XML Hierarchical (Tree) Data Model

- The XML data model's basic object is the XML document
- The 2 main structural elements used to construct an XML document are **elements** and **attributes**
- Additional concepts: **entities**, **identifiers** and **references**
- **Complex elements** (internal tree nodes) are constructed from other elements hierarchically
- **Simple elements** (leaf tree nodes) contain data values
- There is no limit on the level of **nesting** of elements
- Identify complex and simple elements in <projects>

XML Hierarchical (Tree) Data Model

- A complex XML element called <projects>

```
<?xml version="1.0" standalone="yes"?>
<projects>
    <project>
      <Name>ProductX</Name>
      <Number>1</Number>
      <Location>Bellaire</Location>
      <DeptNo>5</DeptNo>
      <Worker>
        <SSN>123456789</SSN>
        <LastName>Smith</LastName>
        <hours>32.5</hours>
      </Worker>
      <Worker>
        <SSN>453453453</SSN>
        <FirstName>Joyce</FirstName>
        <hours>20.0</hours>
      </Worker>
    </project>
    <project>
      <Name>ProductY</Name>
      <Number>2</Number>
      <Location>Sugarland</Location>
      <DeptNo >5</DeptNo >
      <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 DTD

- An XML DTD file called projects

```
<!DOCTYPE projects [
  <!ELEMENT projects (project+)>
  <!ELEMENT project (Name, Number, Location, DeptNo?, Workers)>
  <!ELEMENT Name (#PCDATA)>
  <!ELEMENT Number (#PCDATA)>
  <!ELEMENT Location (#PCDATA)>
  <!ELEMENT DeptNo (#PCDATA)>
  <!ELEMENT Workers (Worker*)>
  <!ELEMENT Worker (SSN, LastName?, FirstName?, hours)>
  <!ELEMENT SSN (#PCDATA)>
  <!ELEMENT LastName (#PCDATA)>
  <!ELEMENT FirstName (#PCDATA)>
  <!ELEMENT hours (#PCDATA)>
] >
```

Main types of XML documents

I. **Data-centric XML documents:**
have many small data items that follow a specific structure and hence may be extracted from a structured database.

II. **Document-centric XML documents:**
contain large amounts of text (i.e. articles, books) They contain few (or none at all) structured data elements

III. **Hybrid XML documents:**
they have parts that contain structured data and other parts that are mostly textual or unstructured

- If an XML document conforms to a specific Schema/DTD, then it is considered as **structured data**
- XML allows for documents that do not conform to any Schema/DTD.
  These are considered as **semistructured data** or **schemaless XML documents**
- The value of the **standalone** attribute is **yes**

XML DTD and XML Schema

- Limitations of XML DTD
  - First, the data types in DTD are not very general.
  - Second, DTD has its own special syntax and so it requires specialized processors.
    - It would be advantageous to specify XML schema documents using the syntax rules of XML itself so that the same processors for XML documents can process XML schema descriptions.
  - Third, all DTD elements are always forced to follow the specified ordering of the document so unordered elements are not permitted.

This is why XML Schema was developed

XML Schema

- 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>
```

XML schema file called company

```
    <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:keyref name="employeeDepartmentNumberKeyRef"
        refer="departmentNumberKey">
        <xsd:selector xpath="employee" />
        <xsd:field xpath="employeeDepartmentNumber" />
    </xsd:keyref>
    <xsd:keyref name="employeeSupervisorSSNKeyRef" refer="employeeSSNKey">
        <xsd:selector xpath="employee" />
        <xsd:field xpath="employeeSupervisorSSN" />
    </xsd:keyref>
    <xsd:keyref name="projectDepartmentNumberKeyRef"
        refer="departmentNumberKey">
        <xsd:selector xpath="project" />
        <xsd:field xpath="projectDepartmentNumber" />
    </xsd:keyref>
    <xsd:keyref name="projectWorkerSSNKeyRef" refer="employeeSSNKey">
        <xsd:selector xpath="project/projectWorker" />
        <xsd:field xpath="SSN" />
    </xsd:keyref>
    <xsd:keyref name="employeeWorksOnProjectNumberKeyRef"
        refer="projectNumberKey">
        <xsd:selector xpath="employee/employeeWorksOn" />
        <xsd:field xpath="projectNumber" />
    </xsd:keyref>
</xsd:element>
```

XML schema file called company

```
<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="employeeSupervisorSSN" 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>
```

XML schema file called company

```
    </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>
```

XML Schema

- The **XML schema** language is a standard for specifying the structure of an XML document
- It uses the same syntax rules as regular XML
- Storing data in native XML format has been proposed as an alternative to relational databases
- The previous XML schema file company would specify the structure of the COMPANY database, if it were stored in a native XML system
- XML schema is based on the tree data model (elements, attributes) borrowing concepts from database and object models (keys, references, identifiers)

XML Schema

- **1. Schema Descriptions and XML Namespaces**
  - It is necessary to identify the specific set of XML schema language elements (tags) by a file stored at a Web site location.
    - The second line in our example specifies the file used in this example: "http://www.w3.org/2001/XMLSchema".
  - Each such definition is called an **XML namespace**.
  - The file name is assigned to the variable **xsd** (XML schema description) using the attribute **xmlns** (XML namespace), and this variable is used as a prefix to all XML schema commands (tag names).
  - When we write **xsd:element**, we refer to the definition of the element tag in this Web file

XML Schema

- **2. Annotations, documentation, language used:**
  - The **xsd:annotation** and **xsd:documentation** tags are used for providing comments and other descriptions in the XML document.
  - The attribute **xml:lang** of the **xsd:documentation** element specifies the language being used. E.g., “en”

XML Schema

- **3. Elements and types:**
  - Next, we specify the **root** element of our XML schema. In XML schema, the **name** attribute of the **xsd:element** tag specifies the element name, which is called company for the root element in our example.
  - The structure of the company root element is specified by **xsd:complexType**.
  - This is further specified to be a sequence of departments, employees and projects, using the **xsd:sequence** tag

XML Schema

- **4. First-level elements in the company database:**
  - These elements are named *employee*, *department*, and *project*, and each is specified in an **xsd:element** tag.
  - If a tag has only attributes and no further sub-elements or data within it, it can be ended with the back slash symbol (/>) directly
    (instead of a separate matching end tag)
  - These are called **Empty Elements**.
  - Examples: the **xsd:element** elements, department, project

XML Schema

- **5. Specifying element type and minimum and maximum occurrences**:
  - The attributes **type**, **minOccurs** and **maxOccurs** in the **xsd:element** tag are used for specifying the type and lower and upper bounds on the number of occurrences of each element. (ER: min/max, DTD: +,*,?)
  - If we specify a type attribute in an **xsd:element**, the structure of the element will be described separately, typically using the **xsd:complexType** element of XML Schema.
    Example: employee, department, projects elements
  - If we don't specify a type attribute in an **xsd:element**, the structure of the element will be described directly following the tag. Example: company (root) element
  - The default is exactly one occurrence.

XML Schema

- **6. Specifying Keys:**
  - For specifying **primary keys**, the tag **xsd:key** is used.
  - For specifying **foreign keys**, the tag **xsd:keyref is** used.
  - The **xsd:unique** tag, specifies elements that correspond to unique attributes in a relational database, that are not primary keys.
  - Such uniqueness constraints can be given a name and must also specify **xsd:selector** and **xsd:field** tags to identify the element type that contains the unique element and the element name within it that is unique, via the **xpath** attribute
  - When specifying a foreign key:
    (1) the attribute refer of the **xsd:keyref** tag specifies the referenced primary key
    (2) the tags **xsd:selector** and **xsd:field** specify the referencing element type and foreign key

XML Schema

- **7. Specifying the structures of complex elements via complex types:**
  - Complex elements in our example are *Department, Employee, Project,* and *Dependent*, which use the tag **xsd:complexType**. We specify each of these as a **sequence of subelements** corresponding to the database attributes of each entity type by using the **xsd:sequence** and **xsd:element** tags of XML schema. Each element is given a **name** and **type** via the corresp. attributes **name** and **type** of xsd:element.
  - We can also specify **minOccurs** and **maxOccurs** attributes if we need to change the default of exactly one occurrence. For (optional) database attributes where null is allowed, we need to specify minOccurs = 0, whereas for multivalued database attributes we need to specify maxOccurs = “unbounded” on the corresponding element.

XML Schema

- **8. Composite (compound) attributes:**
  - Composite attributes from an ER Schema are also specified as complex types in the XML schema, as illustrated by the **Address**, **Name**, **Worker**, and **WorksOn** complex types.
  - Another option is that these could have been directly embedded within their parent elements.

XML Documents and Databases

- Approaches to Storing/Retrieve XML Documents
  - (1) Using a DBMS to store the documents as text:
    - We can use a relational or object DBMS to store whole XML documents as text fields within the DBMS records or objects. This approach can be used if the DBMS has a special module for document processing, and would work for storing schemaless and document-centric XML documents.
  - (2) Using a DBMS to store the document contents as data elements:
    - This approach would work for storing a collection of documents that follow a specific XML DTD or XML schema. Since all the documents have the same structure, we can design a relational (or object) database to store the leaf-level data elements within the XML documents. We need mapping algorithms to design a database schema that is compatible with the XML document structure.

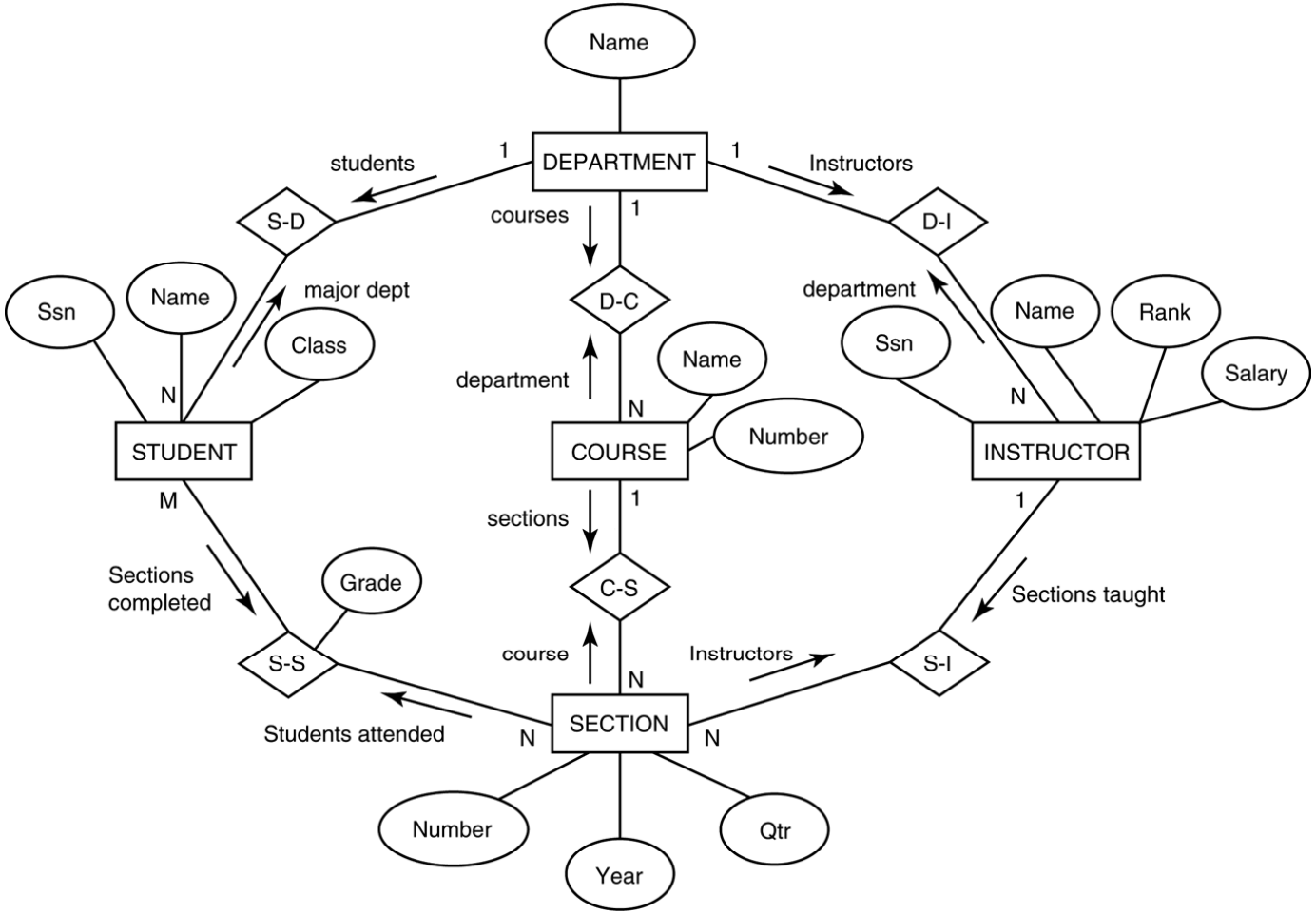
XML Documents and Databases

- Approaches to Storing/Retrieve XML Documents
  - (3) Designing a specialized system for storing native XML data:
    - A new type of database system based on the hierarchical (tree) model could be designed and implemented. **Native XML DBMS**. The system would include specialized indexing and querying techniques, and would work for all types of XML documents.
  - (4) Creating or publishing customized XML documents from pre-existing relational databases:
    - Because there are enormous amounts of data already stored in relational databases, parts of these data may need to be formatted as documents for exchanging or displaying over the Web. Use a separate software that would handle the conversions needed.

Extracting XML Documents from Relational Databases.

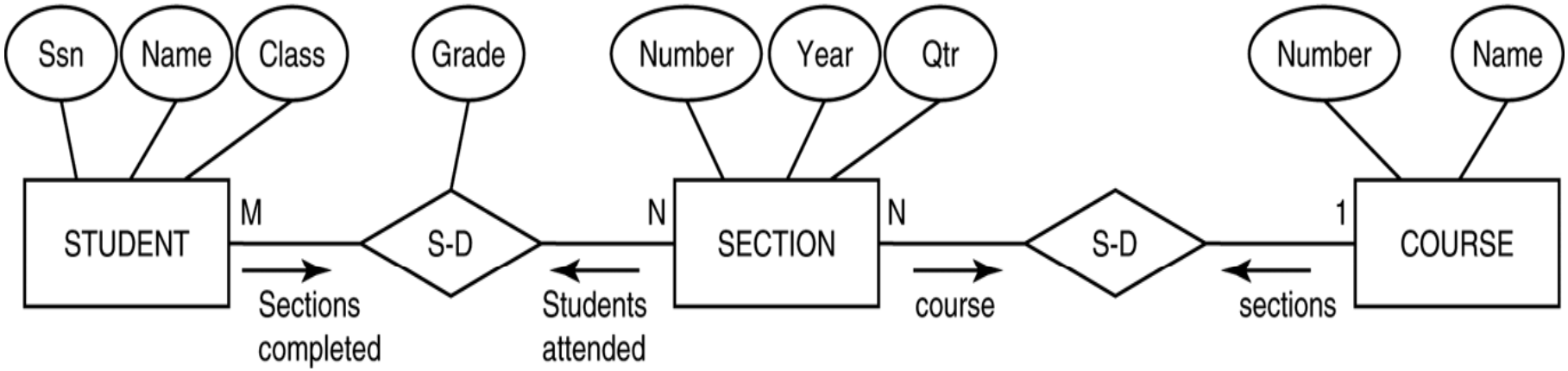
- Issues arising when converting (flat) relational data into XML documents (hier. tree model)
- Suppose that an application needs to extract XML documents for student, course, and grade information from the university database.
- The data needed for these documents is contained in the database attributes of the entity types ***course***, ***section***, and ***student*** as shown below (part of the main ER), and the relationships s-s and c-s between them.

ER schema diagram for the UNIVERSITY database

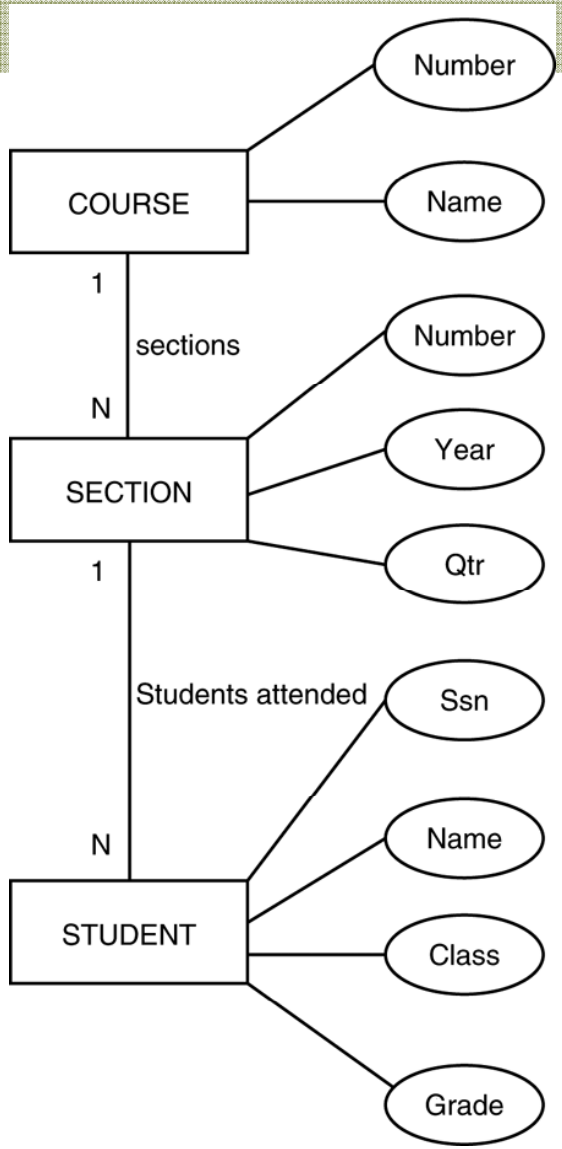


Subset of the UNIVERSITY database schema

- In general, most documents extracted from a database will use only a subset of the entire database schema.
  Here we use the following subset:



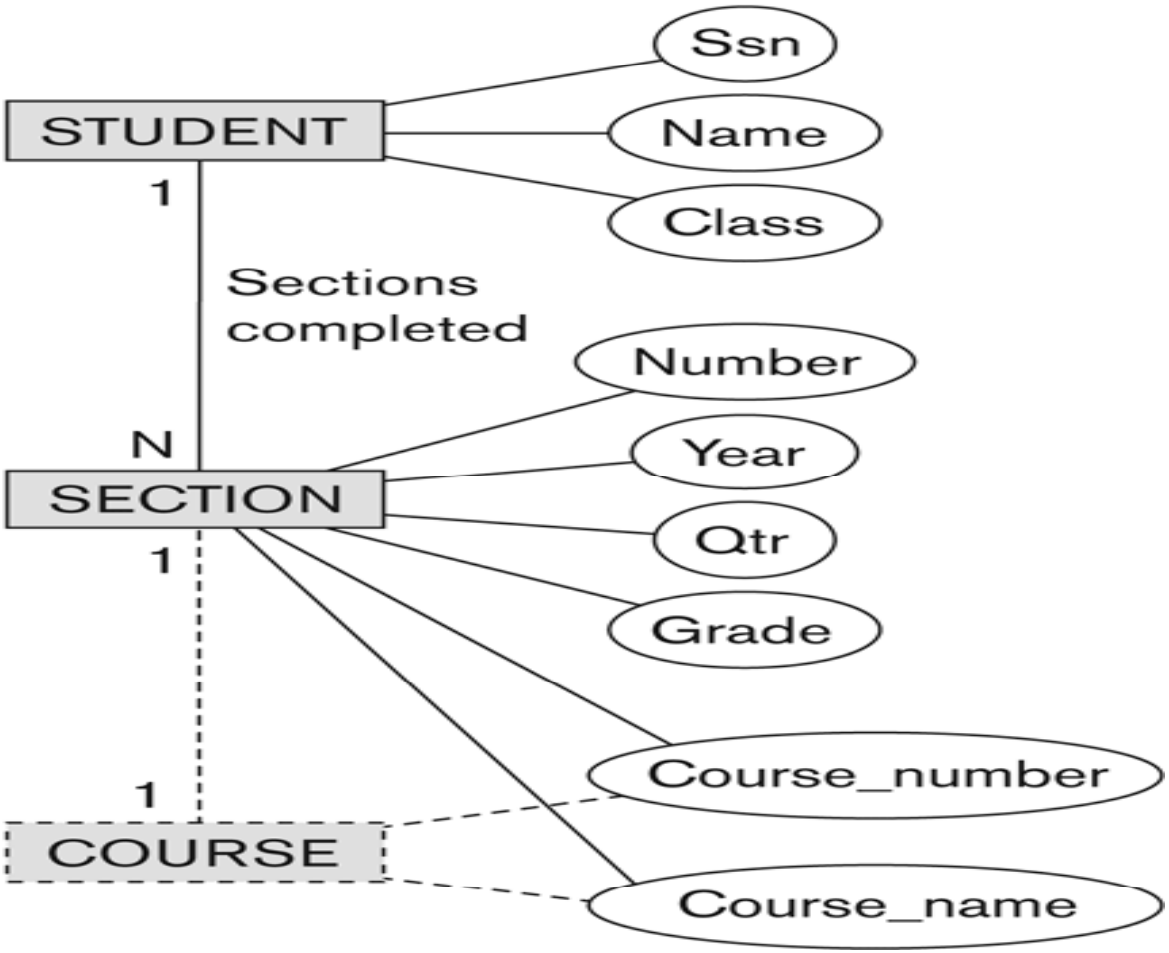
Hierarchical (tree) view with COURSE as the root



XML schema document with COURSE as the root

```
<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>
```

Hierarchical (tree) view with STUDENT as the root



XML schema document 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>
```

XML schema
stude