PAYAP UNIVERSITY

Structured, semistructured, unstructured data and XML

Vaidė Narváez

Computer Information Systems

February 1st, 2011

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

Motivation

- Although HTML is widely used for formatting and structuring Web documents, it is NOT suitable for specifying structured data that is extracted from databases.

- *XML (eXtended Markup Language)* has emerged as the standard for structuring and exchanging data over the Web.

- The formatting aspects are specified separately by using a formatting language *XSL (eXtended Stylesheet Language)*.

© 2009 Vaidė Narváez

Data

- *Structured data*: the information stored in databases is known as structured data because it is represented in a strict format. The DBMS checks to ensure that all data follows the constraints specified in the schema.

- *Semistructured data*: the schema information is mixed in with the data values. (*self-describing data, schemaless data*)

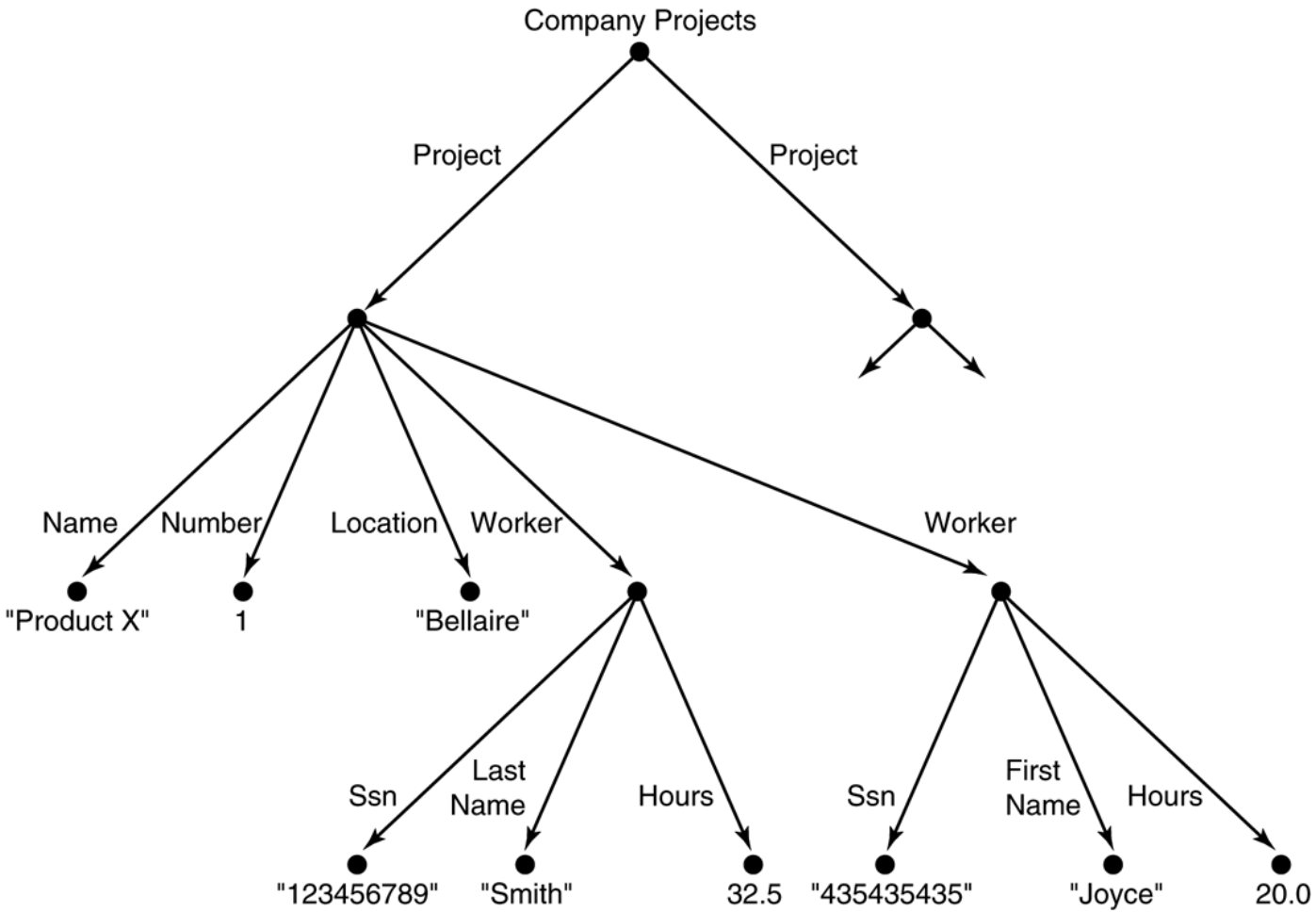
- *Unstructured data*: limited indication of the type of data. (e.g. text document, HTML pages)

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

Semistructured data

Semistructured data is often displayed as a directed graph or a tree.



© 2009 Vaidė Narváez

XML: Hierarchical (tree) data model

- Becoming the standard for E-Commerce data exchange

- A markup language (like HTML)
  - Uses elements, tags, attributes.
  - Includes document type declarations (DTDs), XML schemas, comments, entities, identifiers and references.

- XML Schema (XSD) replacing DTDs

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML documents

- XML documents are composed of elements and attributes. Attributes in XML provide additional information that describe elements.
- Elements are identified in a document by their **start tag** and **end tag**.
- **Complex** elements are constructed from other elements *hierarchically*, whereas **simple** elements contain data values.

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML example

```
<?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>
```

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML documents

- *Data-centric XML documents:*
  These documents have many small data items that follow a specific structure, and hence may be extracted from a structured database. They are formatted as XML documents in order to exchange them or display them over the Web.

- *Document-centric XML documents:*
  These are documents with large amounts of text, such as news articles or books. There is little or no structured data elements in these documents.

- *Hybrid XML documents:*
  These documents may have parts that contains structured data and other parts that are predominantly textual or unstructured.

© 2009 Vaidė Narváez

Well-Formed and Valid documents

- Well-Formed
  - It must start with an XML declaration to indicate the version of XML being used.
  - It must follow the syntactic guidelines of the tree model.
  - A well-formed XML document is syntactically correct. This allows it to be processed by generic processors that traverse the document and create an internal tree representation.

- Valid
  - Must be well-formed, AND the element names used in the start and end tag pairs must follow the structure specified in a separate **XML DTD** *(Document Type Definition)* file or **XML schema** file.

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML DTD example

```
<!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)>
] >
```

© 2009 Vaidė Narváez

XML DTD

- * means that the element can be repeated zero or more times in the document.
- + means that the element can be repeated one or more times in the document.
- ? means that the element can be repeated zero or one times.
- An element appearing without any of the preceding three symbols must appear exactly once in the document.
- The type of the element is specified via parentheses following the element.
  - If the parentheses include names of other elements, these would be the children of the element in the tree structure.
  - If the parentheses include the keyword *#PCDATA* or one of the other data types available in XML DTD, the element is a leaf node.

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

Limitations of DTD

1. the data types are not very general.

2. DTD has its own special syntax and so it requires specialized processors.

3. All DTD elements are always forced to follow the specified ordering of the document. Unordered elements are not permitted!

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML Schema

Schema is a record definition, analogous to the *CREATE SQL statement*, and therefore provides metadata

```
<xsd:schema xmlns:xsd="http:////www.w3.org/2001/XMLSchema">
  <xsd:element name="Salesperson" type="SalespersonType"/>
  <xsd:complexType name="SalespersonType">
          <xsd:sequence>
          <xsd:element name="SalespersonID" type="xsd:integer"/>
          <xsd:element name="SalespersonName"
          type="xsd:string"/>
          <xsd:element name="SalespersonTelephone"
          type="xsd:string"/>
              <pattern value="(?\d{3})?-?\d{3}-?\d{4}"/>
                  <xsd:element name="SalespersonFax"
                  type="xsd:string" minOccurs="0"/>
              <pattern value="(?\d{3})?-?\d{3}-?\d{4}"/>
              <xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML Schema

```
<?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>
```

© 2009 Vaidė Narváez

XML Schema

```
    <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>
```

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML 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="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>
```

© 2009 Vaidė Narváez

XML Schema

```
    </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>
```

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML and Databases

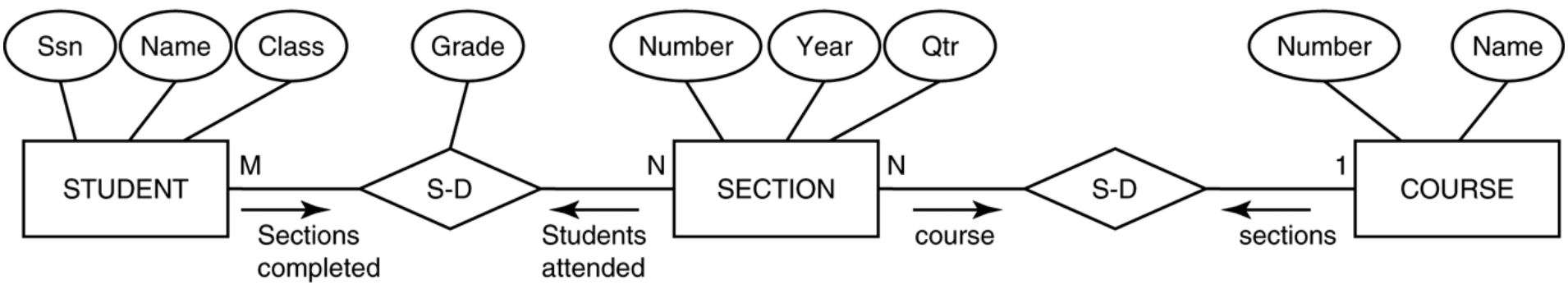
Approaches to storing XML Documents:

- Using a dbms to store the documents as text
- Using a dbms to store the document contents as data elements
- Designing a specialized system for storing native XML data
- Creating or publishing customized XML documents from pre-existing relational databases

© 2009 Vaidė Narváez

XML and Databases

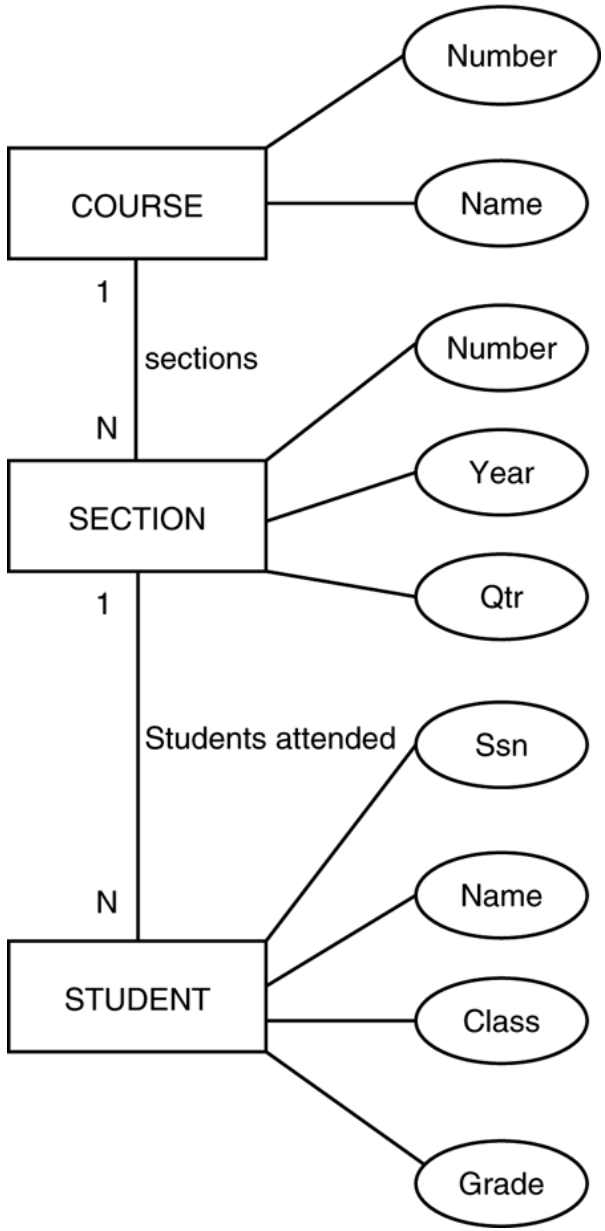
- Suppose that an application needs to extract XML documents for student, course, and grade information from the university database. The data needed for these documents is contained in the database attributes of the entity types COURSE, SECTION and STUDENT.



© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML and Databases



© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XML and Databases

```
<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>
```

© 2009 Vaidė Narváez

XPath

- XPath provides language constructs for specifying path expressions to identify certain elements within an XML document that match specific patterns.

- There are two main separators when specifying a path:
  - single slash (/): specifies that the tag must appear as a direct child of the previous (parent) tag
  - double slash (//): specifies that the tag can appear as a descendant of the previous tag at any level

```
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]
```

© 2009 Vaidė Narváez

PAYAP UNIVERSITY

XPath Example

/furniturecompany/product[0]	Selects the first product element that is the child of furniturecompany.
/furniturecompany/product[standardprice > 300.00]	Selects all product elements with a standardprice greater than $300.00.
/furniturecompany/product/ [standardprice > 150.00]/description	Selects the description of all product elements with a standardprice greater than $150.00.
/furniturecompany/product/*	Uses a wildcard, "*", to match any element node. In this case, all product elements would be returned.

© 2009 Vaidė Narváez