and exchanging data with complex structures [用复杂结构存储和交换数据]

Standard for data exchange for Web services [用于 Web 服务的数据交换标准]

Data mediation [网格数据仲裁服务]

A Web service is a site providing a collection of SOAP procedures [简单对象存取协议]

The Simple Object Access Protocol (SOAP) standard:

Invocation of procedures across applications with distinct databases

XML used to represent procedure input and output

[具有跨不同数据库的应用程序调用过程，XML 用于表示过程输入和输出]

Web Services @ W3C