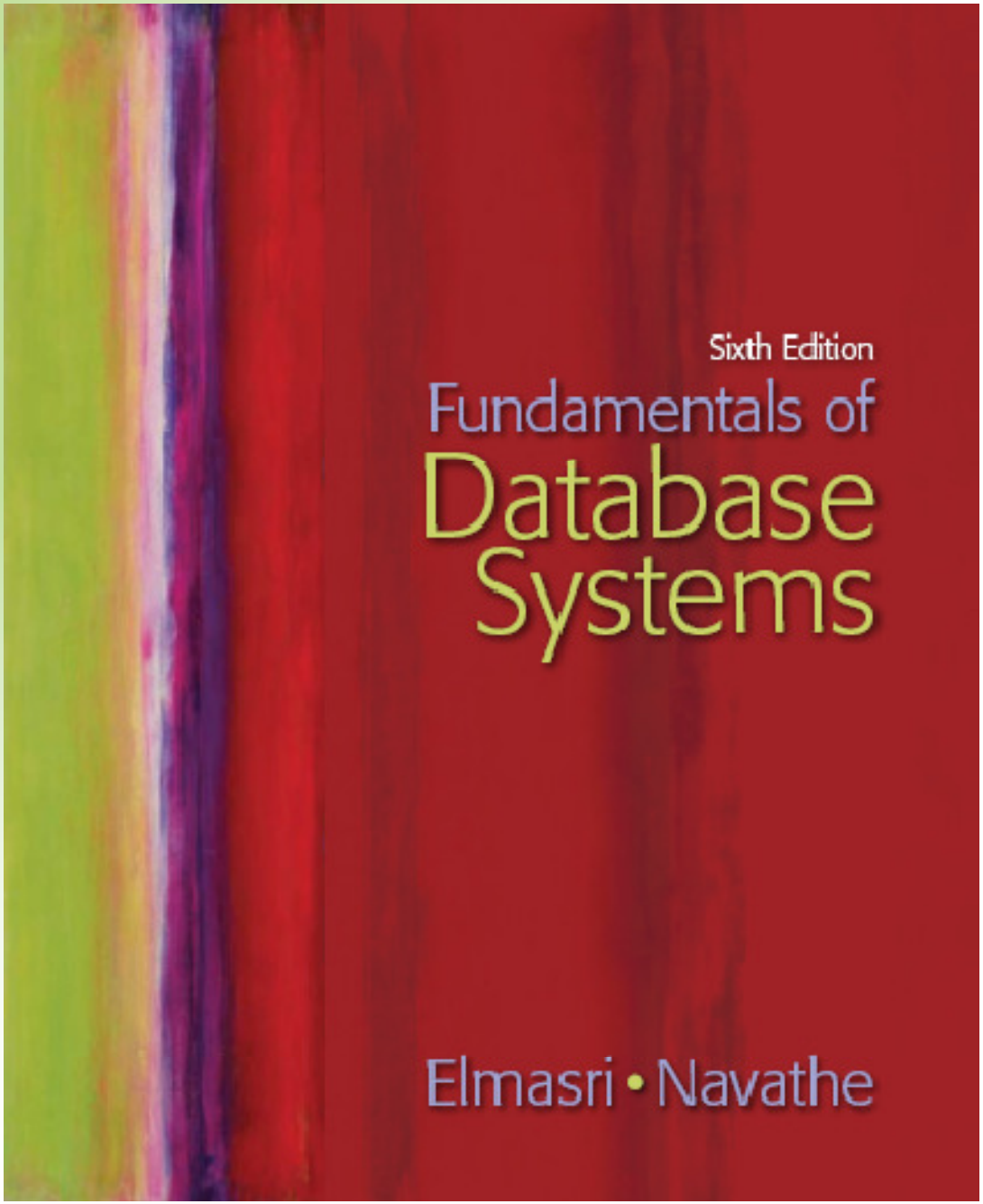
# Chapter 12

# XML: Extensible Markup Language




Addison-Wesley
is an imprint of
PEARSON

# Chapter 12 Outline

- Structured, Semistructured, and Unstructured Data
- XML Hierarchical (Tree) Data Model
- XML Documents, DTD, and XML Schema
- Storing and Extracting XML Documents from Databases
- XML Languages
- Extracting XML Documents from Relational Databases

# XML: Extensible Markup Language

- **Data sources**
  - Database storing data for Internet applications
- **Hypertext documents**
  - Common method of specifying contents and formatting of Web pages
- XML data model

# Structured, Semistructured, and Unstructured 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

# Structured, Semistructured, and Unstructured Data (cont'd.)

- Schema information mixed in with data values
- **Self-describing 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

# Structured, Semistructured, and Unstructured Data (cont'd.)

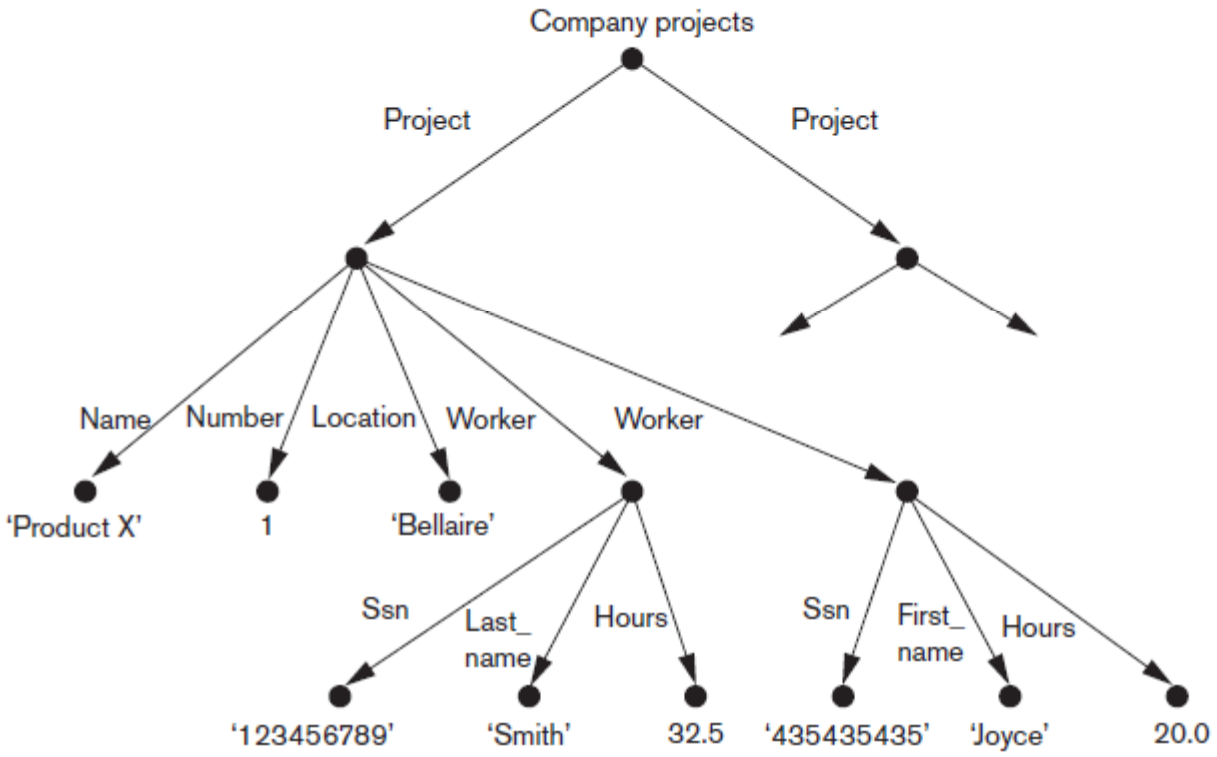



**Figure 12.1**
Representing semistructured data as a graph.

# Structured, Semistructured, and Unstructured Data (cont'd.)

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

# Structured, Semistructured, and 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
- **Complex elements**
  - Constructed from other elements hierarchically
- **Simple elements**
  - Contain data values
- XML tag names
  - Describe the meaning of the data elements in the document

```
<?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.)

- **Tree model** or **hierarchical model**
- 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

# 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
  - Common to use attribute values in one element as the references

# XML Documents, DTD, and XML Schema

- **Well formed**
  - Has **XML declaration**
    - Indicates version of XML being used as well as any other relevant attributes
  - Every element must matching pair of start and end tags
    - Within start and end tags of parent element
- **DOM** (Document Object Model)
  - Manipulate resulting tree representation corresponding to a well-formed XML document

# XML Documents, DTD, and XML Schema (cont'd.)

- **SAX** (Simple API for XML)
  - Processing of XML documents on the fly
    - Notifies processing program through callbacks whenever a start or end tag is encountered
  - Makes it easier to process large documents
  - Allows for **streaming**

# XML Documents, DTD, and XML Schema (cont'd.)

- **Valid**
  - Document must be well formed
  - Document must follow a particular schema
  - Start and end tag pairs must follow structure specified in separate XML **DTD (Document Type Definition)** file or XML schema file

# XML Documents, DTD, and XML Schema (cont'd.)

- Notation for specifying elements
- XML DTD
  - Data types in DTD are not very general
  - Special syntax
    - Requires specialized processors
  - All DTD elements always forced to follow the specified ordering of the document
    - Unordered elements not permitted

# 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.)

- Identify specific set of XML schema language elements (tags) being used
  - Specify a file stored at a Web site location
- **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

# 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

# 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
  - Either values (from leaf nodes) or 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.)

- Attribute name prefixed by the `@` symbol
- **Wildcard** symbol *
  - Stands for any element
  - Example: `/company/*`

# 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
- www.w3.org
  - 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

# Other Languages and Protocols Related to XML (cont'd.)

- Simple Object Access Protocol (SOAP)
  - Platform-independent and programming language-independent protocol for messaging and remote procedure calls
- Resource Description Framework (RDF)
  - Languages and tools for exchanging and processing of meta-data (schema) descriptions and specifications over the Web

# 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

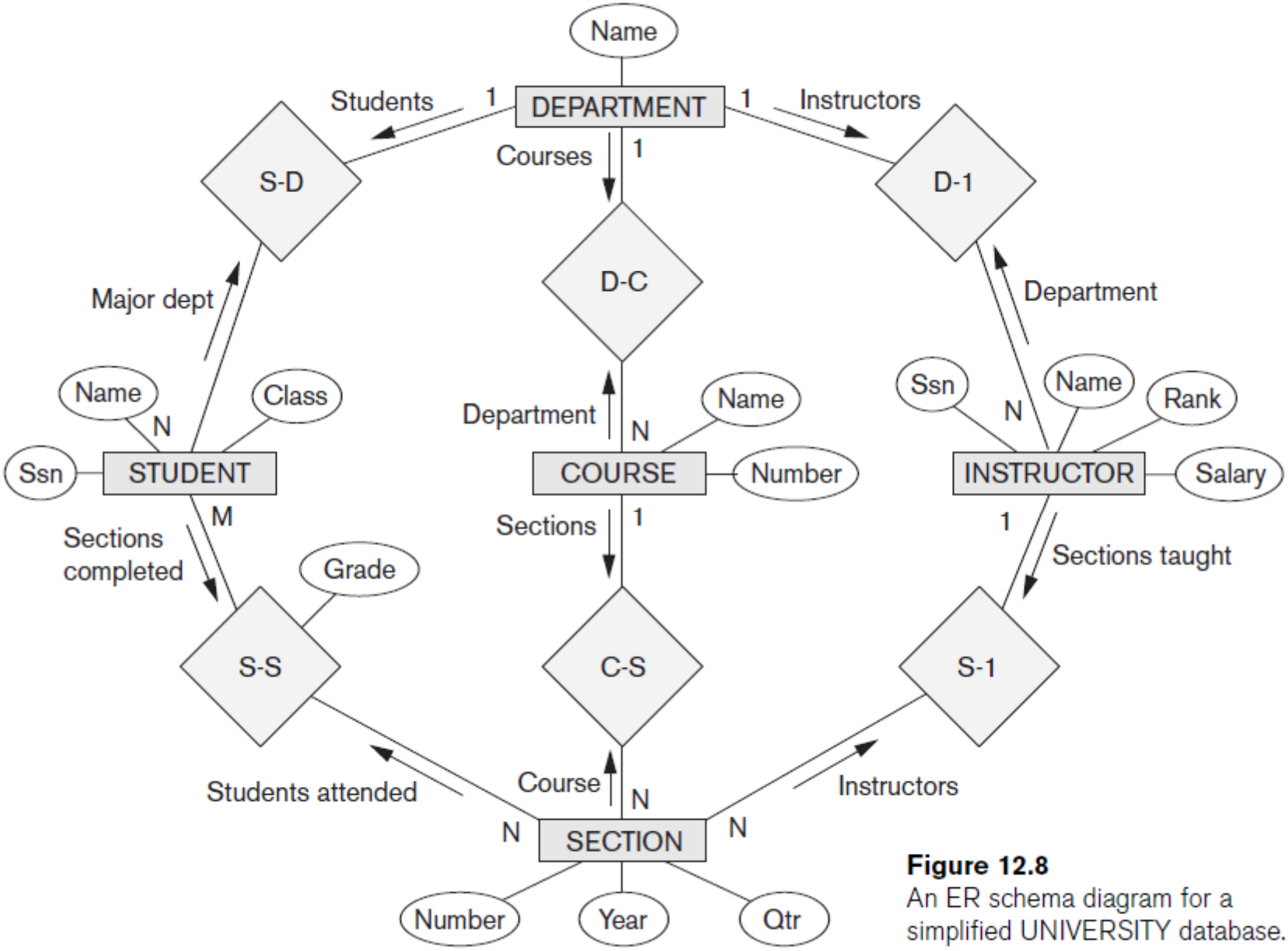



**Figure 12.8**
An ER schema diagram for a simplified UNIVERSITY database.

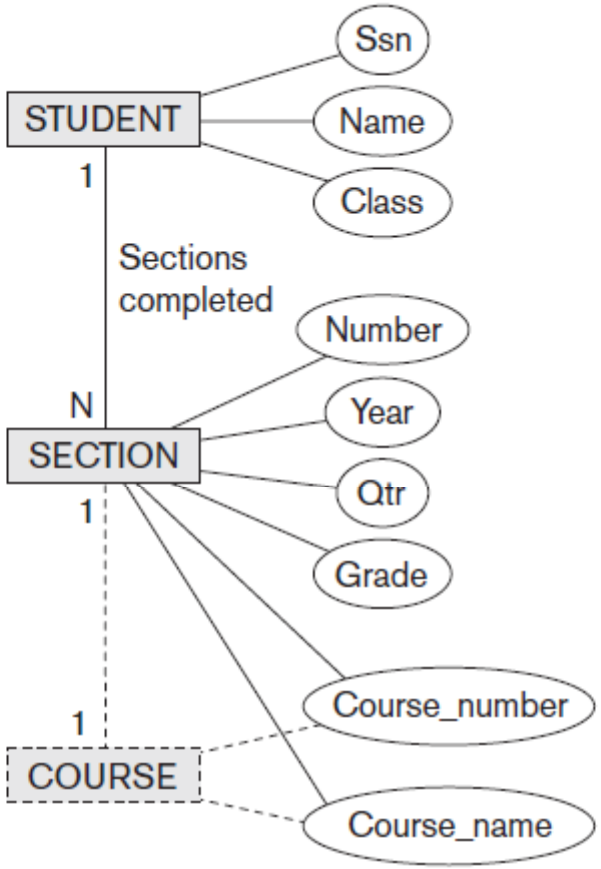



**Figure 12.12**
Hierarchical (tree) view with STUDENT as the root.

# Breaking Cycles to Convert Graphs into Trees

- Complex subset with one or more cycles
  - Indicate multiple relationships among the entities
  - Difficult to decide how to create the document hierarchies
- Can replicate the entity types involved to break the cycles

# Other Steps for Extracting XML Documents from Databases

- Create correct query in SQL to extract desired information for XML document
- Restructure query result from flat relational form to XML tree structure
- Customize query to select either a single object or multiple objects into document

# 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