ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στην Επιστήμη και Τεχνολογία των Υπηρεσιών

Ενότητα 5: Schema - 1

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο v. 3.0

(Attribution – Non Commercial – Non-derivatives)

- Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

XML Schema
605.444 / 635.444

David Silberberg

Lecture 5

Problems with Document Type Definitions (DTDs)

- XML came out of the SGML world
  - So, too, DTDs came out of the SGML world
  - DTDs are very useful for document definitions
  - However, XML has become more than a document language
    - General data exchange mechanism
      - Across applications
      - Across platforms
      - Wireless
      - Multimedia
      - Binary files
    - SGML and specifically DTDs were not created to handle all these data types
  - DTDs are still somewhat useful for general XML definitions

Limitations of DTDs

- Only one DTD can be associated with an XML document
  - SGML intended for there to be one central document definition
  - In the world of XML, documents can employ elements from multiple domains
    - Parts - Suppliers - Customers
    - Students - Faculty - Administration
  - In this environment, a centralized definition of elements and attributes is not practical, desirable, or reasonable
    - Cannot easily incorporate legacy systems
    - Cannot implement B2B or Enterprise systems
    - Need inter-corporation and inter-agency cooperation to achieve (not likely)
  - A mechanism is needed to enable the definition of an XML document from different sources

Limitations of DTDs (2)

- DTDs are not extensible
  - Unlike eXtensible Markup Language documents, which DTDs describe
    - XML syntax is not required to define XML documents
    - However, there is an elegant symmetry if the definition language of XML is defined using the XML syntax
  - Would like some way to add to the definition of a document
    - Parent banking institution uses a standards XML definition
    - Fred's bank, which has giveaways (like toasters), needs to augment the XML syntax a little
      - DTDs allow this to be done within a specific document
      - However, if multiple banks decide to give away toasters, these banks also need to augment their DTDs
      - It is poor practice to define the data in the data document itself

Limitations of DTDs (3)

- DTDs do not support Namespaces in a reasonable manner
  - In fact, Namespaces do not make sense in the context of DTDs
    - Namespaces resolve conflicts between named elements (and maybe attributes)
    - DTDs specify definitions in a single document
    - Thus, unique names can be defined to circumvent conflicts
  - Namespaces assume that there are multiple definitions
    - Each definition is associated with a unique namespace
    - There is not a pressing need to define multiple namespaces within a single document
  - If namespaces are defined with respect to DTDs, they must be specified in a single document
    - Defeats the purpose of namespaces
    - DTDs become unwieldy

Limitations of DTDs (4)

- DTDs hardly define datatypes
  - SGML was meant to work with document text
    - XML, however, has rapidly evolved to work with many datatypes
    - Thus, DTDs, which were meant for text documents, are limiting to general XML documents
  - Types of data used now
    - Integer
    - Float
    - Decimal
    - Date/time
    - Duration
    - Byte, hexadecimal, etc.
    - Binary
    - String templates
    - Etc., etc., etc.

Limitations of DTDs (5)

- No inheritance
  - Cannot derive data types from other data types
  - Cannot construct complex types from
    - Simple types
    - Other complex types
  - (Schema enables data type inheritance, not conceptual inheritance.)
- Internal DTDs are a bad idea
  - Internal DTDs can augment external DTDs (OK)
  - Internal DTDs can also override external DTDs (bad)
    - Standard exchange structures (and protocols) may be ignored
    - Defeats the whole purpose of standardization

Limitations of DTDs (6)

- DTDs do not represent all the elements of XML documents
  - Processing Instructions (PIs) are not represented
  - DTDs represent only the information in the data portion of a document
  - However, one would like to specify that PIs are necessary, if they are indeed necessary
    - Perhaps, one needs to specify that a specific type of stylesheet is necessary
    - Perhaps, certain classes of parsers are necessary
    - PIs are necessary to identify such specifications
    - Thus, PIs may need to be defined

Schema Addresses the Limitations of DTDs

- Like all other World Wide Web Consortium (W3C) specifications, the Schema specification and other supporting documents can be found at:
  - http://www.w3c.org/
  - Can easily find Schema documents from the home page
- Syntax is XML
  - Makes Schema specifications easy to parse
  - DOM tools support reading Schema documents
  - Can use other tools like XPath and Namespaces, etc.
  - Easy to interpret and understand

Schema Benefits (1)

- Multiple Schema definitions can be associated with an XML document
  - Documents can employ elements from multiple domains
    - Parts - Suppliers - Customers
    - Students - Faculty - Administration
  - Supports distributed environments
    - Can incorporate legacy systems
    - Can be used to implement B2B or Enterprise systems
    - Does not need inter-corporation and inter-agency cooperation
    - Can use multiple standards

Schema Benefits (2)

- Schema definitions are extensible
  - Enables more detailed definitions of Schemas
    - Enables more robust definitions within a sub-schema
    - Often, general standards are developed by a company or agency
    - Schema enables corporate subdivisions to define comprehensive standards for sub-areas
      - Spacecraft design
      - Mechanical design
  - Enables the extension of certain types
    - Integer -> Positive Integer
    - Date -> Valid Date
  - Enables complex type definitions
    - From primitive types
    - From other complex types

Schema Benefits (3)

- Support for namespaces
  - Can incorporate definitions from different domains
  - Can resolve identical element names from different definitions
- Support for datatypes
  - Support for general datatypes such as integer, decimal, date, time, binary, hex, string, etc.
  - Support for XML-specific datatypes such as URIs, URLs, IDREFs, NMTOKENs, etc.
- Eliminates internal definitions
  - Separates data and definition
  - Allows others to access special (or overriding) definitions

Example

```
<?xml version="1.0"?>
<store xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="store.xsd">
      <cust_list>
          <customer cid="C-1" >
              <name title="Mr.">
                  <first>Sam</first>
                  <middle>Ta</middle>
                  <middle>Chuan</middle>
                  <last>Chu</last>
              </name>
              <address>
                  <street>123-A Kensington Circle</street>
                  <city>London</city><country>England</country>
              </address>
              <purchase date="2002-01-12" items="I-34 I-62 I-15" qty="3 1 1" />
              <purchase date="2002-01-13" items="I-15" qty="2" />
          </customer>
```

Example (2)

```
        <customer cid="C-2" ctype="bad">
            <name title="Dr.">
                <first>Darsana</first>
                <last>Sudarsen</last>
            </name>
            <address>
                <street>11100 Johns Hopkins</street>
                <city>Baltimore</city><state>MD</state>
                <zip>21207</zip>
            </address>
            <purchase date="2002-02-02" items="I-62 I-5" qty="2 1" />
        </customer>
    </cust_list>
    <inventory_list>
        <item item_no="I-5" supplier_id="S-1">
            <description>Ivory Soap</description>
            <in_stock>50</in_stock>
            <price>1.09</price>
            <cost>.28</cost>
        </item>
```

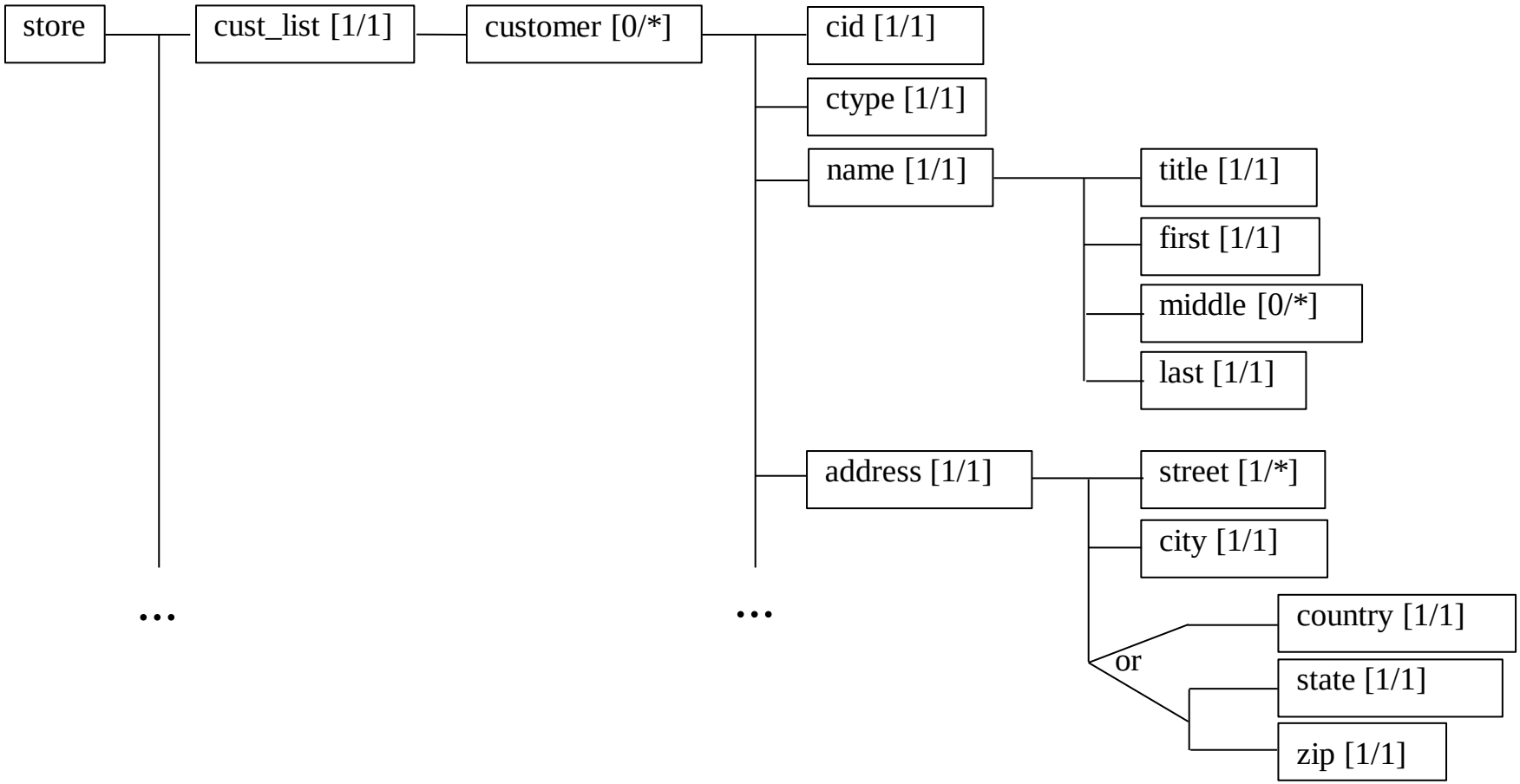
Example (3)

```
<item item_no="I-15" supplier_id="S-1">
        <description>Cheer Detergent</description>
        <in_stock>37</in_stock>
        <price>5.98</price>
        <cost>2.10</cost>
</item>
<item item_no="I-34" supplier_id="S-1">
        <description>Bounty Paper Towels</description>
        <in_stock>112</in_stock>
        <price>1.48</price>
        <cost>.57</cost>
</item>
<item item_no="I-62" supplier_id="S-2">
        <description>Sunshine Tissues</description>
        <in_stock>320</in_stock>
        <price>1.65</price>
        <cost>.98</cost>
</item>
</inventory_list>
```

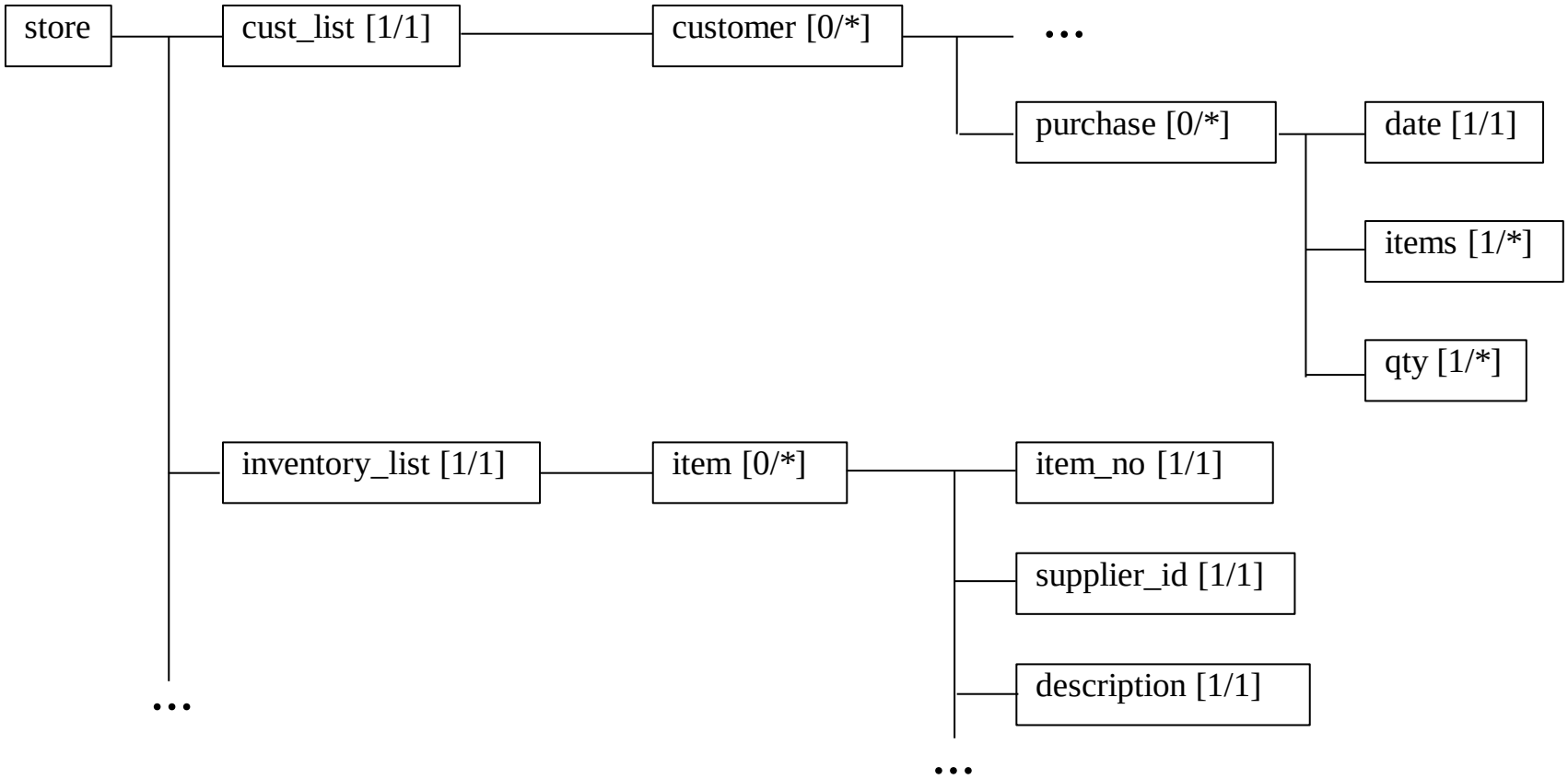
Example (4)

```
    <supplier_list>
        <supplier sid =“S-1”>
            <company>Proctor and Gamble</company>
            <telephone>1-800-PAMPERS</telephone>
        </supplier>
        <supplier sid =“S-2”>
            <company>Sunshine Products</company>
            <telephone>1-888-344-9881</telephone>
        </supplier>
    </supplier_list>
</store>
```

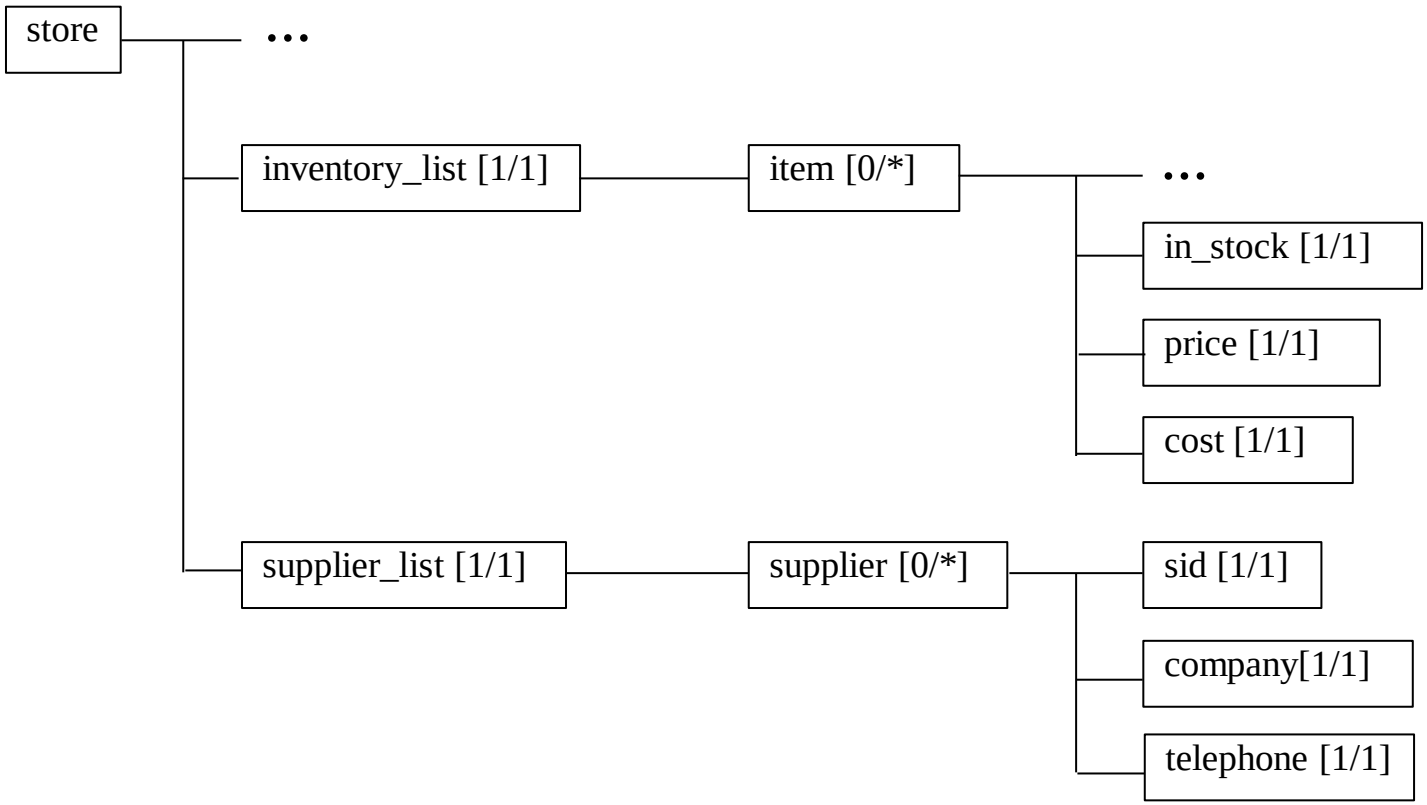
Data Model


store
cust_list [1/1]
customer [0/*]
cid [1/1]
ctype [1/1]
name [1/1]
title [1/1]
first [1/1]
middle [0/*]
last [1/1]
address [1/1]
street [1/*]
city [1/1]
country [1/1]
or
state [1/1]
zip [1/1]
...
...

Data Model (2)


store
cust_list [1/1]
customer [0/*]
...
purchase [0/*]
date [1/1]
items [1/*]
qty [1/*]
inventory_list [1/1]
item [0/*]
item_no [1/1]
supplier_id [1/1]
description [1/1]
...
...

Data Model (3)


store
...
inventory_list [1/1]
item [0/*]
...
in_stock [1/1]
price [1/1]
cost [1/1]
supplier_list [1/1]
supplier [0/*]
sid [1/1]
company[1/1]
telephone [1/1]

XML Schema Definition

- In Schema, everything is built from the bottom up
- High-level Schema declarations do not make sense without knowledge of the datatype declarations
- Thus, we start with datatypes
- Datatypes
  - Two flavors: simple and complex
  - Simple: Built-in types
  - Complex
    - Derived from simple and other complex types
    - Can define data structures

Simple Types

- string
- token
- byte
- integer
- positive Integer
- negative Integer
- unsignedInt
- long
- short
- decimal (1.33, 4.0, 4)
- float (12.3E-3, INF, NaN)
- double
- boolean (true, false, 1, 0)
- time (13:20:00.000)
- date (2001-10-24)
- Name
- QName (nmspc:element)
- anyURI
- language (en-US)
- ID
- IDREF
- NMTOKEN

Simple Schema Definition

- Standard XML Schema Definition (xsd) file header:

  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  …

  </xsd:schema>

  – All schema terms must be prefixed by xsd (standard abbreviation)
    - <xsd:element> … </xsd:element>
    - <xsd:attribute> … </xsd:attribute>

- It is easier if this is the default namespace

  <schema xmlns="http://www.w3.org/2001/XMLSchema">

  …

  </schema>

Simple Type Definitions - Min and Max

- Min and max values:

```
<simpleType name="nonNegativeDecimal" >
    <restriction base="decimal">
        <minInclusive value="0" />
    </restriction>
</simpleType>
```

- Schema definition using simple type

```
<element name="price" type="nonNegativeDecimal" />
```

- XML document instance

```
<price>1.09</price>
```

Enumeration

- Enumeration

```
<simpleType name="USState" >
    <restriction base="string">
        <enumeration value="AK" />
        <enumeration value="AL" />
        <!-- etc. -->
    </restriction>
</simpleType>
```

- Schema definition using simple type

```
<element name="state" type="USState" />
```

- XML document instance

```
<state>MD</state>
```

Enumeration (2)

- Enumeration

```
<simpleType name="Title" >
    <restriction base="string">
        <enumeration value="Mr." />
        <enumeration value="Ms." />
        <enumeration value="Mrs." />
        <enumeration value="Miss" />
        <enumeration value="Dr." />
    </restriction>
</simpleType>
```

- Schema definition using simple type

```
<attribute name="title" type="Title" />
```

- XML document instance

```
<name title="Mr.">…</name>
```

Patterns

- Patterns:

```
<simpleType name="StdTelNum" >
    <restriction base="string">
        <pattern value="\d{1}\-\d{3}\-\d{3}\-\d{4}" />
    </restriction>
</simpleType>

<simpleType name="StringTelNum" >
    <restriction base="string">
        <pattern value="\d{1}\-\d{3}\-[A-Z0-9]{7}" />
    </restriction>
</simpleType>
```

Patterns (2)

- Schema definition using simple type

  <element name="telephone" type="StdTelNum" />
- XML document instance

  <telephone>1-888-344-9881</telephone>

- Schema definition using simple type

  <element name="telephone" type="StringTelNum" />
- XML document instance

  <telephone>1-800-PAMPERS</telephone>

More Patterns

- Chapter \d - Chapter 0, Chapter 1, …
- Chapter\s\d - Chapter followed by whitespace character (space, tab, newline, etc.) followed by a single digit
- Appendix\s\w - Appendix followed by whitespace character followed by a word character (letter or digit)
- a*x - x, ax, aax, aaax, …
- a?x - x, ax
- a+x - ax, aax, aaax, …
- (a|b)+x - ax, bx, aax, abx, bax, bbx, aaax, aabx, abax, abbx, etc.
- [aeiou]x - ax, ex, ix, ox, ux

More Patterns

- [d-f]x - dx, ex, fx
- [\-df]x - -x, dx, fx
- [^0-9]x - non-digit character followed by x
- \Dx - non-digit character followed by x
- .x - any character followed by x
- .*a.* - 0 or more characters followed by a followed by 0 or more characters
- ab{2}x - abbx
- ab(2,4)x - abbx, abbbx, abbbbx
- ab(2,)x - abbx, abbbx, abbbbx, abbbbbx, …
- (ab){2}x - ababx

Unions

- Union of types

```
<simpleType name="AllTelNum" >
    <union memberTypes="StdTelNum StringTelNum">
</simpleType>
```

- Schema definition using simple type

```
<element name="telephone" type="AllTelNum" />
```

- XML document instances

```
<telephone>1-888-344-9881</telephone>
<telephone>1-800-PAMPERS</telephone>
```

List Types

- Simple type - zipCode

```
<simpleType name="ZipCode" >
    <restriction base="string">
        <pattern value="[0-9]{5}" />
    </restriction>
</simpleType>
```

- Alternatively

```
<simpleType name="ZipCode" >
    <restriction base="integer">
        <minExclusive value="00000" />
        <maxInclusive value="99999" />
    </restriction>
</simpleType>
```

  - Actually, this will not work because it will accept '35' as a zip code

Simple Type Usage

- Schema definition using simple type

```
<element name="zip" type="ZipCode" />
```

- XML document instances

```
<zip>21207</zip>
<zip>20723</zip>
```

- Zip+4 definition

```
<simpleType name="ZipPlus4" >
    <restriction base="string">
        <pattern value="[0-9]{5}(\-[0-9]{4})?" />
    </restriction>
</simpleType>
```

List Type Creation

- List of Zip Codes

```
<simpleType name="ZipCodeList" >
    <list itemType="ZipCode" />
</simpleType>
```

- Schema definition of an element

```
<element name="zipList" type="ZipCodeList" />
```

- Usage in XML document

```
<zipList>20134  95123  78665</zipList>
```

More List Types

- List of three Zip Codes

```
<simpleType name="ThreeZipCodes">
    <restriction base="ZipCodeList" />
        <length value="3" />
    </restriction>
</simpleType>
```

- Schema definition of an element

```
<element name="limitedZipList" type="ThreeZipCodes" />
```

- Usage in XML document

```
<limitedZipList>20134  95123  78665</ limitedZipList >
```

Simple Elements and Attributes

- Elements and attributes with simple types:
  - <element name="first" type="string" />
  - <element name="in_stock" type="nonNegativeInteger" />
  - <attribute name="date" type="date" />
  - <attribute name="item_no" type="ID" />
  - <attribute name="items" type="IDREFS" />

Complex Types

- Define a complex Name type:

```
<complexType name="NameType">
    <sequence>
        <element name="first" type="string" />
        <element name="middle" type="string" />
        <element name="last" type="string" />
    </sequence>
    <attribute name="title" type="string" />
</complexType>
```

- Improvements to be made
  - Better data types
  - Cardinalities
  - IDs

Defining General Types

- Define a proper name and a mixed name

```
<simpleType name="ProperName">
    <restriction base="string">
        <pattern value="[A-Z][a-zA-Z]*" />
    </restriction>
</simpleType>
<simpleType name="MixedName">
    <restriction base="string">
        <pattern value ="[a-zA-Z0-9\-\s]*" />
    </restriction>
</simpleType>
```

- Define a title, as before

```
<simpleType name="Title" >
    <restriction base="string">
        <enumeration value="Mr." />
        <!-- etc. -->
    </restriction>
</simpleType>
```

A Better Name Definition

- Define a complex Name type:

```
<complexType name="NameType">
    <sequence>
        <element name="first" type="ProperName" />
        <element name="middle" type="ProperName" />
        <element name="last" type="ProperName" />
    </sequence>
    <attribute name="title" type="Title" />
</complexType>
```

- Still must represent cardinalities

Cardinality of Elements

- minOccurs
  - Can be any non-negative integer
  - If the value is 0, then the element is optional
  - If the value is 1 or more, then the element is required If
  - the value is 3, there must be at least three elements The
  - default value (when minOccurs is not declared) is 1
- maxOccurs
  - Can be any non-negative integer
  - The value must be greater than or equal to the minOccurs value
  - If the value is 5, there can be at most five elements
  - ‘unbounded’ is used to describe an unbounded number of occurrences

Attribute Cardinalities

- Attributes can only appear one time per element (at most)
- Attribute specification
  - ‘use’ attribute
  - Values
    - required - attribute must appear in element
    - optional - attribute may appear in element
    - prohibited - attribute may never appear in element

Default Attributes

- Used to set default value of attributes
- Only makes sense if attribute is optional
  - Proper use of default in Schema definition
    - <attribute name="…" use="optional" default="…">
  - Incorrect use of default in Schema definition
    - <attribute name="…" use="required" default="…">
    - <attribute name="…" use="prohibited" default="…">
- If the attribute's value appears in the XML document, the value specified in the document is used
- If the attribute's value does not appear in the XML document, the schema processor provides the default value

Default Elements

- Proper use of default in Schema Definition
  - <element name="city" default="Baltimore">
  - minOccurs and maxOccurs can be any value
- If element with value is present, the value is used
  - <city>Laurel</city>
- If element is present without the value, the default value is used
  - <city />
  - The value of city is Baltimore
- If element is not present, no value is assigned to the element

Fixed Values

- Applies to both attributes and elements
- It specifies value to be used
- If attribute or element is in document, the fixed value is used
- If attribute or element is not in document, the schema processor will supply the fixed value
  - <attribute name="country" use="optional" value="US" />
- Fixed and default attributes are mutually exclusive

A Better Name Definition

- Define a complex Name type:

```
<complexType name="NameType">
    <sequence>
        <element name="first" type="ProperName" />
        <element name="middle" type="ProperName"
            minOccurs="0" maxOccurs="unbounded"/>
        <element name="last" type="ProperName" />
    </sequence>
    <attribute name="title" type="Title"
        use="optional" default="Ms." />
</complexType>
```

Τέλος Ενότητας