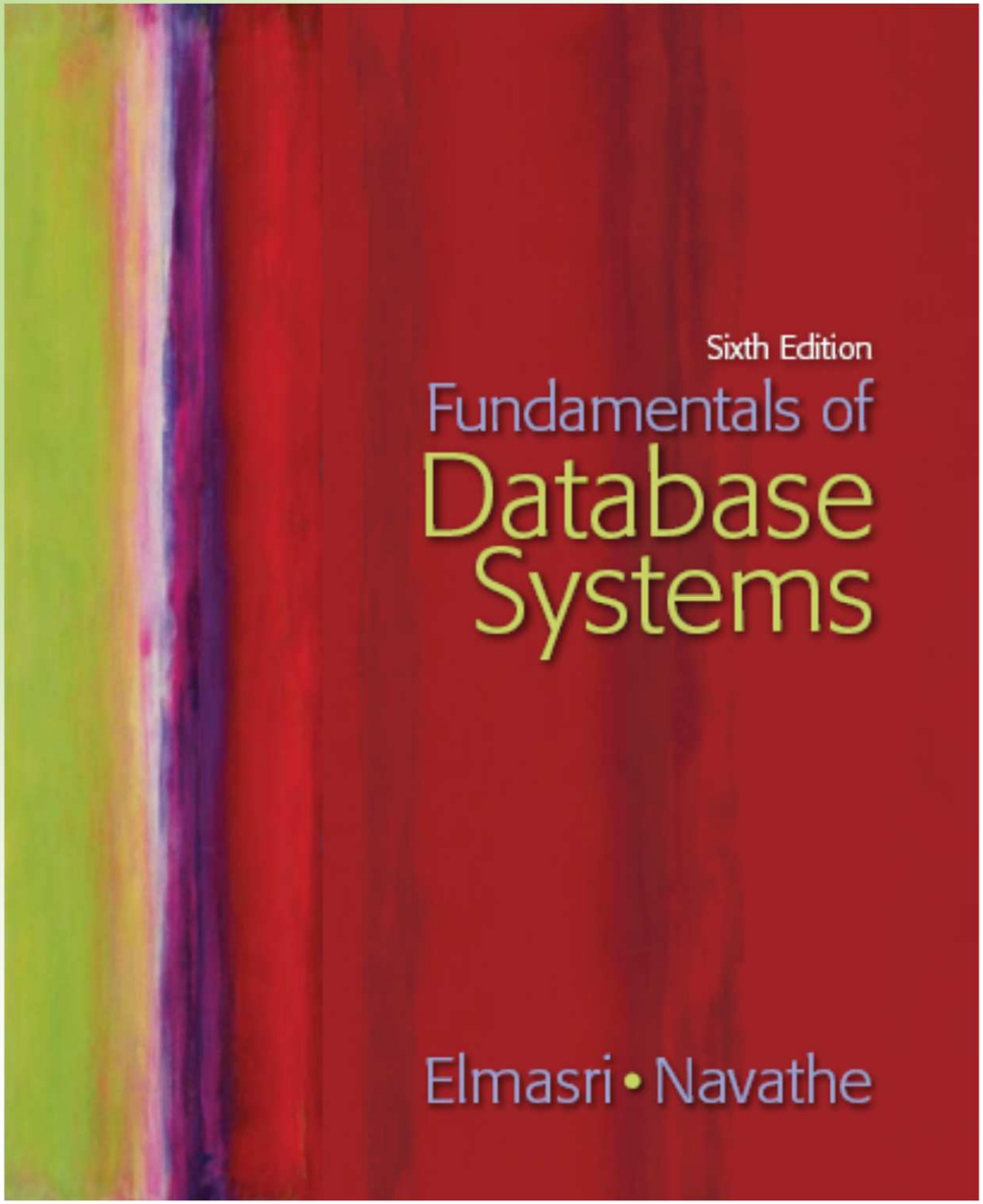
Chapter 12

XML: Extensible Markup Language



Addison-Wesley
is an imprint of
PEARSON

XML: Extensible Markup Language

- **Data sources**
  - Database storing data for Internet applications
- **Hypertext documents (HTML)**
  - Common method of specifying contents and formatting of Web pages
- **XML data model**
  - To structure and exchange data on Web
  - Standard since may-2001, by W3C

Structured and Semistructured Data

- **Structured data**
  - Represented in a strict format
  - Example: information stored in databases
- **Semistructured data**
  - Has a certain structure
  - Not all information collected will have identical structure
  - Additional attributes can be introduced in some of the newer data items at any time

Semistructured Data

- **Self-describing data:** schema information mixed in with data values
- Data may be displayed as a directed graph
  - **Labels** or **tags** on directed edges represent schema names:
    - Names of attributes
    - Object types (or entity types or classes)
    - Relationships
  - The **internal nodes** represent individual objects or composite attributes
  - The **leaf nodes** represent data values

Semistructured Data (cont'd.)

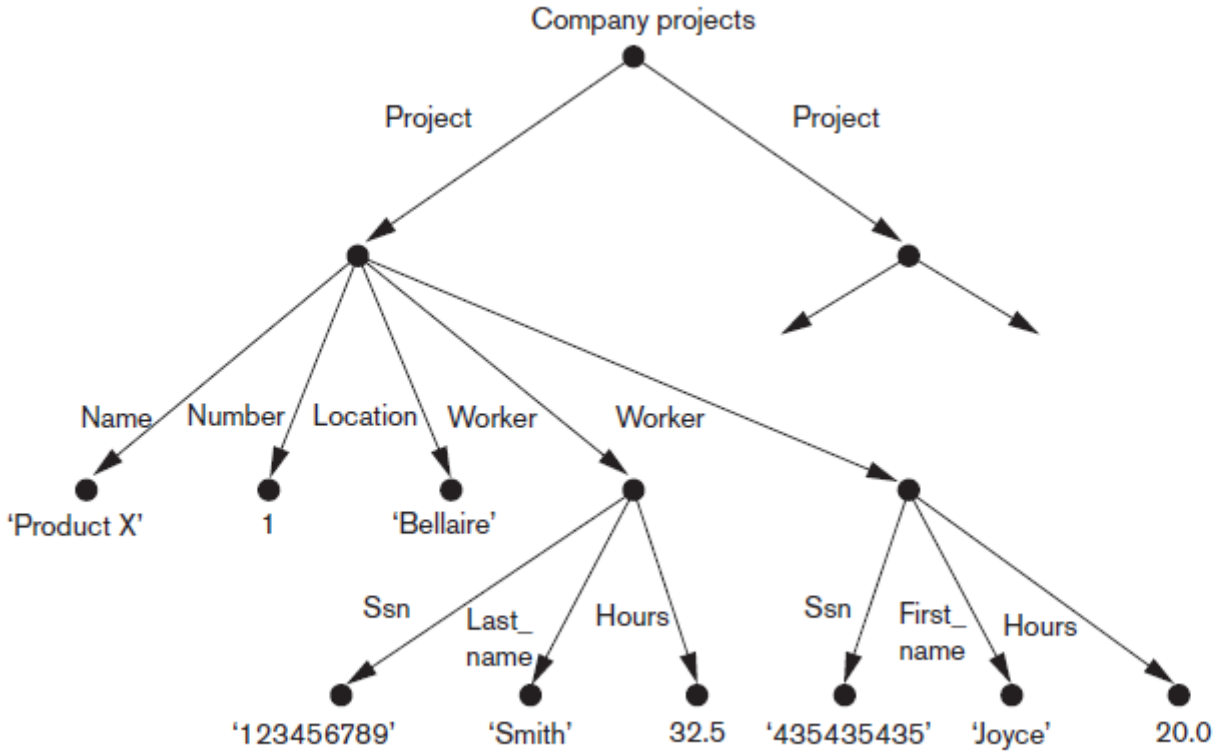


Figure 12.1
Representing semistructured data as a graph.

Unstructured Data

- **Unstructured data**
  - Limited indication of the type of data. An example is a text document that contains information embedded within it
- **HTML tag**
  - Text that appears between angled brackets: <. . .>
- **End tag**
  - Tag with a slash: </ . . .>

Unstructured Data (cont'd.)

- HTML uses a large number of predefined tags
- HTML documents
  - Do not include schema information about type of data
- **Static** HTML page
  - All information to be displayed explicitly spelled out as fixed text in HTML file

Figure 12.2
Part of an HTML document representing unstructured data.

```
<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>
```

XML Hierarchical (Tree) Data Model

- **Elements** and **attributes**
  - Main structuring concepts used to construct an XML document
- **Simple elements**
  - Contains data values
- **Complex elements**
  - Constructed from other elements hierarchically
- **XML tag names**
  - Describe the meaning of the data elements in the document, not how the text is to be displayed.

```
<?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>
```

Figure 12.3
A complex XML element called <Projects>.

XML Hierarchical (Tree) Data Model (cont'd.)

- Main types of XML documents
  - **Data-centric XML documents**
  - **Document-centric XML documents**
  - **Hybrid XML documents**
- **Schemaless XML documents**
  - Do not follow a predefined schema of element names and corresponding tree structure (standalone="yes")

XML Hierarchical (Tree) Data Model (cont'd.)

- XML attributes
  - Describe properties and characteristics of the elements (tags) within which they appear
- May **reference** another element in another part of the XML document
  - It is common to use attribute values in one element as the references (resembles the concept of foreign keys in relational databases)

XML Documents and XML Schema

- An XML document is **well formed** if:
  - It has a **XML declaration**
    - Indicates version of XML being used as well as any other relevant attributes
  - Every element includes a matching pair of start and end tags
    - Within start and end tags of parent element
- **DOM** (Document Object Model)
  - APIs to allow programs to manipulate resulting tree representation corresponding to a well-formed XML document

XML Documents and XML Schema (cont'd.)

- **Valid XML document**
  - Document must be well formed
  - Document must follow a particular schema
  - Start and end tag pairs must follow structure specified in a separate XML schema file

XML Schema

- **XML schema language**
  - Standard for specifying the structure of XML documents
  - Uses same syntax rules as regular XML documents
    - Same processors can be used on both

Figure 12.5
An XML schema file called *company*.

```
<?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:element name="project" type="Project" minOccurs="0" maxOccurs="unbounded" />
        </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>
```

XML Schema (cont'd.)

- This schema identify specific set of XML schema language elements (tags) being used
- XML namespace
  - Defines the set of commands (names) that can be used

XML Schema (cont'd.)

- XML schema concepts:
  - Description and XML namespace
  - Annotations, documentation, language
  - Elements and types
  - First level element
  - Element types, minOccurs, and maxOccurs
  - Keys
  - Structures of complex elements
  - Composite attributes

XML Languages

- Two query language standards
  - **XPath**
    - Specify path expressions to identify certain nodes (elements) or attributes within an XML document that match specific patterns
  - **XQuery**
    - Uses XPath expressions but has additional constructs

XPath: Specifying Path Expressions in XML

- XPath expression
  - Returns a sequence of items that satisfy a certain pattern as specified by the expression: values (from leaf nodes), elements or attributes
  - **Qualifier conditions**
    - Further restrict nodes that satisfy pattern
- **Separators** used when specifying a path:
  - Single slash (/) and double slash (//)

XPath: Specifying Path Expressions in XML (cont'd.)

Figure 12.6
Some examples of XPath expressions on XML documents that follow the XML schema file *company* in Figure 12.5.

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]

XPath: Specifying Path Expressions in XML (cont'd.)

- **Axes**
  - Move in multiple directions from current node in path expression
  - Include self, child, descendent, attribute, parent, ancestor, previous sibling, and next sibling

XPath: Specifying Path Expressions in XML (cont'd.)

- Main restriction of XPath path expressions
  - Path that specifies the pattern also specifies the items to be retrieved
  - Difficult to specify certain conditions on the pattern while separately specifying which result items should be retrieved

XQuery: Specifying Queries in XML

- XQuery FLWR expression
  - Four main clauses of XQuery
  - Form:

```
FOR <variable bindings to individual
  nodes (elements)>
LET <variable bindings to collections
  of nodes (elements)>
WHERE <qualifier conditions>
RETURN <query result specification>
```

  - Zero or more instances of `FOR` and `LET` clauses

```
LET $d := doc(www.company.com/info.xml)
FOR $x IN $d/company/project[projectNumber = 5]/projectWorker,
       $y IN $d/company/employee
WHERE $x/hours gt 20.0 AND $y.ssn = $x.ssn
RETURN <res> $y/employeeName/firstName, $y/employeeName/lastName,
                  $x/hours </res>
```

1. FOR $x IN
   doc(www.company.com/info.xml)
   //employee [employeeSalary gt 70000]/employeeName
   RETURN <res> $x/firstName, $x/lastName </res>
2. FOR $x IN
   doc(www.company.com/info.xml)/company/employee
   WHERE $x/employeeSalary gt 70000
   RETURN <res> $x/employeeName/firstName, $x/employeeName/lastName </res>
3. FOR $x IN
   doc(www.company.com/info.xml)/company/project[projectNumber = 5]/projectWorker,
   $y IN doc(www.company.com/info.xml)/company/employee
   WHERE $x/hours gt 20.0 AND $y.ssn = $x.ssn
   RETURN <res> $y/employeeName/firstName, $y/employeeName/lastName, $x/hours </res>

Figure 12.7
Some examples of XQuery queries on XML documents that follow the XML schema file *company* in Figure 12.5.

XQuery: Specifying Queries in XML (cont'd.)

- XQuery contains powerful constructs to specify complex queries
  - Universal and existential quantifiers, aggregate functions, ordering of query results,…
- www.w3c.org
  - World Wide Web Consortium (W3C).
  - Contains documents describing the latest standards related to XML and XQuery

Other Languages and Protocols Related to XML

- Extensible Stylesheet Language (XSL)
  - Define how a document should be rendered for display by a Web browser
- Extensible Stylesheet Language for Transformations (XSLT)
  - Transform one structure into different structure
- Web Services Description Language (WSDL)
  - Description of Web Services in XML

Storing and Extracting XML Documents from Databases

- Most common approaches
  - Using a DBMS to store the documents as text
    - Can be used if DBMS has a special module for document processing
  - Using a DBMS to store document contents as data elements
    - Require mapping algorithms to design a database schema that is compatible with XML document structure

Storing and Extracting XML Documents from Databases (cont'd.)

- Designing a specialized system for storing native XML data
  - Called **Native XML DBMSs**
- Creating or publishing customized XML documents from preexisting relational databases
  - Use a separate middleware software layer to handle conversions

Extracting XML Documents from Relational Databases

- Creating hierarchical XML views over flat or graph-based data
  - Representational issues arise when converting data from a database system into XML documents
- UNIVERSITY database example

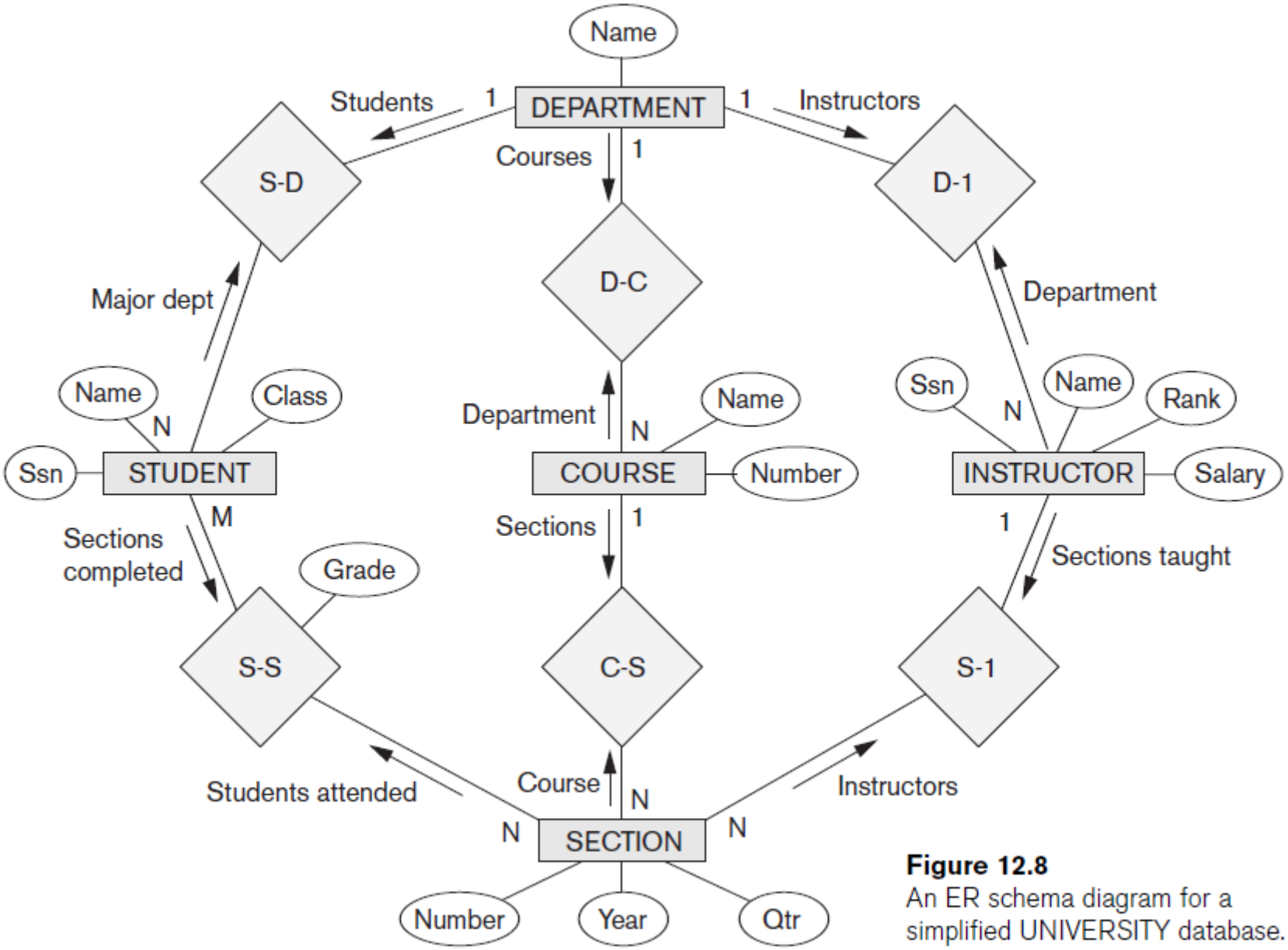


Figure 12.8
An ER schema diagram for a simplified UNIVERSITY database.

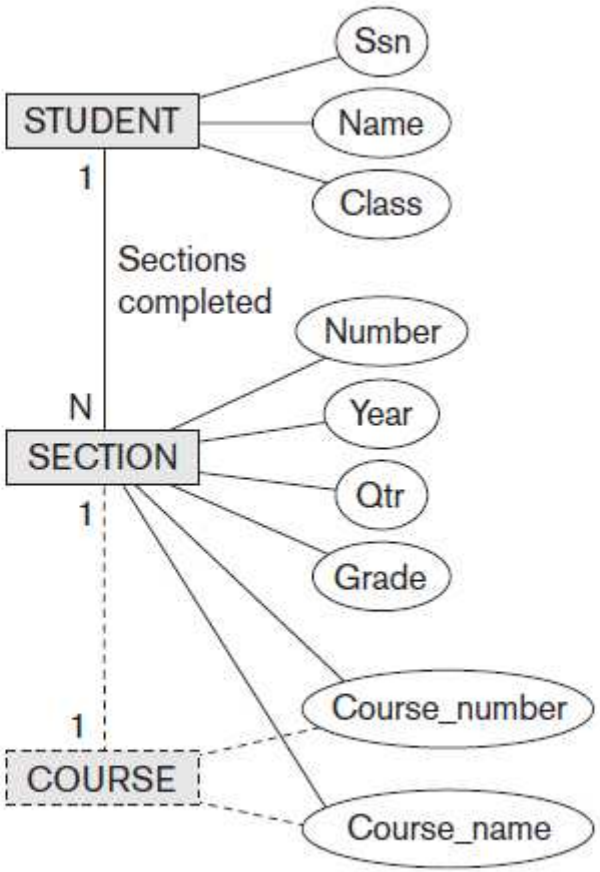


Figure 12.12
Hierarchical (tree) view with STUDENT as the root.

```
<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:sequence>
</xsd:element>
</xsd:sequence>
</xsd:element>
```

Figure 12.13
XML schema document with *student* as the root.

Summary

- Three main types of data: structured, semi-structured, and unstructured
- XML standard
  - Tree-structured (hierarchical) data model
  - XML documents and the languages for specifying the structure of these documents
- XPath and XQuery languages
  - Query XML data