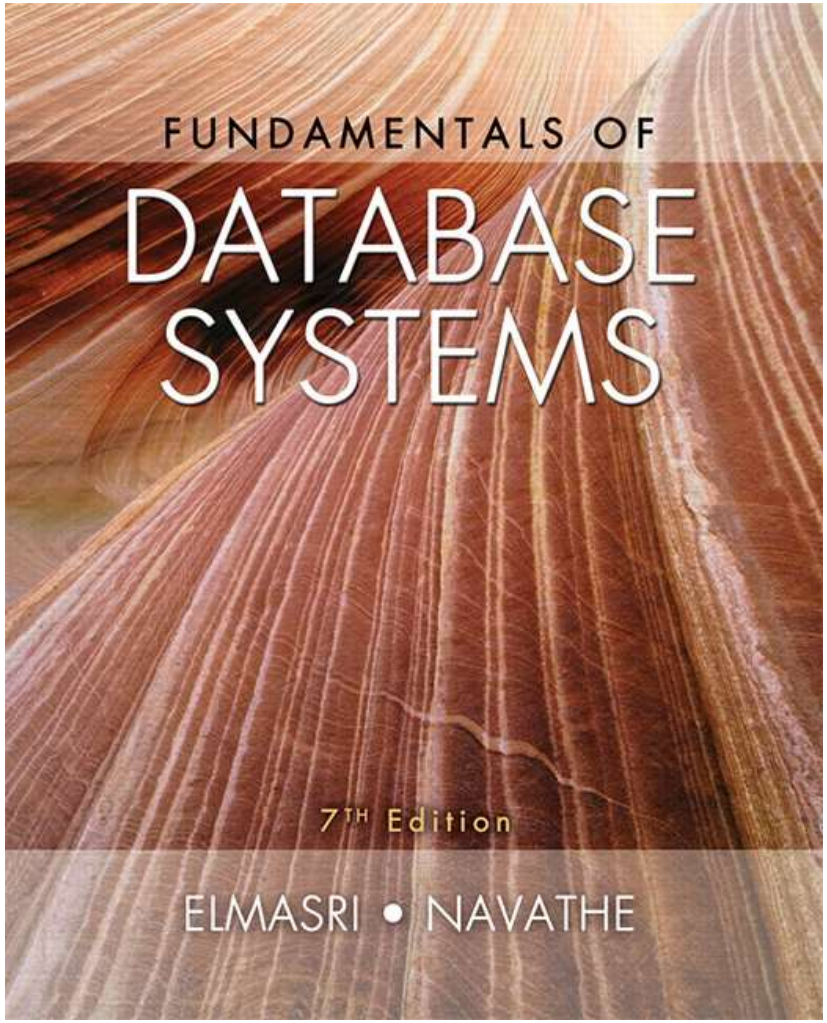


CHAPTER 13

XML: Extensible Markup Language

Introduction

- Databases function as data sources for Web applications
  - HTML
    - Used in static Web pages
  - XML, JSON
    - Self-describing documents
    - Dynamic Web pages
- Chapter focus: XML data model and languages

13.1 Structured, Semistructured, and Unstructured Data

- Structured data
  - Stored in relational database
- Semistructured data
  - Not all data has identical structure
  - Schema information mixed in
    - Self-describing
  - Directed graph model
- Unstructured data
  - Limited data type indication
  - Example: Web pages in HTML

Directed Graph Model for Semistructured Data

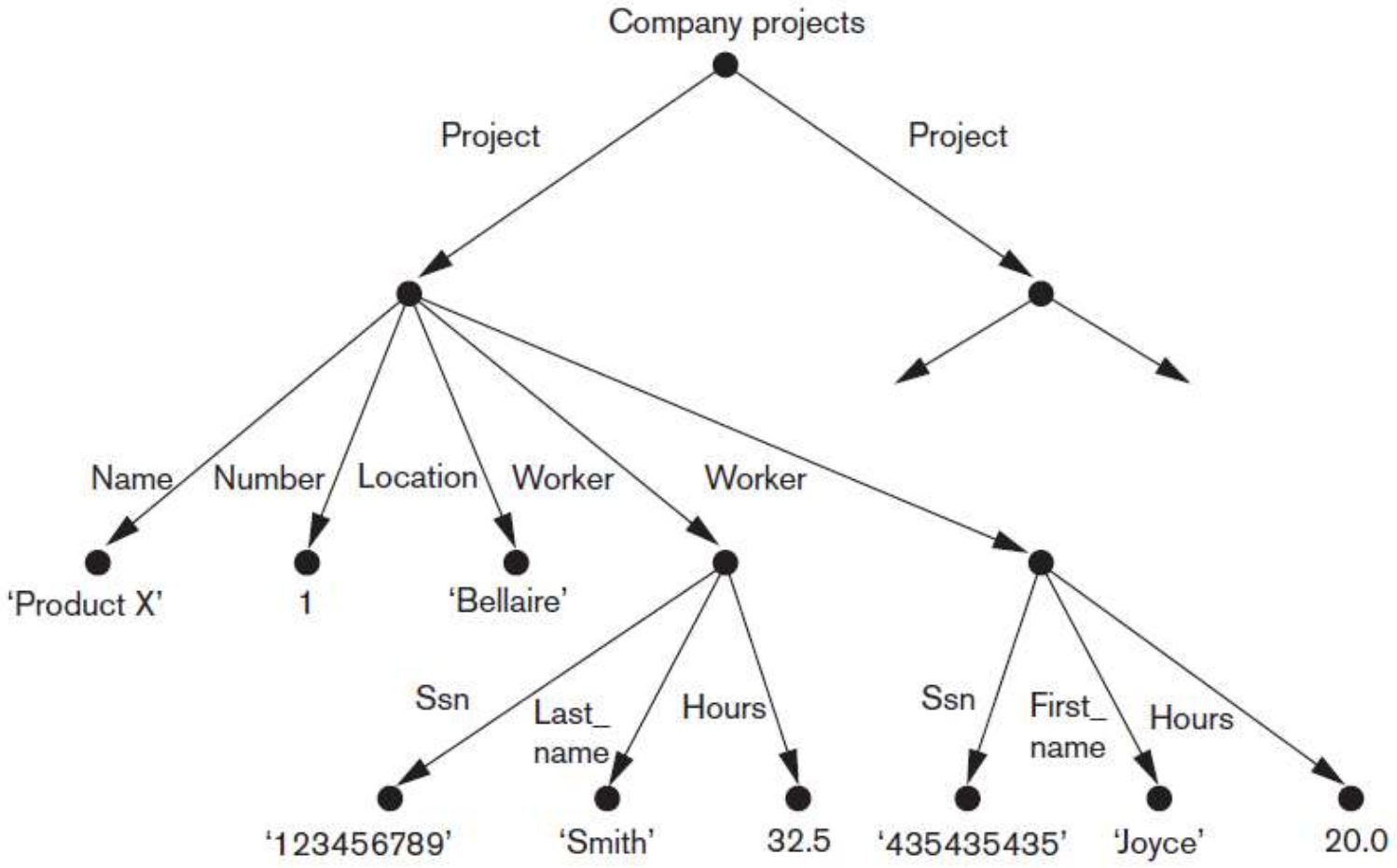


Figure 13.1 Representing semistructured data as a graph

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

- HTML uses predefined tags
  - Document header
    - Script functions, formatting styles
  - Body
  - Table tags
  - Attributes
  - Large number of predefined tags
- XHTML
  - Extends tags for different applications

13.2 XML Hierarchical (Tree) Data Model

- Basic object: XML document
  - Element
    - Simple
    - Complex
    - Schema document defines element names
  - Attribute
- Document types
  - Data-centric
  - Document-centric
  - Hybrid

13.3 XML Documents, DTD, and XML Schema

- Conditions for well-formed XML documents
  - Begins with XML declaration
  - Syntactically correct
  - Valid
    - Must follow a particular schema

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

- Notation for specifying elements
  - elementname* indicates optional multivalued (repeating) element
  - elementname+ indicates required multivalued (repeating) element
  - elementname? indicates optional single-valued (non-repeating) element
  - elementname indicates required single-valued (non-repeating) element

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

- Notation for specifying elements (cont'd.)
  - Type specified by parentheses following the element name
  - !ATTLIST used to specify attributes within the element
  - Parentheses can be nested when specifying elements
  - Bar symbol ($e_1|e_2$) indicates either e_1 or e_2 can appear in the document

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

(a)
```
<!DOCTYPE Projects [
    <!ELEMENT Projects (Project+)>
    <!ELEMENT Project (Name, Number, Location, Dept_no?, Workers)>
        <!ATTLIST Project
            ProjId 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 13.4(a) An XML DTD file called 'Projects'

XML Schema

- XML schema language
  - Specifies document structure
  - Same syntax rules as XML documents
  - Elements, attributes, keys, references, and identifiers
  - Example: Figure 13.5 in the text

XML Schema (cont'd.)

- XML namespace
  - Defines set of commands that can be used
- Annotations, documentation, and language used
- Elements and types
  - Root element
  - First-level elements
  - Specifying element type and min and max occurrences

XML Schema (cont'd.)

- Keys
  - Constraints that correspond to relational database
  - Primary key
  - Foreign key
- Complex elements
  - xsd:complexType
- Composite (compound) attributes

13.4 Storing and Extracting XML Documents from Databases

- Use file system or DBMS to store documents as text
- Use DBMS to store document contents as data elements
- Design specialized system to store XML data
- Create or publish custom XML documents from preexisting relational databases
  - This approach explored further in 13.6

13.5 XML Languages

- Query language standards
  - XPath
  - XQuery
- Specifying XPath expressions in XML
  - Returns sequence of items satisfying certain pattern
    - Values, elements, or attributes
  - Qualifier conditions
  - Separators
    - Single slash / or double slash //

XPath (cont'd.)

- Example
  - For COMPANY.XML document stored at location www.company.com/info.XML
  - doc(www.company.com/infor.XML)/company returns company root node and all descendant nodes

XPath (cont'd.)

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]

Figure 13.6 Some examples of XPath expressions on XML documents that follow the XML schema file company in Figure13.5.

XQuery: Specifying Queries in XML

- FLWOR expression

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

- Variables preceded with $
- For assigns variable to a range
- Where specifies additional conditions
- Order by specifies order of result elements
- Return specifies elements for retrieval

XQuery: Specifying Queries in XML

- Example

```
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
ORDER BY $x/hours
RETURN <res> $y/employeeName/firstName, $y/employeeName/lastName,
                  $x/hours </res>
```

Other Languages and Protocols Related to XML

- Extensible Stylesheet Language (XSL)
- Extensible Stylesheet Language for Transformations (XSLT)
- Web Services Description Language (WSDL)
- Simple Object Access Protocol (SOAP)
- Resource Description Framework (RDF)

13.6 Extracting XML Documents from Relational Databases

- XML uses hierarchical (tree) model
- Common database model is flat relational database
- Conceptually represent using ER schema
- University example (follows)
  - Choices for root: course, student, section

University Example

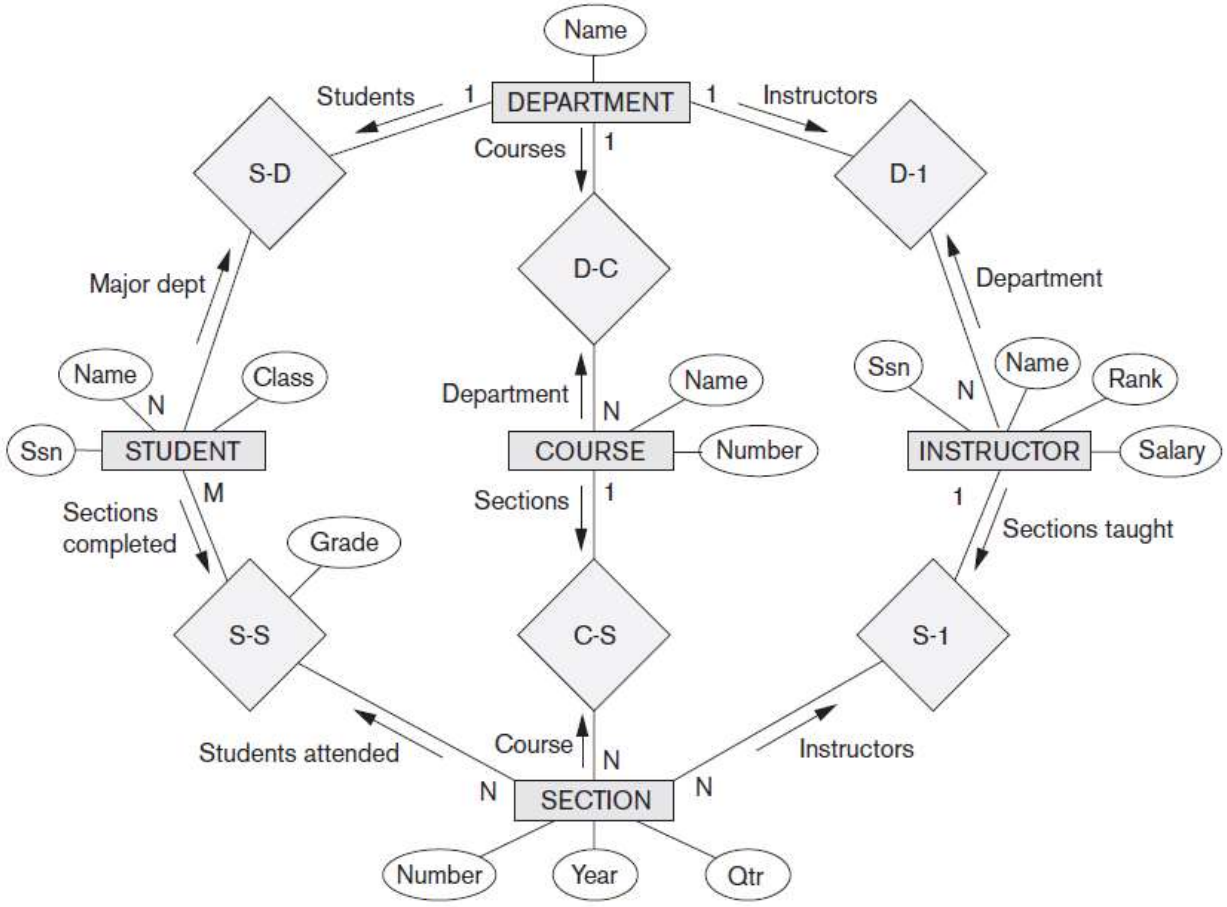


Figure 13.8 An ER schema diagram for a simplified UNIVERSITY database

University Example (cont'd.)

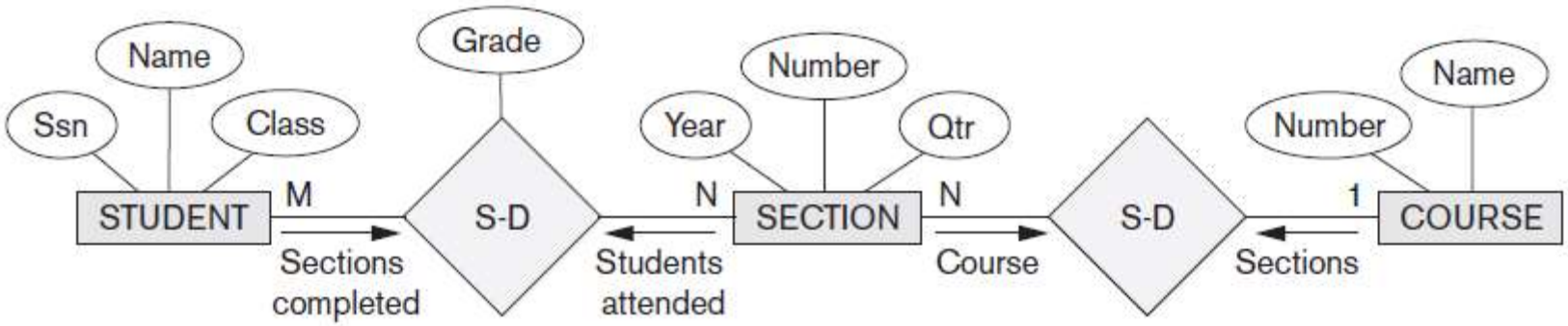


Figure 13.9 Subset of the UNIVERSITY database schema needed for XML document extraction

University Example (cont'd.)

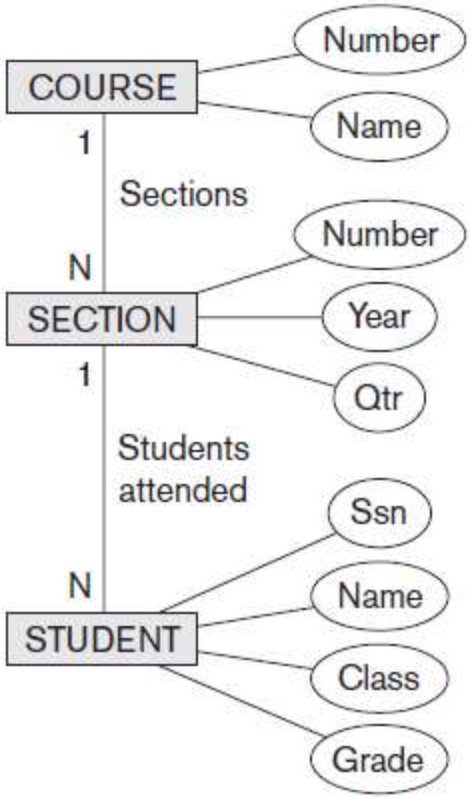


Figure 13.10 Hierarchical (tree) view with 'COURSE' as the root

University Example (cont'd.)

```
<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>
```

Figure 13.11 XML schema with 'COURSE' as the root

University Example (cont'd.)

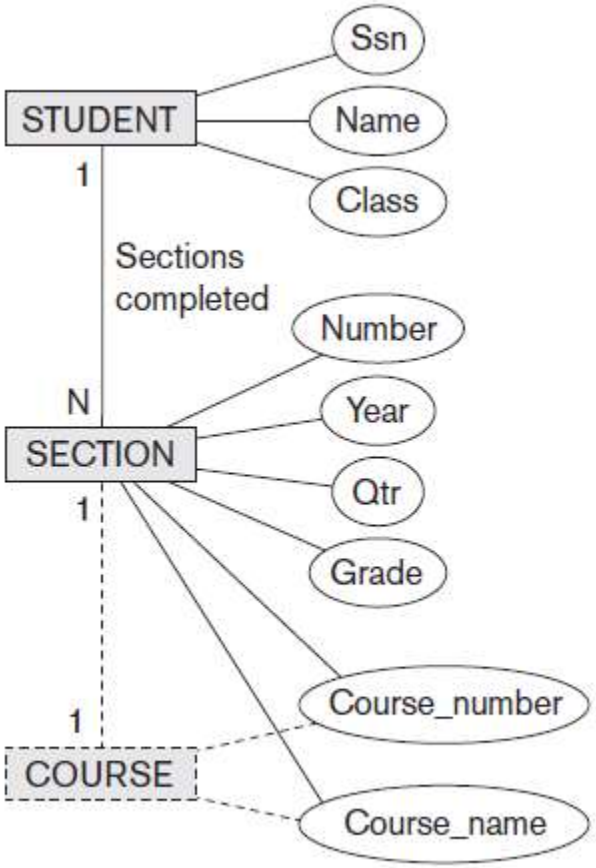


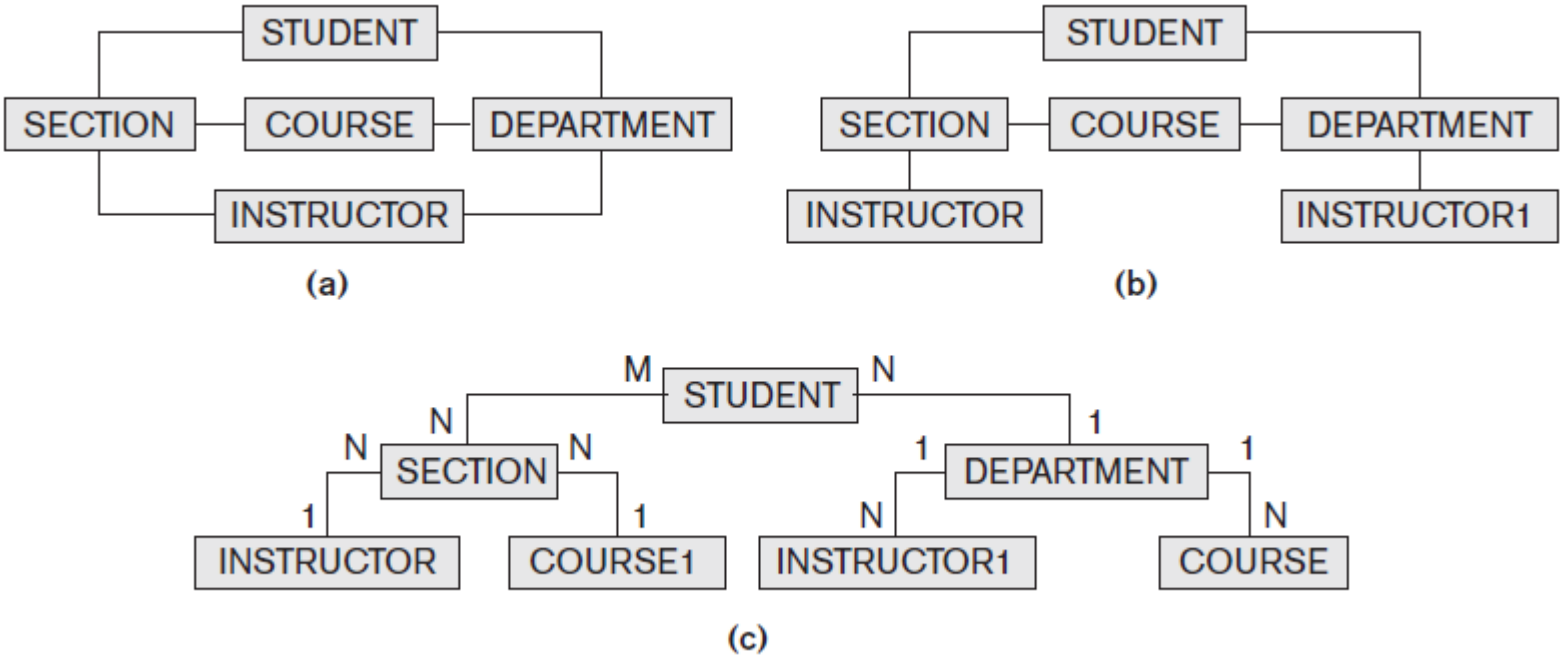
Figure 13.12 Hierarchical (tree) view with 'STUDENT' as the root

University Example (cont'd.)

```
<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 13.13 XML schema document with 'STUDENT' as the root

Breaking Cycles to Convert Graphs into Trees



Other Steps for Extracting XML Documents from Databases

- Create SQL query to extract desired information
- Execute the query
- Restructure from flat to tree structure
- Customize query to select single or multiple objects into the document

13.7 XML/SQL: SQL Functions for Creating XML Data

- XMLELEMENT
  - Specifies tag (element) name that will appear in XML result
- XMLFOREST
  - Specifies multiple element names
- XMLAGG
  - Aggregate several elements
- XMLROOT
  - Selected elements formatted as XML document with single root element

XML/SQL: SQL Functions for Creating XML Data (cont'd.)

- XMLATTRIBUTES
  - Creates attributes for the elements of the XML result
- Example: create XML element containing the EMPLOYEE lastname for the employee with SSN 123456789

```
X1: SELECT    XMLELEMENT (NAME "lastname", E.LName)
    FROM      EMPLOYEE E
    WHERE     E.Ssn = "123456789" ;
```

  - Result: <lastname>Smith</lastname>

Employee Example (Query 2)

- To retrieve multiple columns for a single row:

```
X2:  SELECT    XMLELEMENT (NAME "employee",
                    XMLFOREST (
                        E.Lname AS "ln",
                        E.Fname AS "fn",
                        E.Salary AS "sal" ) )
       FROM    EMPLOYEE AS E
       WHERE   E.Ssn = "123456789" ;
```

- Result:

```
<employee><ln>Smith</ln><fn>John</fn><sal>30000</sal></employee>
```

Employee Example (Query 3)

- To create XML document with last name, first name, and salary of employees from Dept. 4:

```
X3: SELECT    XMLROOT (
              XMLELEMENT (NAME "dept4emps",

              XMLAGG (
                   XMLELEMENT (NAME "emp"
                   XMLFOREST (Lname, Fname, Salary)
                   ORDER BY Lname ) ) )

    FROM      EMPLOYEE
    WHERE     Dno = 4 ;
```

Employee Example (Query 3 cont'd.)

- Result:

```
<dept4emps>
<emp><Lname>Jabbar</Lname><Fname>Ahmad</Fname><Salary>25000
  </Salary></emp>
<emp><Lname>Wallace</Lname><Fname>Jennifer
  </Fname><Salary>43000</Salary></emp>
<emp><Lname>Zelaya</Lname><Fname>Alicia</Fname><Salary>25000
  </Salary></emp>
</dept4emps>
```

13.8 Summary

- Structured, semistructured, and unstructured data
- Hierarchical data model of XML standard
- Languages for specifying structure
  - XML DTD and XML schema
- Approaches for storing XML documents
- XPath and XQuery languages
- Mapping issues
- SQL/XML allows formatting query results as XML data