Chapter 13

XML: Extensible Markup Language

- Internet applications provide Web interfaces to databases (data sources)
- Three-tier architecture
  Client
  V
  Application Programs Webserver
  V
  Database Server

- HTML common language for Web pages—Not suitable for structure data extracted from DB
- XML is used to represent structured data and exchange data on the Web for self-describing documents
- HTML, static Web pages; when they describe database data, they are called dynamic Web pages

1. Structured, Semi-structured and Unstructured data

- Information in databases is a structured data (employee, company, etc), the database checks to ensure that all data follows the structure and constraints specified in the schema
- some attributes may be shared among various entities, other attributes may exist only in few entities

Semi-structured data; additional attributes can be introduced later

Self-describing data as schema can change

Example: Collect a list of biographical references related to a certain research project
- Books
- Technical reports
- Research articles in Journals or Conferences

Each have different attributes and information; one reference has all information, others have partial; some references may occur in the future Web, tutorials, etc..

Semi-structured data can be represented using a directed graph.
Figure 13.1 Representing semistructured data as a graph.
Two main differences between semi-structured and object model:

1. Schema intermixed with objects
2. No requirements for a predefined schema

**Unstructured Data:** Limited indication of type of data
Example: text document; information is embedded in it

**Figure 13.2** Part of an HTML document representing unstructured data.

```html
<html>
<head>
...
</head>
<body>
<h1>List of company projects and the employees in each project</h1>
<h2>The ProductX project</h2>
<table width="100%" border=0 cellpadding=0 cellspacing=0>
  <tr>
    <td width="50%"><font size="2" face="Arial">John Smith</font></td>
    <td>32.5 hours per week</td>
  </tr>
  <tr>
    <td width="50%"><font size="2" face="Arial">Joyce English</font></td>
    <td>20.0 hours per week</td>
  </tr>
</table>
<h2>The ProductY project</h2>
<table width="100%" border=0 cellpadding=0 cellspacing=0>
  <tr>
    <td width="50%"><font size="2" face="Arial">John Smith</font></td>
    <td>7.5 hours per week</td>
  </tr>
  <tr>
    <td width="50%"><font size="2" face="Arial">Joyce English</font></td>
    <td>20.0 hours per week</td>
  </tr>
  <tr>
    <td width="50%"><font size="2" face="Arial">Franklin Wong</font></td>
    <td>10.0 hours per week</td>
  </tr>
</table>
...
</body>
</html>

HTML document has tags, which specify how to display a document, they also specify structure of a document.

HTML uses large number of predefined tags:

- Document header
- Body
- Heading levels
- Table
- Attributes (tags have attributes)

(static and dynamic html)

**XML Hierarchical Data Model (Tree)**

The basic object is the XML document. Two main concepts are used to construct XML document; elements, attributes (not same as DB)

- XML tag names are defined to describe the meaning of the data elements in the document, rather than how the text is to be described
- XML tag (element) names can be defined in another document known as schema document, which can be used for multiple programs and users
Figure 13.3  A complex XML element called <Projects>.

```xml
<?xml version="1.0" standalone="yes"?>
<Projects>
  <Project>
    <Name>ProductX</Name>
    <Number>1</Number>
    <Location>Bellaire</Location>
    <Dept_no>5</Dept_no>
    <Worker>
      <Ssn>123456789</Ssn>
      <Last_name>Smith</Last_name>
      <Hours>32.5</Hours>
    </Worker>
    <Worker>
      <Ssn>453453453</Ssn>
      <First_name>Joyce</First_name>
      <Hours>20.0</Hours>
    </Worker>
  </Project>
  <Project>
    <Name>ProductY</Name>
    <Number>2</Number>
    <Location>Sugarland</Location>
    <Dept_no>5</Dept_no>
    <Worker>
      <Ssn>123456789</Ssn>
      <Hours>7.5</Hours>
    </Worker>
    <Worker>
      <Ssn>453453453</Ssn>
      <Hours>20.0</Hours>
    </Worker>
    <Worker>
      <Ssn>333445555</Ssn>
      <Hours>10.0</Hours>
    </Worker>
  </Project>
...
</Projects>
```
XML is called a tree or an hierarchical model (nodes represent complex elements, leaf nodes represent simple elements)

Three main types of XML documents:

- Data-centric (predefined schema, DB used to exchange data)
- Document-centric (large amounts of text, news articles, books)
- Hybrid (partially structured, some non-structured, textual)

Schema-less XML document do not follow predefined schema

Predefined XML schema is known as DTD structured data

**XML Document, DTD and XML Schema**

Well-formed: if it follows few conditions:

- Start with XML declaration (version)
- Follow the specific guidelines of the tree data model (single root, every element must include a matching pair of start and end tags)
- Nested elements specify a well-formed tree structure
- A well-formed XML document is syntactically correct
- Preprocessor that traverse the document and create an internal tree
- Document object model (DOM) is a standard model with DOM API manipulate the tree
- The whole document must be parsed before converting it to DOM internal data structure API
- Another API called SAX (Simple API for XML) processing on the fly through callback whenever a start and an end tag
- Criteria is an XML document to be valid
  - Document must be well-formed
  - Must follow a particular schema
- The element names used in the start and end tag pairs must follow the structure specified in DTD (document type definition) or XML schema file
- Special syntax for DTD
- When specifying elements, special notation is used

**Figure 13.4a** An XML DTD file called *Projects*.

```
(a) <!DOCTYPE Projects [  
  <!ELEMENT Projects (Project+)>  
  <!ELEMENT Project (Name, Number, Location, Dept_no?, Workers)>  
    <!ATTLIST Project  
      Projld ID #REQUIRED>  
  <!ELEMENT Name (#PCDATA)>  
  <!ELEMENT Number (#PCDATA)>  
  <!ELEMENT Location (#PCDATA)>  
  <!ELEMENT Dept_no (#PCDATA)>  
  <!ELEMENT Workers (Worker*)>  
    <!ELEMENT Worker (Ssn, Last_name?, First_name?, Hours)>  
  <!ELEMENT Ssn (#PCDATA)>  
  <!ELEMENT Last_name (#PCDATA)>  
  <!ELEMENT First_name (#PCDATA)>  
  <!ELEMENT Hours (#PCDATA)>  
  ]>
```
Figure 13.4b  An XML DTD file called Company.

(b)  <!DOCTYPE Company [  
<!ELEMENT Company( (Employee|Department|Project)*)>  
<!ELEMENT Department (DName, Location+)>  
   <!ATTLIST Department  
   DeptId ID #REQUIRED>  

<!ELEMENT Employee (EName, Job, Salary)>  
   <!ATTLIST Project  
   EmpId ID #REQUIRED  
   DeptId IDREF #REQUIRED>  
<!ELEMENT Project (PName, Location)  
   <!ATTLIST Project  
   ProjId ID #REQUIRED  
   Workers IDREFS #IMPLIED>  
<!ELEMENT DName (#PCDATA)>  
<!ELEMENT EName (#PCDATA)>  
<!ELEMENT PName (#PCDATA)>  
<!ELEMENT Job (#PCDATA)>  
<!ELEMENT Location (#PCDATA)>  
<!ELEMENT Salary (#PCDATA)>  
] >
#PCDATA  parsed character data (string)

<ELEMENT  * can be repeated 0 or more
      + can be repeated one or more
      ? zero or once repeated
      Nothing (once)

IDREF similar to foreign key

Parenthesis can be nested

**XML Schema**

- Is a standard language for specifying the structure of XML documents
- It uses the same syntax rules as XML documents (XML instance and XML)
- Contains tag names, tree structure, constraints, and other descriptors, but no data
- The schema is based on tree data model and also borrows some DB concepts
- It has the following:
Figure 13.5  An XML schema file called company.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">Company Schema (Element Approach) - Prepared by Babak Hojabri</xsd:documentation>
  </xsd:annotation>
  <xsd:element name="company">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="department" type="Department" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="employee" type="Employee" minOccurs="0" maxOccurs="unbounded">
          <xsd:unique name="dependentNameUnique">
            <xsd:selector xpath="employeeDependent"/>
            <xsd:field xpath="dependentName"/>
          </xsd:unique>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="departmentNameUnique">
      <xsd:selector xpath="department"/>
      <xsd:field xpath="departmentName"/>
    </xsd:unique>
    <xsd:unique name="projectNameUnique">
      <xsd:selector xpath="project"/>
      <xsd:field xpath="projectName"/>
    </xsd:unique>
    <xsd:key name="projectNumberKey">
      <xsd:selector xpath="project"/>
      <xsd:field xpath="projectNumber"/>
    </xsd:key>
    <xsd:key name="departmentNumberKey">
      <xsd:selector xpath="department"/>
      <xsd:field xpath="departmentNumber"/>
    </xsd:key>
    <xsd:key name="employeeSSNKey">
      <xsd:selector xpath="employee"/>
      <xsd:field xpath="employeeSSN"/>
    </xsd:key>
    <xsd:keyref name="departmentManagerSSNKeyRef" refer="employeeSSNKey">
      <xsd:selector xpath="department"/>
      <xsd:field xpath="departmentManagerSSN"/>
    </xsd:keyref>
  </xsd:element>
</xsd:schema>
```
<xsd:complexType name="Department">
    <xsd:sequence>
        <xsd:element name="departmentName" type="xsd:string"/>
        <xsd:element name="departmentNumber" type="xsd:string"/>
        <xsd:element name="departmentManagerSSN" type="xsd:string"/>
        <xsd:element name="departmentManagerStartDate" type="xsd:date"/>
        <xsd:element name="departmentLocation" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Employee">
    <xsd:sequence>
        <xsd:element name="employeeName" type="Name"/>
        <xsd:element name="employeeSSN" type="xsd:string"/>
        <xsd:element name="employeeSex" type="xsd:string"/>
        <xsd:element name="employeeSalary" type="xsd:unsignedInt"/>
        <xsd:element name="employeeBirthDate" type="xsd:date"/>
        <xsd:element name="employeeDepartmentNumber" type="xsd:string"/>
        <xsd:element name="employee SupervisorSSN" type="xsd:string"/>
        <xsd:element name="employeeAddress" type="Address"/>
        <xsd:element name="employeeWorksOn" type="WorksOn" minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="employeeDependent" type="Dependent" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Project">
    <xsd:sequence>
        <xsd:element name="projectName" type="xsd:string"/>
        <xsd:element name="projectNumber" type="xsd:string"/>
        <xsd:element name="projectLocation" type="xsd:string"/>
        <xsd:element name="projectDepartmentNumber" type="xsd:string"/>
        <xsd:element name="projectWorker" type="Worker" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Dependent">
    <xsd:sequence>
        <xsd:element name="dependentName" type="xsd:string"/>
        <xsd:element name="dependentSex" type="xsd:string"/>
        <xsd:element name="dependentBirthDate" type="xsd:date"/>
        <xsd:element name="dependentRelationship" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Address">
    <xsd:sequence>
        <xsd:element name="number" type="xsd:string"/>
        <xsd:element name="street" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
        <xsd:element name="state" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Name">
    <xsd:sequence>
        <xsd:element name="firstName" type="xsd:string"/>
        <xsd:element name="middleName" type="xsd:string"/>
        <xsd:element name="lastName" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Worker">
    <xsd:sequence>
        <xsd:element name="SSN" type="xsd:string"/>
        <xsd:element name="hours" type="xsd:float"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="WorksOn">
    <xsd:sequence>
        <xsd:element name="projectNumber" type="xsd:string"/>
        <xsd:element name="hours" type="xsd:float"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:schema>
- Schema descriptions and XML name spaces
- Annotations, documentation and language used
- Elements and types
- First level elements in the COMPANY database
- Specifying element type and minimum and max occurrences
- Specifying keys
- Specifying the structure of complex elements via complex types
- Composite (compound) attributes

**Storing and Extracting XML documents from Databases**

Several approaches

(1) Using a file system or a DBMS to store documents as text (as a text field within a DBMS record)

(2) Using a DBMS to store the document contents as data elements
    (one can design a relational database to store the leaf level data elements within the XML document; need mapping algorithms to design a database schema that is compatible with XML DTD or schema; also recreate XML document from stored data)

**Designing a specialized system for storing native XML data**

A new DBMS based on hierarchical (tree) model could be designed to native XML DBMS

- Require specialized indexing and querying technique
- Include data compression techniques to reduce data
- Tamino, eXcelon
- Oracle offers native storage option
Creating or publishing Customized XML documents from preexisting RDBMS

- Middleware to handle conversion
- This approach gives a good conceptual understanding of the differences between XML tree data model and the ER model

**XML Languages**

XML Query Languages

- XPath
- XQuery

XPath provides language constructs for applying path expressions to identify certain nodes or element attributes within an XML document.

XQuery uses XPath expressions and has additional constructs.
Figure 13.6  Some examples of XPath expressions on XML documents that follow the XML schema file company in Figure 13.5.

1. /company

2. /company/department

3. //employee [employeeSalary gt 70000]/employeeName

4. /company/employee [employeeSalary gt 70000]/employeeName

5. /company/project/projectWorker [hours ge 20.0]

Two main separators are used:

- Single slash (tag must appear as a direct child of the previous tag (parent)
- Double slash (tag can appear as a descendant of the previous tag at any level)

**SKIP section 13.5**
Extracting XML Documents from RDBMS

Issues in converting data from a database system to XML documents (most commonly used Relational model)

Figure 13.8  An ER schema diagram for a simplified UNIVERSITY database.
Suppose that an application needs to extract XML documents for student, course, and grade information. This information is contained in STUDENT, SECTION and COURSE as shown in Fig. 13.9.

Three possible hierarchies can be created from Fig. 13.9.

(1) Course as the root
(2) Student as the root
(3) Section as the root
Figure 13.10  Hierarchical (tree) view with COURSE as the root.
Figure 13.12  Hierarchical (tree) view with STUDENT as the root.
Figure 13.14  Course_name Hierarchical (tree) view with SECTION as the root.
Figure 13.11  XML schema document with course as the root.

```xml
<xsd:element name="root">
  <xsd:sequence>
    <xsd:element name="course" minOccurs="0" maxOccurs="unbounded">
      <xsd:sequence>
        <xsd:element name="cname" type="xsd:string" />
        <xsd:element name="cnumber" type="xsd:unsignedInt" />
        <xsd:element name="section" minOccurs="0" maxOccurs="unbounded">
          <xsd:sequence>
            <xsd:element name="secnumber" type="xsd:unsignedInt" />
            <xsd:element name="year" type="xsd:string" />
            <xsd:element name="quarter" type="xsd:string" />
            <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
              <xsd:sequence>
                <xsd:element name="ssn" type="xsd:string" />
                <xsd:element name="sname" type="xsd:string" />
                <xsd:element name="class" type="xsd:string" />
                <xsd:element name="grade" type="xsd:string" />
              </xsd:sequence>
            </xsd:element>
          </xsd:sequence>
        </xsd:element>
      </xsd:sequence>
    </xsd:element>
  </xsd:sequence>
</xsd:element>
```

22
<xsd:element name="root">
  <xsd:sequence>
    <xsd:element name="student" minOccurs="0" maxOccurs="unbounded">
      <xsd:sequence>
        <xsd:element name="ssn" type="xsd:string" />
        <xsd:element name="sname" type="xsd:string" />
        <xsd:element name="class" type="xsd:string" />
        <xsd:element name="section" minOccurs="0" maxOccurs="unbounded">
          <xsd:sequence>
            <xsd:element name="secnumber" type="xsd:unsignedInt" />
            <xsd:element name="year" type="xsd:string" />
            <xsd:element name="quarter" type="xsd:string" />
            <xsd:element name="cnumber" type="xsd:unsignedInt" />
            <xsd:element name="cname" type="xsd:string" />
            <xsd:element name="grade" type="xsd:string" />
          </xsd:sequence>
        </xsd:element>
      </xsd:element>
    </xsd:element>
  </xsd:element>
</xsd:element>
Each course entity has the set of its sections as subelements and each section has its students as subelements. If a student has taken multiple sections, that student’s information will appear multiple times once for each section in the document. The grade attribute is moved to STUDENT. Fig. 13.11 show a possible XML schema.

SKIP 13.7