SEMI-STRUCTURED DATA (XML)

CS561-SPRING 2012
WPI, MOHAMED ELTABAKH

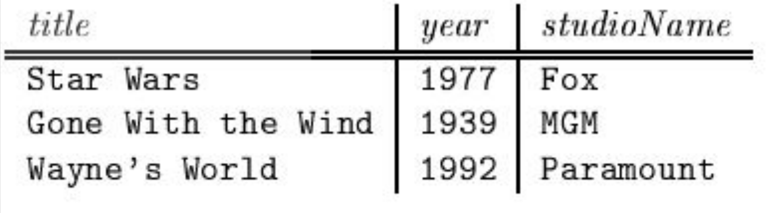
SEMI-STRUCTURED DATA

- **ER, Relational, ODL data models are all based on schema**
  - Structure of data is rigid and known is advance
  - Efficient implementation and various storage and processing optimizations

- **Semistructured data is schemaless**
  - Flexible in representing data
  - Different objects may have different structure and properties
  - Self-describing (data is describing itself)
  - Harder to optimize and efficiently implement

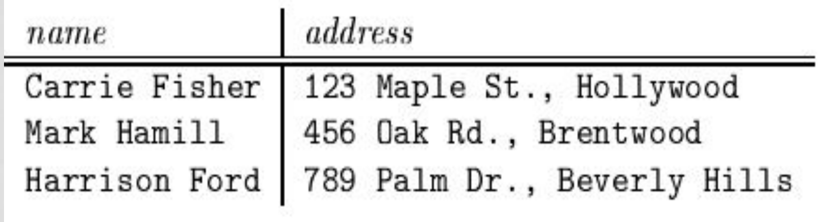
RELATIONAL MODEL FOR MOVIE DB

Collection of records (tuples)



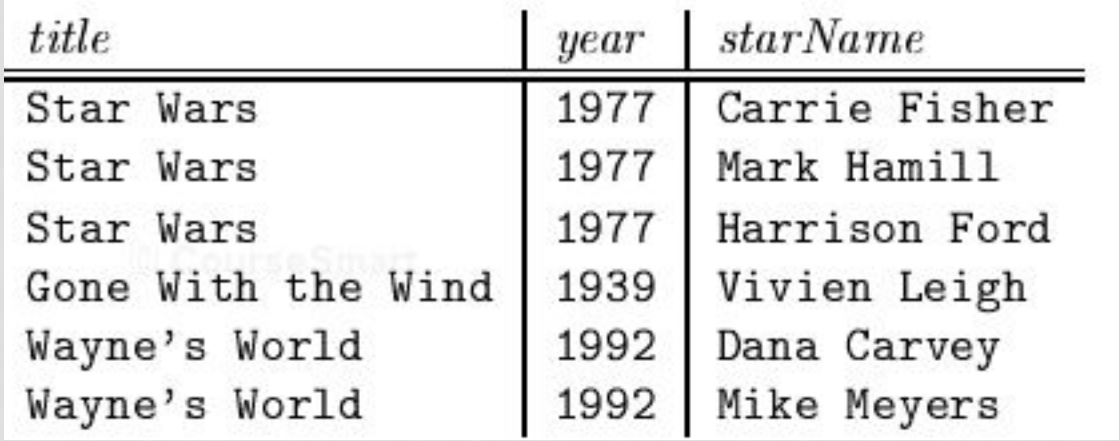
title	*year*	*studioName*
Star Wars	1977	Fox
Gone With the Wind	1939	MGM
Wayne's World	1992	Paramount

Movie



name	*address*
Carrie Fisher	123 Maple St., Hollywood
Mark Hamill	456 Oak Rd., Brentwood
Harrison Ford	789 Palm Dr., Beverly Hills

Star

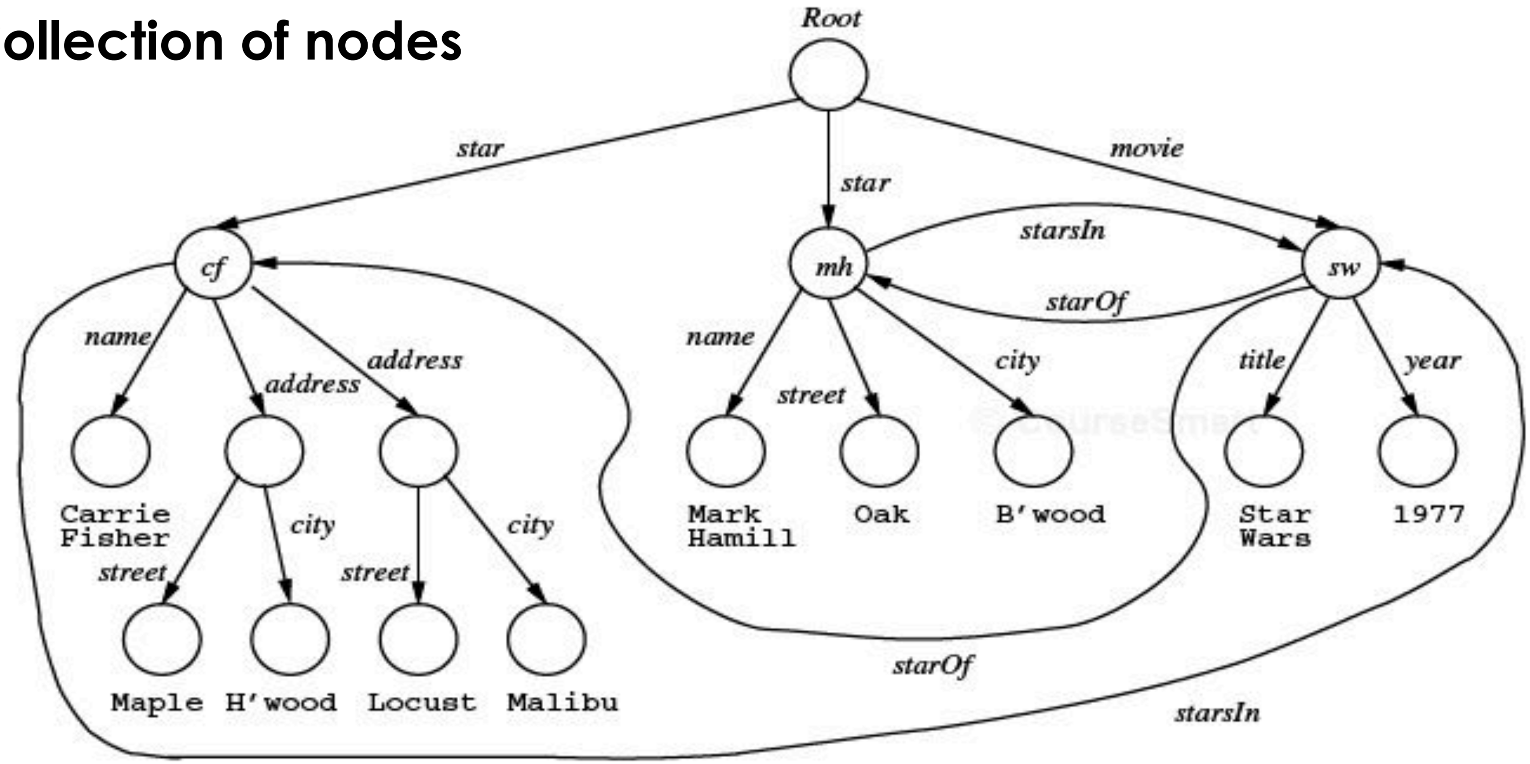


title	*year*	*starName*
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With the Wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

Stars-in Relationship

SEMI-STRUCTURED MODEL

Collection of nodes



- Leaf nodes contain data
- Internal nodes represent either ***objects*** or ***attributes***
- Each link is either an ***attribute link*** or ***relationship link***

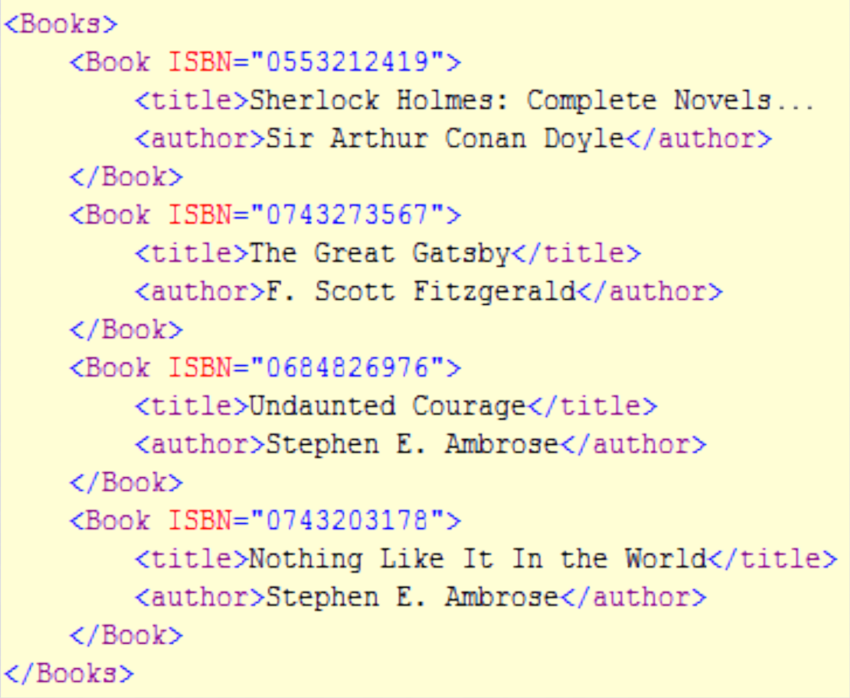
XML

- XML: Extensible Markup Language

- XML is a tag-based notation (language) to describe data

- **XML has two modes**
  - **Well-formed XML** ---No Schema at all

  - **Valid XML** --- governed by DTD (Document Type Definition)
    - Allows validation and more optimizations and pre-processing



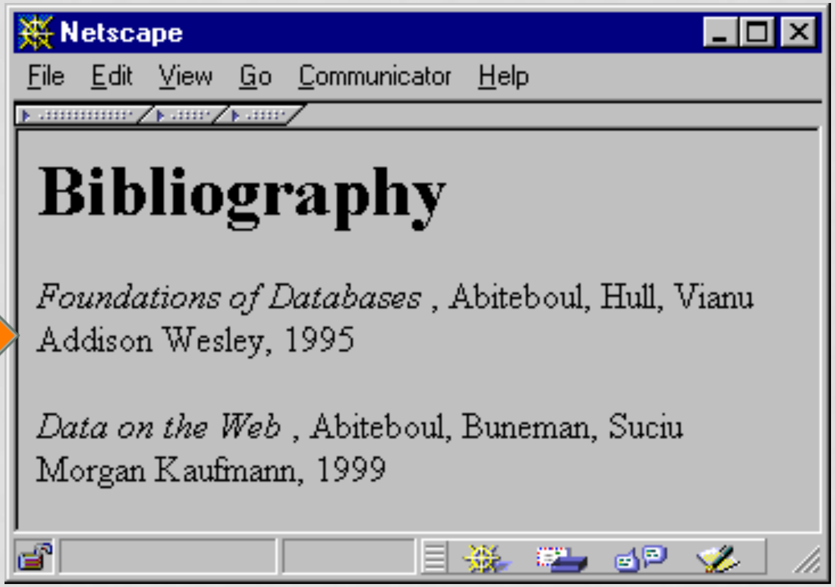
```xml
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

XML document

HTML TAGS VS. XML TAGS

- HTML tags describe structure/presentation

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>
        Abiteboul, Hull, Vianu
        <br> Addison Wesley, 1995
<p> <i> Data on the Web </i>
        Abiteboul, Buneman, Suciu
        <br> Morgan Kaufmann, 1999
```



Bibliography

Foundations of Databases , Abiteboul, Hull, Vianu
Addison Wesley, 1995

Data on the Web , Abiteboul, Buneman, Suciu
Morgan Kaufmann, 1999

HTML TAGS VS. XML TAGS (CONT'D)

- XML tags describe content (have semantics)

```
<bibliography
    <book>    <title> Foundations… </title>
              <author> Abiteboul </author>
              <author> Hull </author>
               <author> Vianu </author>
           <publisher> Addison Wesley </publisher>
               <year> 1995 </year>
    </book>
         …
</bibliography>
```

XML TERMINOLOGY

- **tags**: book, title, author, …
- **start tag**: <book>, end tag: </book>
- **elements**: <book>…</book>,<author>…</author>
- elements are nested
- empty element: <red></red> abbrv. <red/>
- an XML document: single *root element*

Well-formed XML document: if it has matching tags

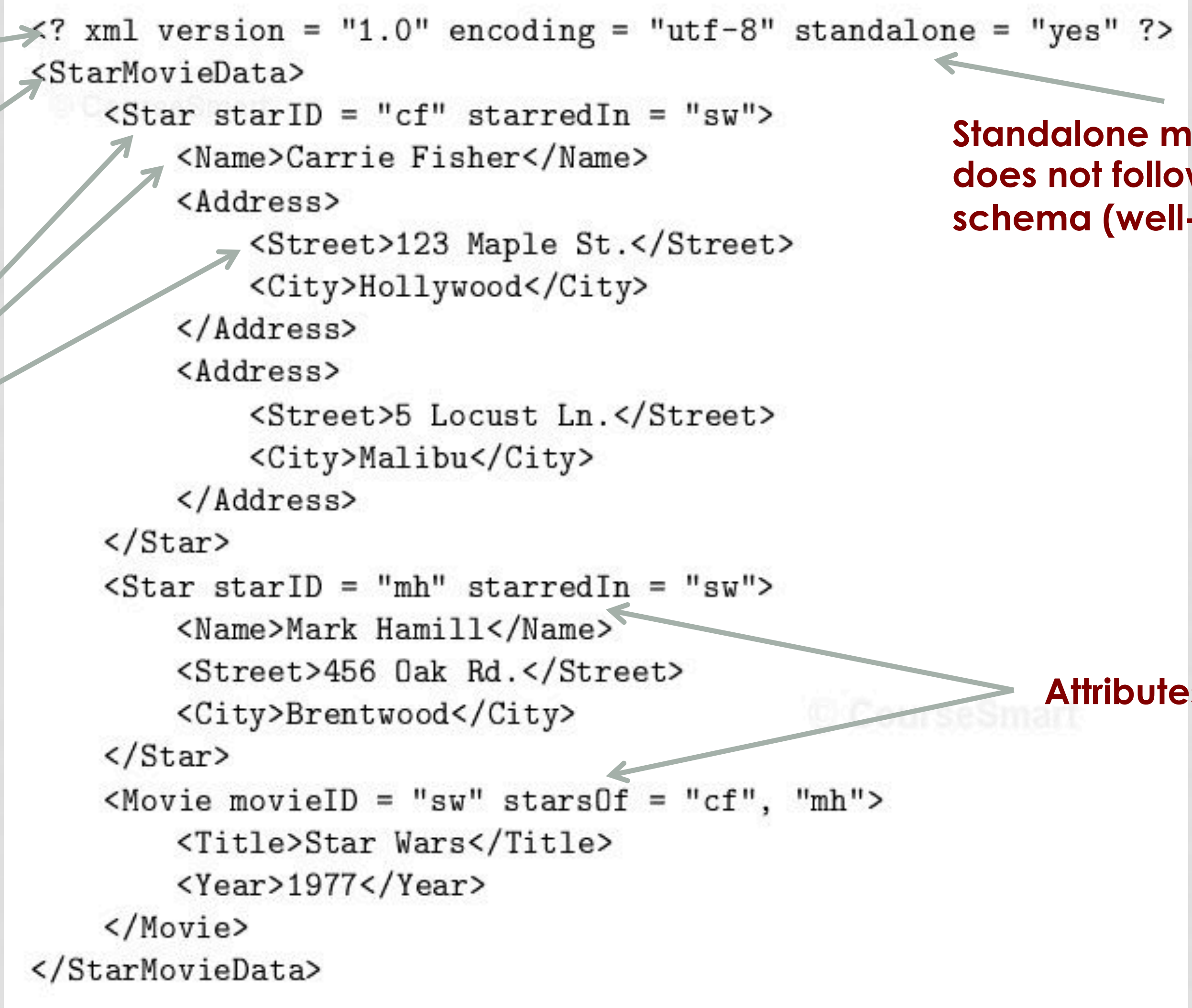
XML: ATTRIBUTES

Inside the start tag

```
<book price = “55” currency = “USD”>
  <title> Foundations of Databases </title>
  <author> Abiteboul </author>
   …
  <year> 1995 </year>
</book>
```

Attributes are alternative ways to represent data

SEMANTIC TAGS



```
<? xml version = "1.0" encoding = "utf-8" standalone = "yes" ?>
<StarMovieData>
    <Star starID = "cf" starredIn = "sw">
        <Name>Carrie Fisher</Name>
        <Address>
            <Street>123 Maple St.</Street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street>
            <City>Malibu</City>
        </Address>
    </Star>
    <Star starID = "mh" starredIn = "sw">
        <Name>Mark Hamill</Name>
        <Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie movieID = "sw" starsOf = "cf", "mh">
        <Title>Star Wars</Title>
        <Year>1977</Year>
    </Movie>
</StarMovieData>
```

Instructional tag (the doc. Is XML)

Root element

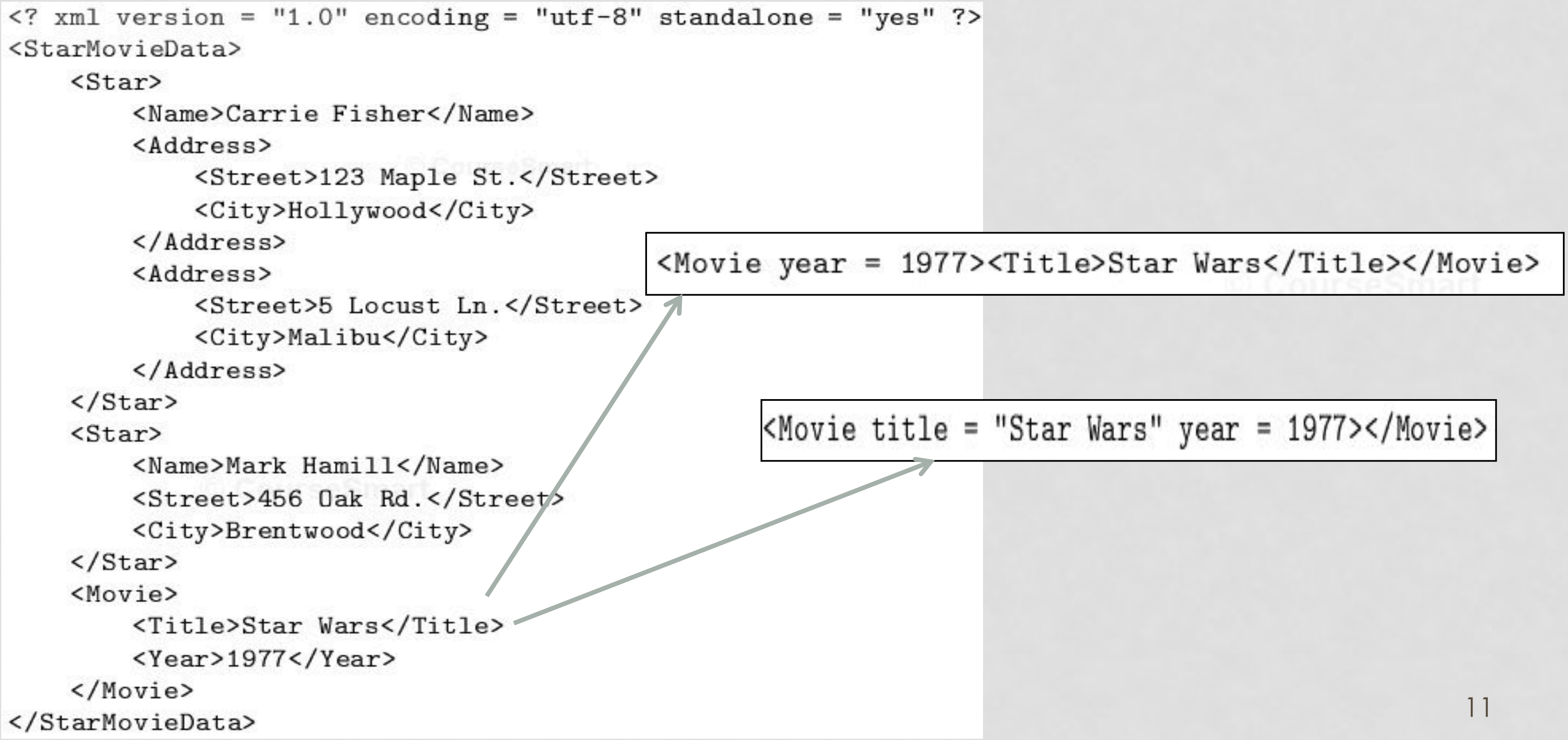
Sub elements

Standalone means it does not follow a schema (well-formed)

Attributes

ATTRIBUTES VS. SUB-ELEMENTS

- Two alternative ways to describe the attributes of an object
- Attributes are also used to define IDs and references

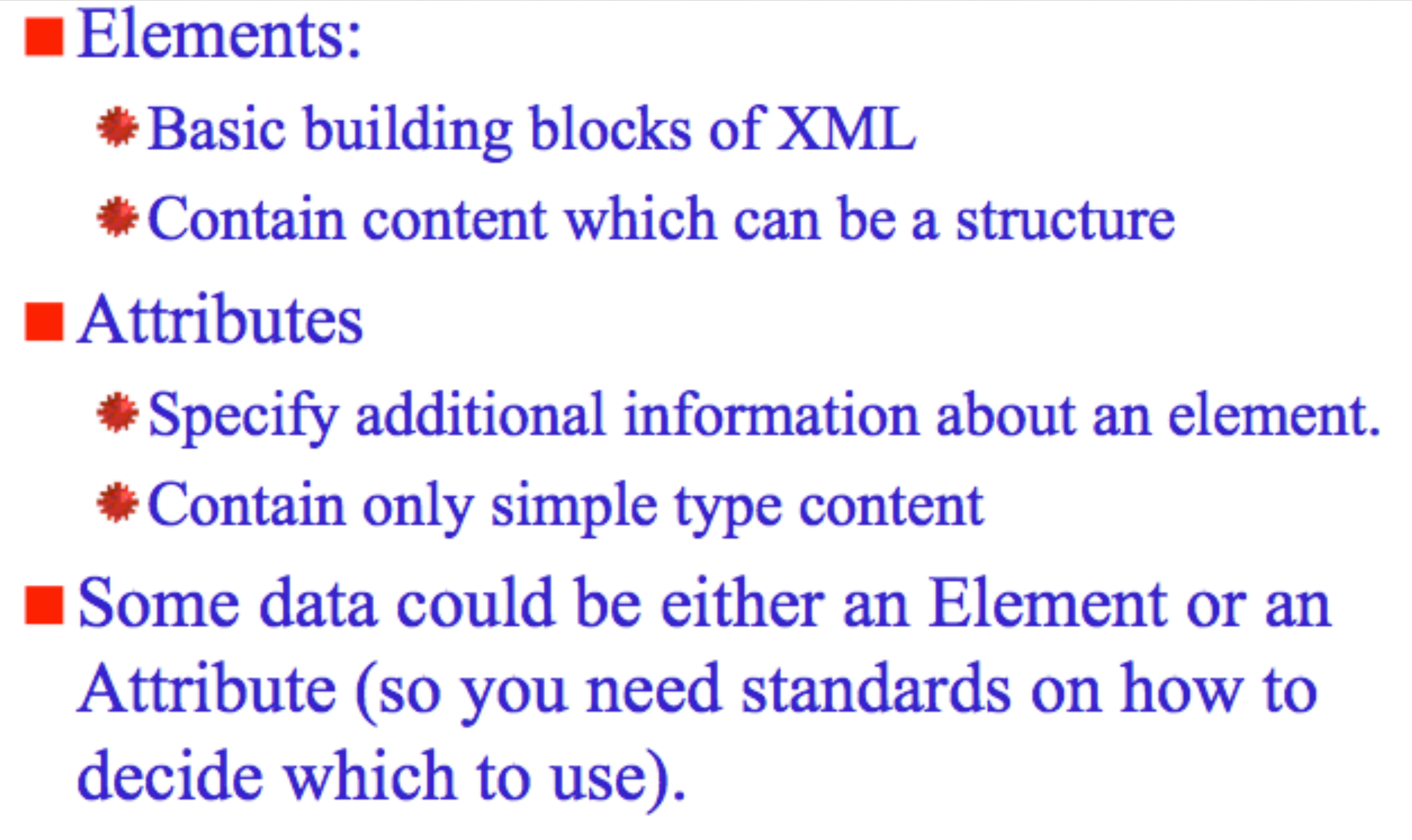


```
<? xml version = "1.0" encoding = "utf-8" standalone = "yes" ?>
<StarMovieData>
    <Star>
        <Name>Carrie Fisher</Name>
        <Address>
            <Street>123 Maple St.</Street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street>
            <City>Malibu</City>
        </Address>
    </Star>
    <Star>
        <Name>Mark Hamill</Name>
        <Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie>
        <Title>Star Wars</Title>
        <Year>1977</Year>
    </Movie>
</StarMovieData>
```

```
<Movie year = 1977><Title>Star Wars</Title></Movie>
```

```
<Movie title = "Star Wars" year = 1977></Movie>
```

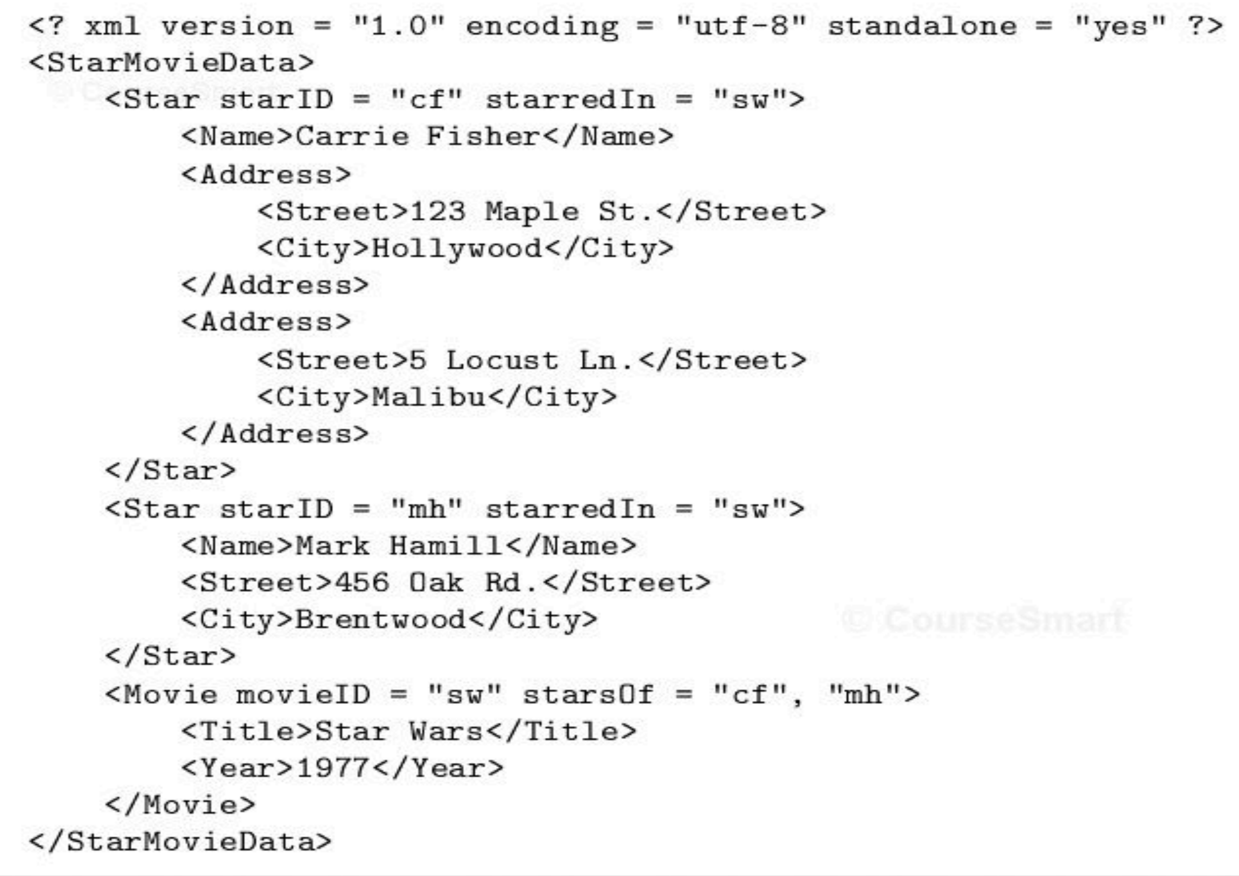
ATTRIBUTES VS. SUB-ELEMENTS



- Elements:
  - Basic building blocks of XML
  - Contain content which can be a structure
- Attributes
  - Specify additional information about an element.
  - Contain only simple type content
- Some data could be either an Element or an Attribute (so you need standards on how to decide which to use).

XML: ID AND IDREF

- In XML document they appear like any other attribute
- ID and IDREF are formally defined in DTD or XML Schema



```
<? xml version = "1.0" encoding = "utf-8" standalone = "yes" ?>
<StarMovieData>
    <Star starID = "cf" starredIn = "sw">
        <Name>Carrie Fisher</Name>
        <Address>
            <Street>123 Maple St.</Street>
            <City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street>
            <City>Malibu</City>
        </Address>
    </Star>
    <Star starID = "mh" starredIn = "sw">
        <Name>Mark Hamill</Name>
        <Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie movieID = "sw" starsOf = "cf", "mh">
        <Title>Star Wars</Title>
        <Year>1977</Year>
    </Movie>
</StarMovieData>
```

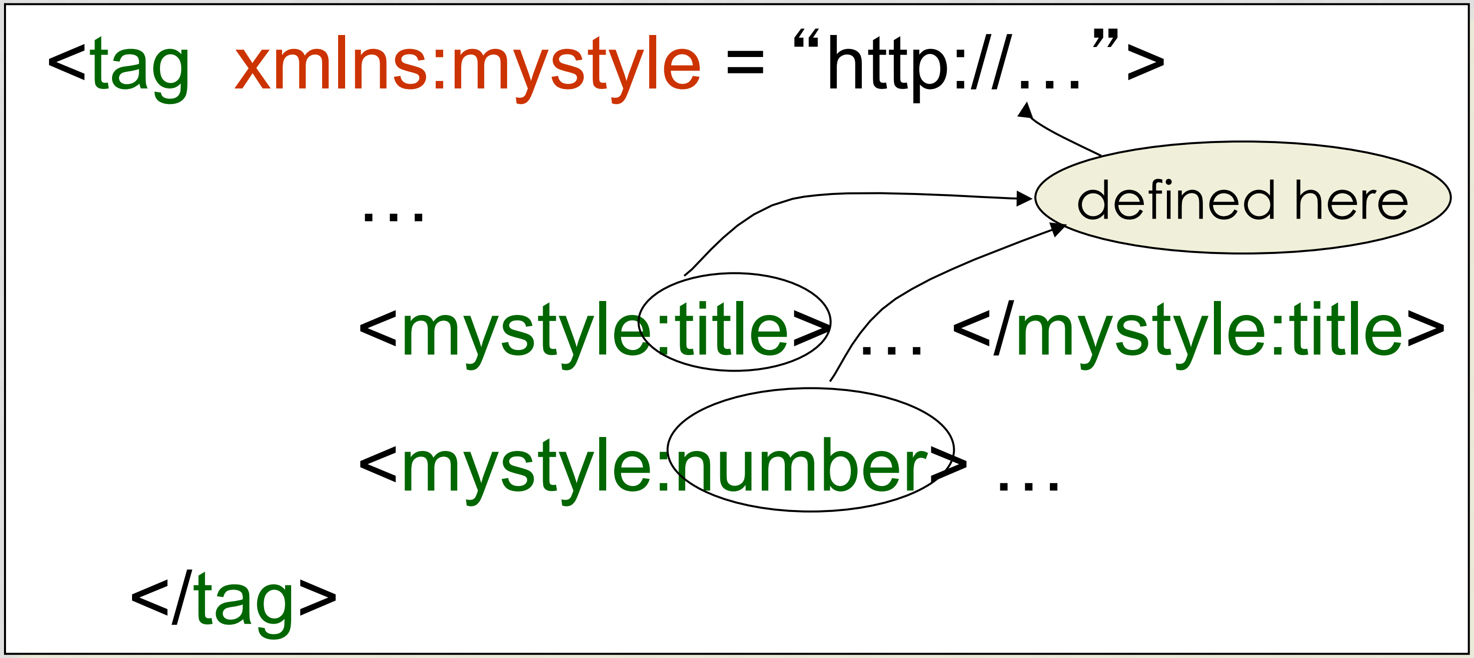
XML NAMESPACES

- Tags may have namespaces
  - They define where the tag is defined (its format or structure)
- Namespace format ➔ ***xmlns:<name>=…***

```
<book xmlns:isbn="www.isbn-org.org/def">
        <title> … </title>
        <number> 15 </number>
        <isbn:number> …. </isbn:number>
</book>
```

XML NAMESPACES

- syntactic: <number> , <isbn:number>
- semantic: provide URL for “shared” schema



```
<tag  xmlns:mystyle = “http://…”>
            …
            <mystyle:title> … </mystyle:title>
            <mystyle:number> …
   </tag>
```

defined here

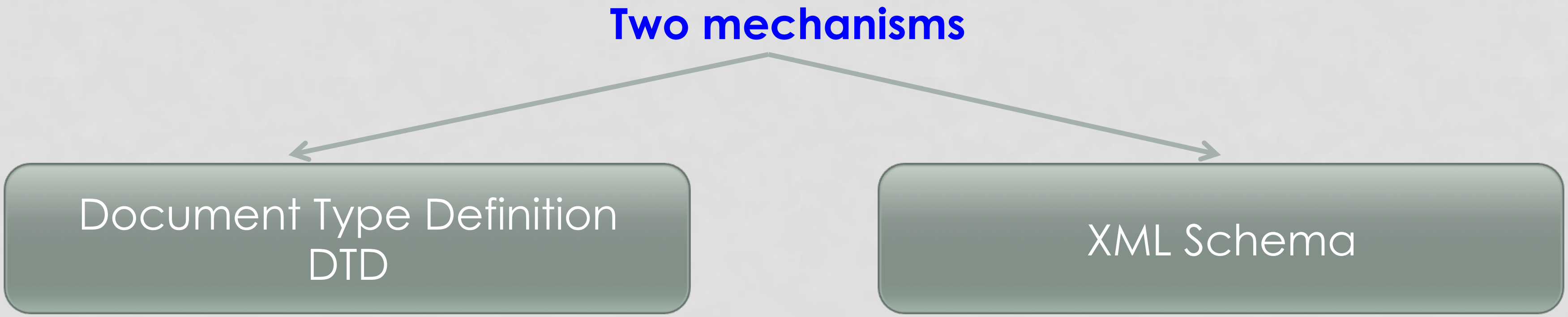
COVERED SO FAR…

- What are XML documents
- XML Structure
  - Tags, start and end tags, elements, attributes
- XML Types
  - Well-formed XML (No schema)
  - Valid XML (has a schema)

XML Schema

XML SCHEMA

- An XML document is usually (but not always) validated by an XML Schema
- The XML Schema provides the information on whether the XML document “followed the rules” set up in the XML Schema
- An XML Schema is an *agreement* between the sender and the receiver of a document as to the structure of that document

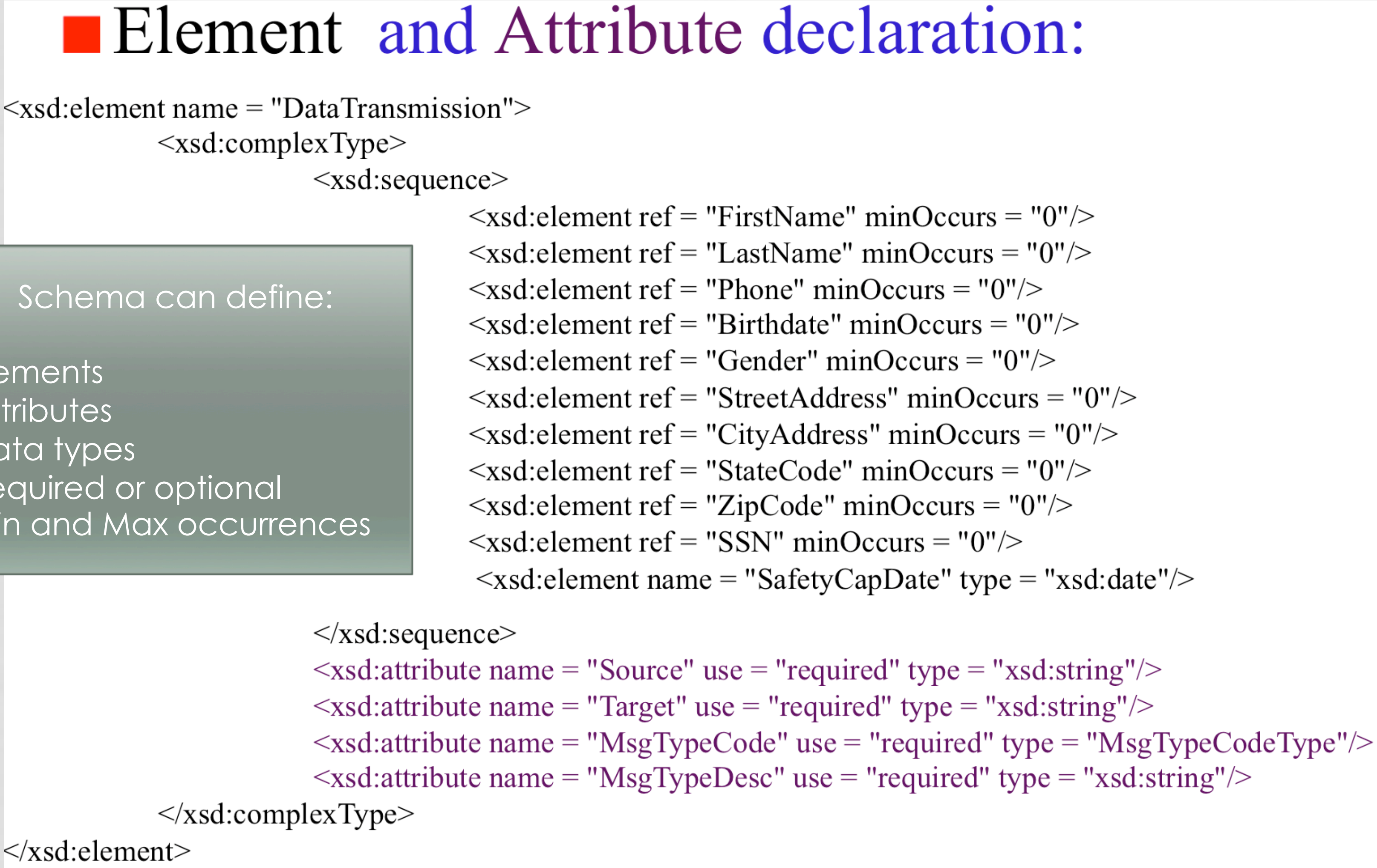


Two mechanisms

Document Type Definition
DTD

XML Schema

XML SCHEMA



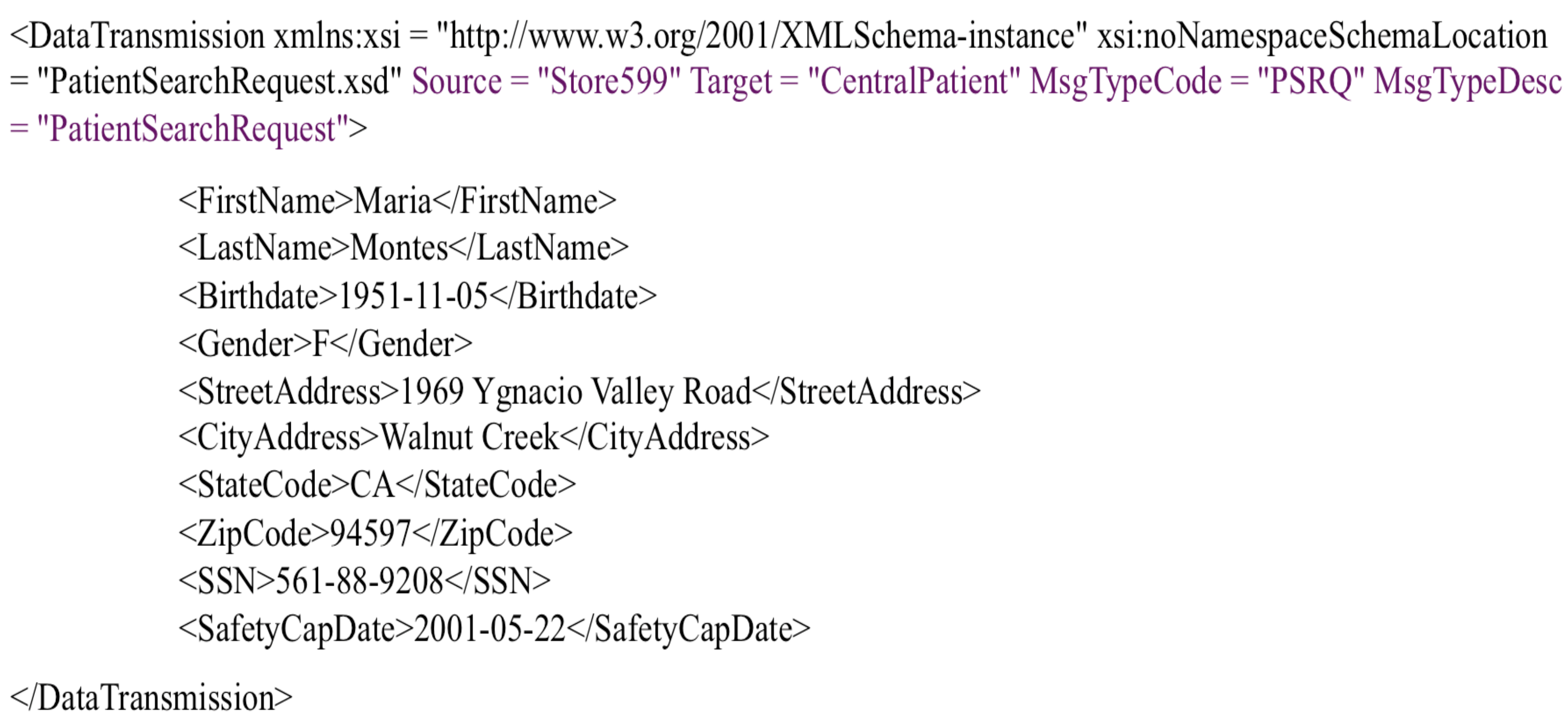
- Element and Attribute declaration:

```
<xsd:element name = "DataTransmission">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element ref = "FirstName" minOccurs = "0"/>
                        <xsd:element ref = "LastName" minOccurs = "0"/>
                        <xsd:element ref = "Phone" minOccurs = "0"/>
                        <xsd:element ref = "Birthdate" minOccurs = "0"/>
                        <xsd:element ref = "Gender" minOccurs = "0"/>
                        <xsd:element ref = "StreetAddress" minOccurs = "0"/>
                        <xsd:element ref = "CityAddress" minOccurs = "0"/>
                        <xsd:element ref = "StateCode" minOccurs = "0"/>
                        <xsd:element ref = "ZipCode" minOccurs = "0"/>
                        <xsd:element ref = "SSN" minOccurs = "0"/>
                         <xsd:element name = "SafetyCapDate" type = "xsd:date"/>

                </xsd:sequence>
                <xsd:attribute name = "Source" use = "required" type = "xsd:string"/>
                <xsd:attribute name = "Target" use = "required" type = "xsd:string"/>
                <xsd:attribute name = "MsgTypeCode" use = "required" type = "MsgTypeCodeType"/>
                <xsd:attribute name = "MsgTypeDesc" use = "required" type = "xsd:string"/>
        </xsd:complexType>
</xsd:element>
```

Schema can define:

-Elements
-Attributes
-Data types
-Required or optional
-Min and Max occurrences

EXAMPLE



```
<DataTransmission xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation
= "PatientSearchRequest.xsd" Source = "Store599" Target = "CentralPatient" MsgTypeCode = "PSRQ" MsgTypeDesc
= "PatientSearchRequest">

            <FirstName>Maria</FirstName>
            <LastName>Montes</LastName>
            <Birthdate>1951-11-05</Birthdate>
            <Gender>F</Gender>
            <StreetAddress>1969 Ygnacio Valley Road</StreetAddress>
            <CityAddress>Walnut Creek</CityAddress>
            <StateCode>CA</StateCode>
            <ZipCode>94597</ZipCode>
            <SSN>561-88-9208</SSN>
            <SafetyCapDate>2001-05-22</SafetyCapDate>

</DataTransmission>
```

Data Types in XML Schema

SIMPLE DATA TYPES IN XML SCHEMA

- Comes with “atomic” simple data types
  - Integer, boolean, date, decimal, string, etc.
- You can build user-defined simple data types
  - Built on the included “atomic” data types
  - Allows declaration of
    - valid values, ranges, Patterns, Length, total digits
    - And more…
  - Attributes or Elements can be of a simple data type (either atomic or user-defined).

EXAMPLE: SIMPLE TYPES

```
<xsd:simpleType name = "SevenPlaceInteger">
        <xsd:restriction base = "xsd:integer">          builds on atomic simple data type
                <xsd:totalDigits value = "7"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name = "GenderType">
        <xsd:restriction base = "xsd:string">
                <xsd:enumeration value = "M"/>
                <xsd:enumeration value = "F"/>
                <xsd:length value = "1"/>
        </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name = "RelationshipCodeType">
        <xsd:restriction base = "xsd:string">
                <xsd:enumeration value = "self"/>
                <xsd:enumeration value = "spouse"/>
                <xsd:enumeration value = "dependent"/>
                <xsd:enumeration value = "other"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name = "SevenPlacePositiveInteger">          builds on custom simple data type
        <xsd:restriction base = "SevenPlaceInteger">
                <xsd:minInclusive value = "0"/>
        </xsd:restriction>
</xsd:simpleType>
```

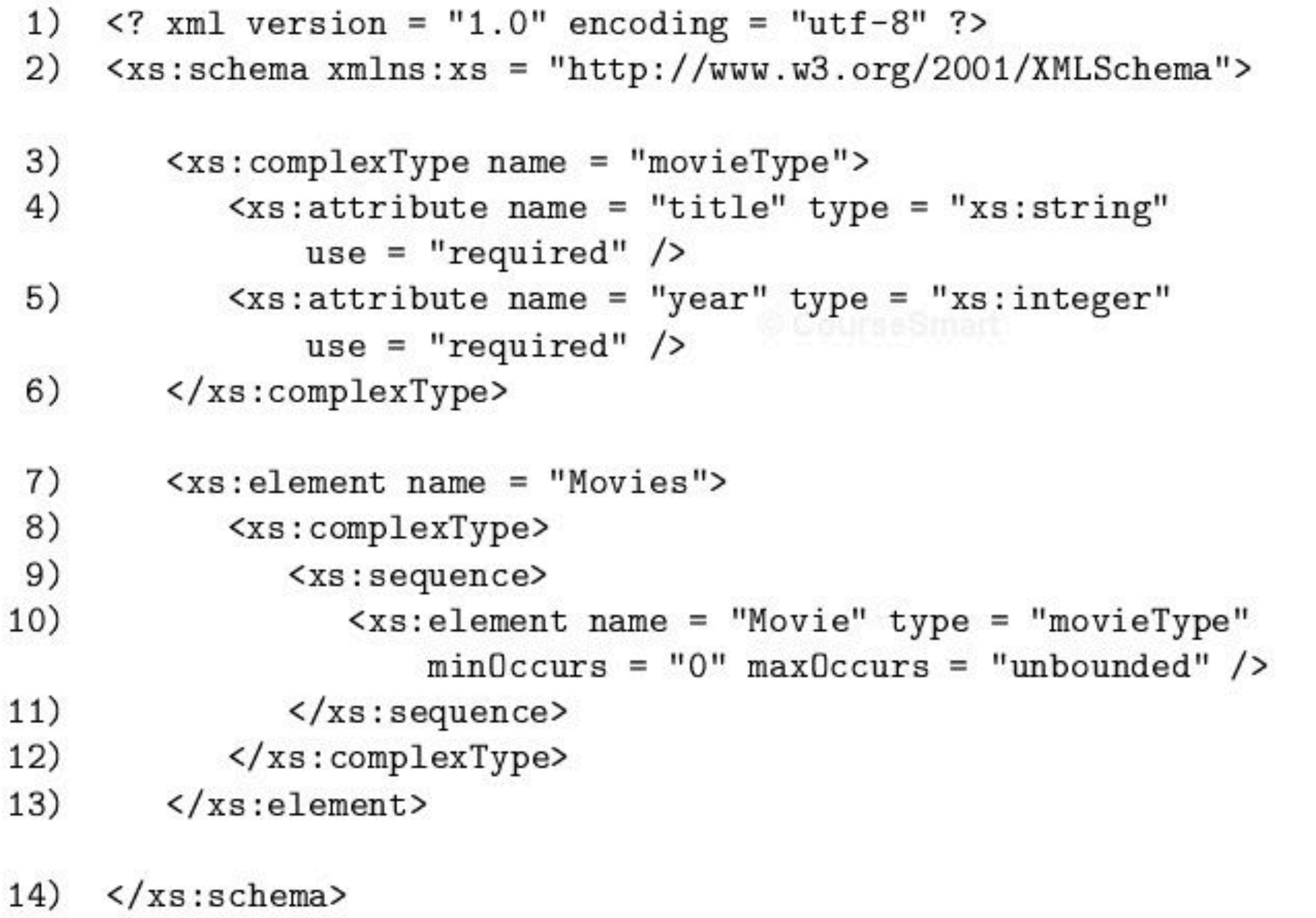
COMPLEX TYPES IN XML SCHEMA

- Builds a structure of Elements.
- Each subelement is either a simple data type or another structure of Elements.
- Only Elements can be of a complex data type.
- Can be named and reusable or anonymous and used only by a single Element.
- Can be an extension or restriction of another complex type.

EXAMPLE: COMPLEX DATA TYPES

```
<xsd:complexType name = "AddressType">      declaration of named complex data type
        <xsd:sequence>
                <xsd:element ref = "StreetAddress"/>
                <xsd:element ref = "CityAddress"/>
                <xsd:element ref = "StateCode"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:element name = "WorkAddress" type = "AddressType"/>  association of Element with named complex data type

<xsd:complexType name = "AddressWithCountryType">  new complex data type extends existing complex data type
        <xsd:complexContent>
                <xsd:extension base = "AddressType">
                        <xsd:sequence>
                                <xsd:element name = "CountryCode" type = "xsd:string"/>
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
<xsd:element name = "PatientInsurance"> element with anonymous complex data type
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element ref = "Patient"/>
                        <xsd:element ref = "TPMembership" minOccurs = "0" maxOccurs = "unbounded"/>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

MOVIES SCHEMA



```
<? xml version = "1.0" encoding = "utf-8" ?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

   <xs:complexType name = "movieType">
      <xs:attribute name = "title" type = "xs:string"
          use = "required" />
      <xs:attribute name = "year" type = "xs:integer"
          use = "required" />
   </xs:complexType>

   <xs:element name = "Movies">
      <xs:complexType>
         <xs:sequence>
            <xs:element name = "Movie" type = "movieType"
                minOccurs = "0" maxOccurs = "unbounded" />
         </xs:sequence>
      </xs:complexType>
   </xs:element>

</xs:schema>
```

TYPE INHERITANCE

```
<complexType name="Address">
 <sequence>  <element name="street" type="string"/>
             <element name="city"   type="string"/>
 </sequence>
</complexType>

<complexType name="USAddress">
 <complexContent>
  <extension base= ”Address">
  <sequence>  <element name="state" type=”string"/>
              <element name="zip"   type="positiveInteger"/>
  </sequence>
 </extension>
 </complexContent>
</complexType>
```

Keys in XML Schema

KEYS IN XML SCHEMA

- Elements in XML can have keys (unique identifiers)
  - Keys can be attributes or subelements
  - A key can be a single field or multiple fields
- Key fields (attributes or subelements) cannot be missing
- Keys are defined in XML schema using special syntax
- Attributes do not have keys

KEYS IN XML SCHEMA

```
<purchaseReport>
        <regions>
         <zip code="95819">
                 <part number="872-AA" quantity="1"/>
                 <part number="926-AA" quantity="1"/>
                 <part number="833-AA" quantity="1"/>
                 <part number="455-BX" quantity="1"/>
         </zip>
         <zip code="63143">
                 <part number="455-BX" quantity="4"/>
         </zip>
        </regions>

        <parts>
                <part number="872-AA">Lawnmower</part>
                <part number="926-AA">Baby Monitor</part>
                <part number="833-AA">Lapis Necklace</part>
                <part number="455-BX">Sturdy Shelves</part>
        </parts>
</purchaseReport>
```

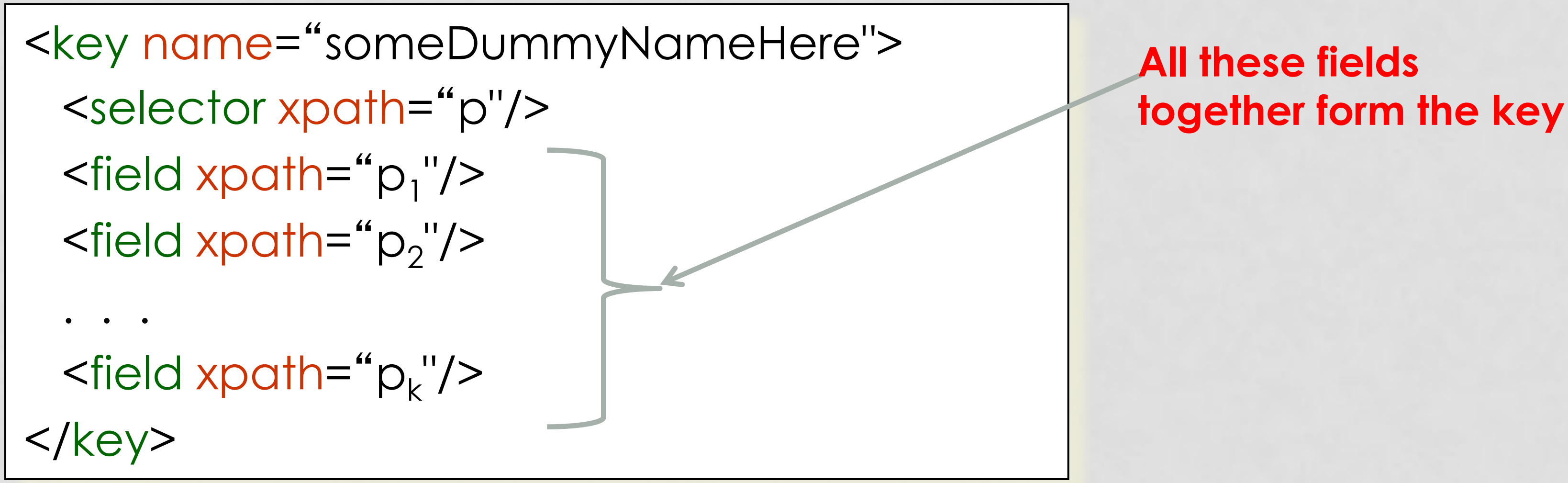
XML Schema for Key :

```
<key name="NumKey">
  <selector xpath="parts/part"/>
  <field xpath="@number"/>
</key>
```

- **Key**: give a name to the key
- **Selector**: following the selector xpath starting from the root, it will return a list of objects
- **Field**: in the returned objects, the xpath defined in ‘field’ has to be unique
  - **@ symbol** refers to attributes

KEYS IN XML SCHEMA

- In general, the key syntax is:

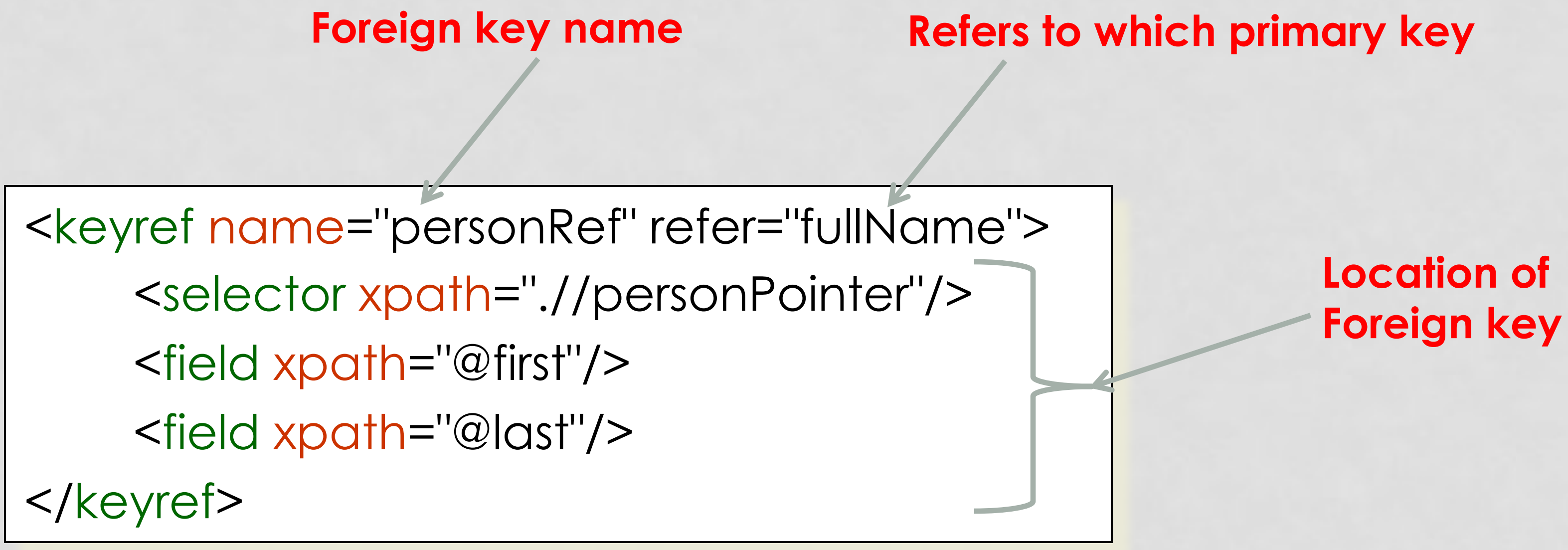


```
<key name="someDummyNameHere">
  <selector xpath="p"/>
  <field xpath="p_1"/>
  <field xpath="p_2"/>
  . . .
  <field xpath="p_k"/>
</key>
```

All these fields together form the key

FOREIGN KEYS IN XML SCHEMA

- Foreign key syntax:



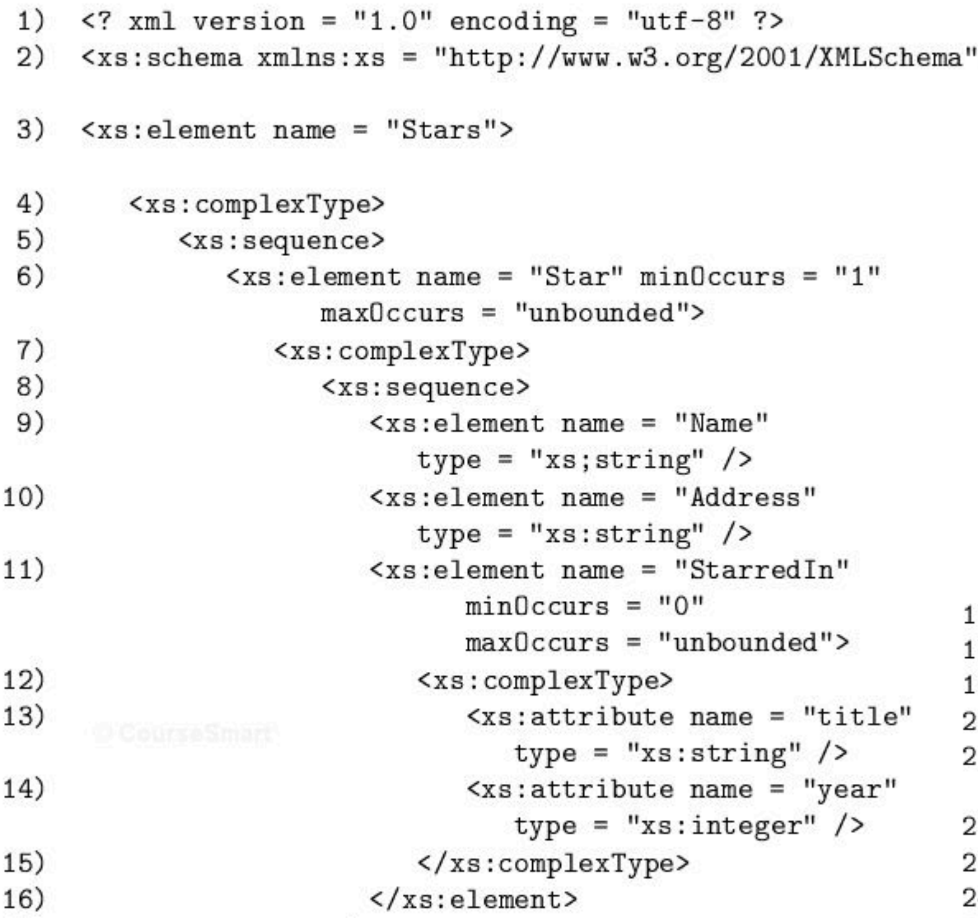
Foreign key name

Refers to which primary key

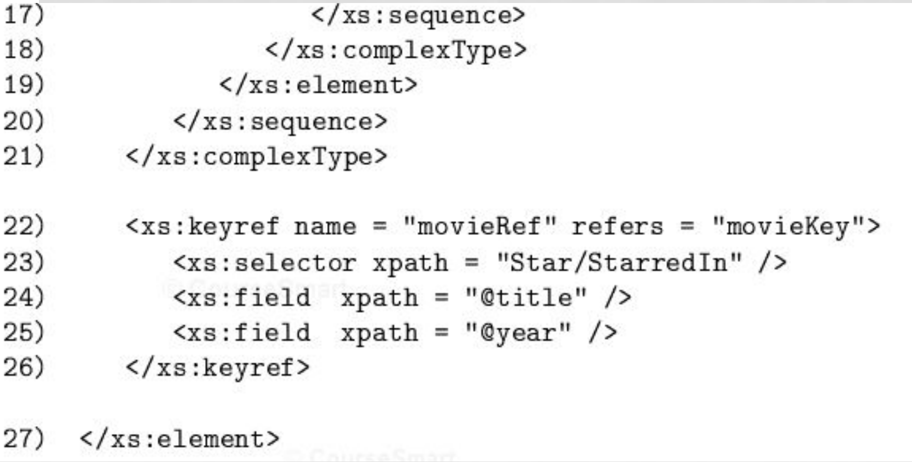
```
<keyref name="personRef" refer="fullName">
        <selector xpath=".//personPointer"/>
        <field xpath="@first"/>
        <field xpath="@last"/>
</keyref>
```

Location of
Foreign key

EXAMPLE: MOVIE SCHEMA

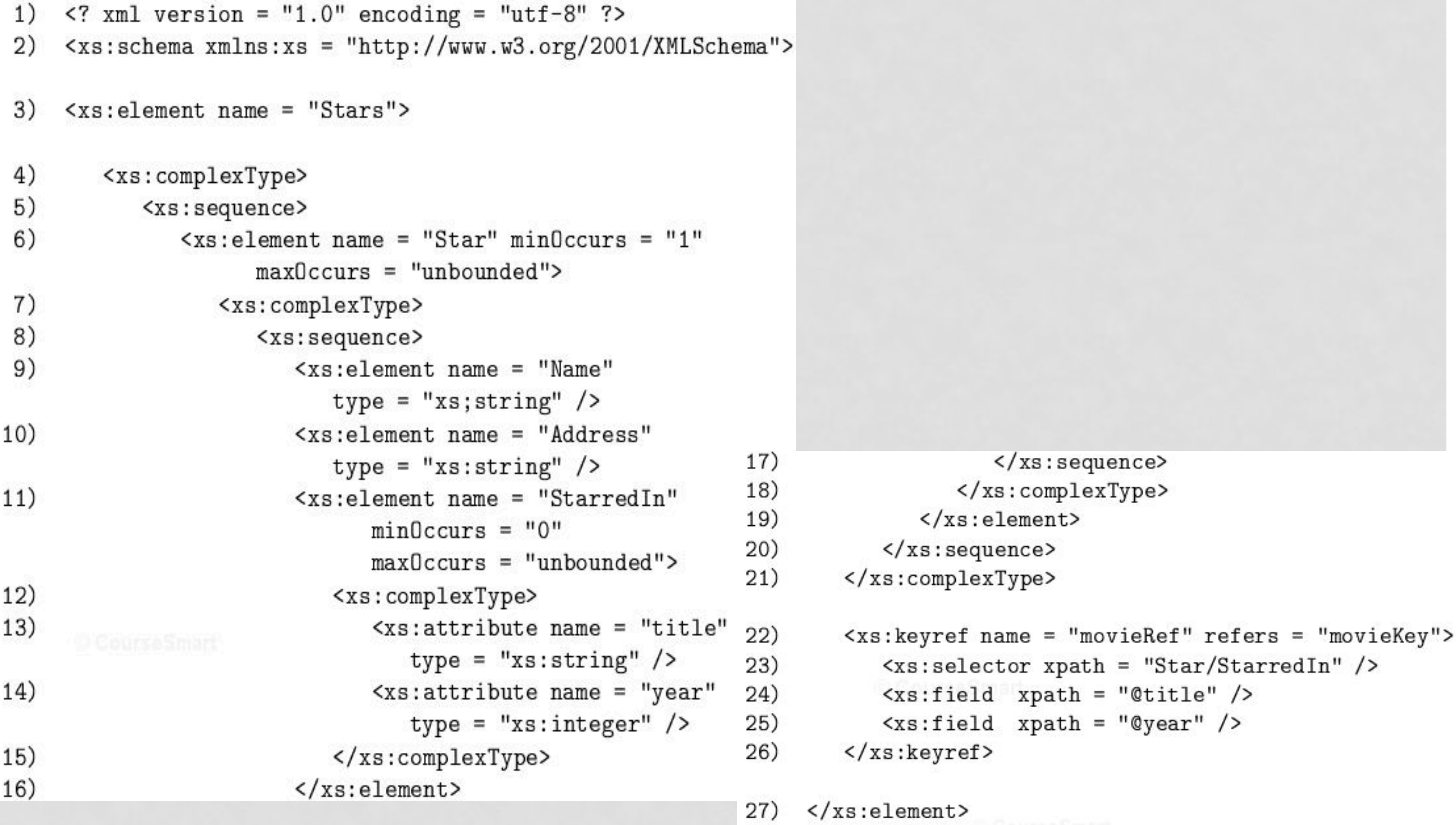


```
<? xml version = "1.0" encoding = "utf-8" ?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

<xs:element name = "Stars">

   <xs:complexType>
      <xs:sequence>
         <xs:element name = "Star" minOccurs = "1"
               maxOccurs = "unbounded">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name = "Name"
                     type = "xs;string" />
                  <xs:element name = "Address"
                     type = "xs:string" />
                  <xs:element name = "StarredIn"
                        minOccurs = "0"
                        maxOccurs = "unbounded">
                     <xs:complexType>
                        <xs:attribute name = "title"
                           type = "xs:string" />
                        <xs:attribute name = "year"
                           type = "xs:integer" />
                     </xs:complexType>
                  </xs:element>
```



```
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>

   <xs:keyref name = "movieRef" refers = "movieKey">
      <xs:selector xpath = "Star/StarredIn" />
      <xs:field  xpath = "@title" />
      <xs:field  xpath = "@year" />
   </xs:keyref>

</xs:element>
```

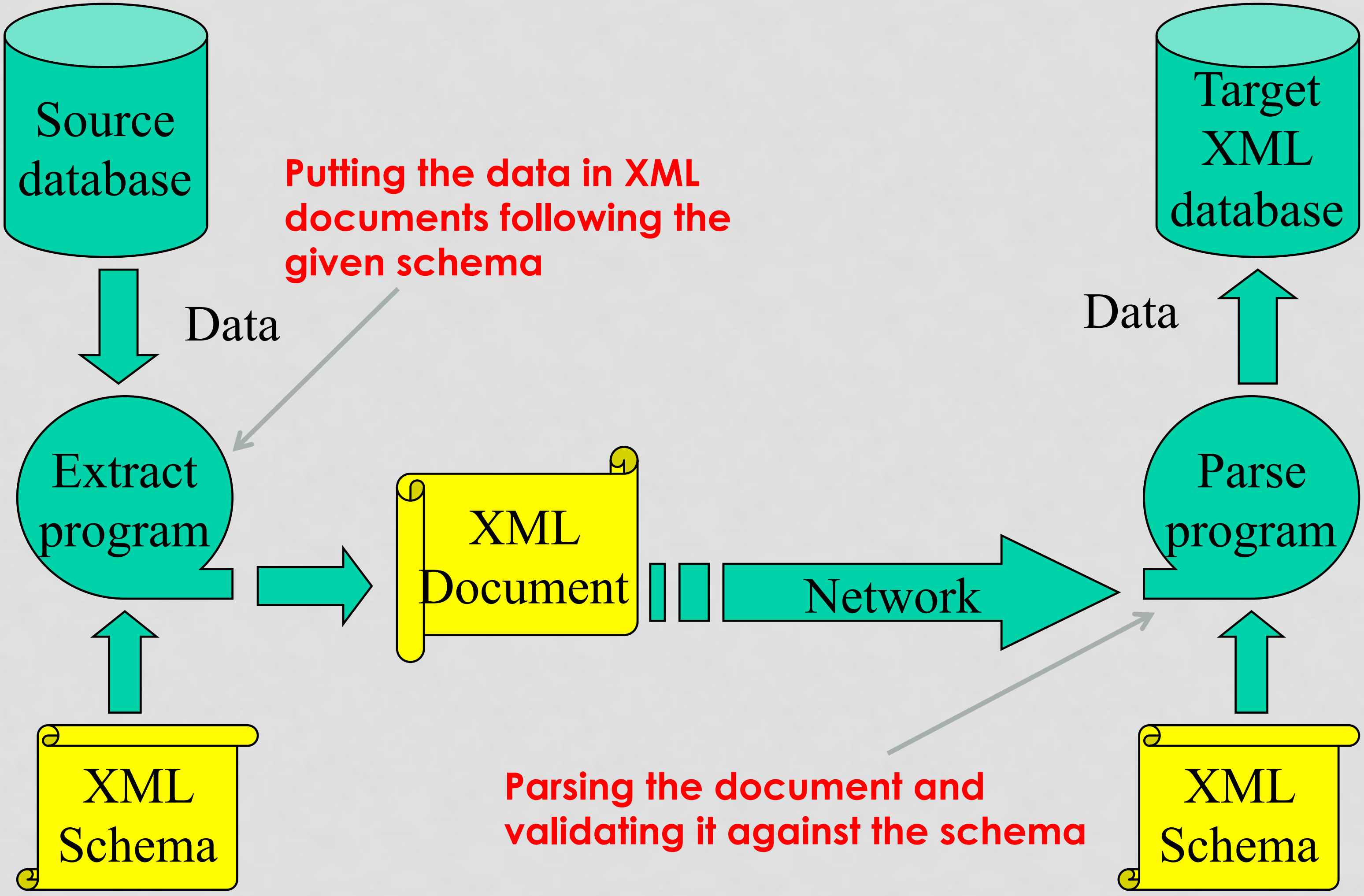
EXAMPLE: STARS SCHEMA



```
<? xml version = "1.0" encoding = "utf-8" ?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">

<xs:element name = "Stars">

   <xs:complexType>
      <xs:sequence>
         <xs:element name = "Star" minOccurs = "1"
               maxOccurs = "unbounded">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name = "Name"
                     type = "xs;string" />
                  <xs:element name = "Address"
                     type = "xs:string" />
                  <xs:element name = "StarredIn"
                        minOccurs = "0"
                        maxOccurs = "unbounded">
                     <xs:complexType>
                        <xs:attribute name = "title"
                           type = "xs:string" />
                        <xs:attribute name = "year"
                           type = "xs:integer" />
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>

   <xs:keyref name = "movieRef" refers = "movieKey">
      <xs:selector xpath = "Star/StarredIn" />
      <xs:field  xpath = "@title" />
      <xs:field  xpath = "@year" />
   </xs:keyref>

</xs:element>
```

Using XML Schema

USING XML SCHEMA

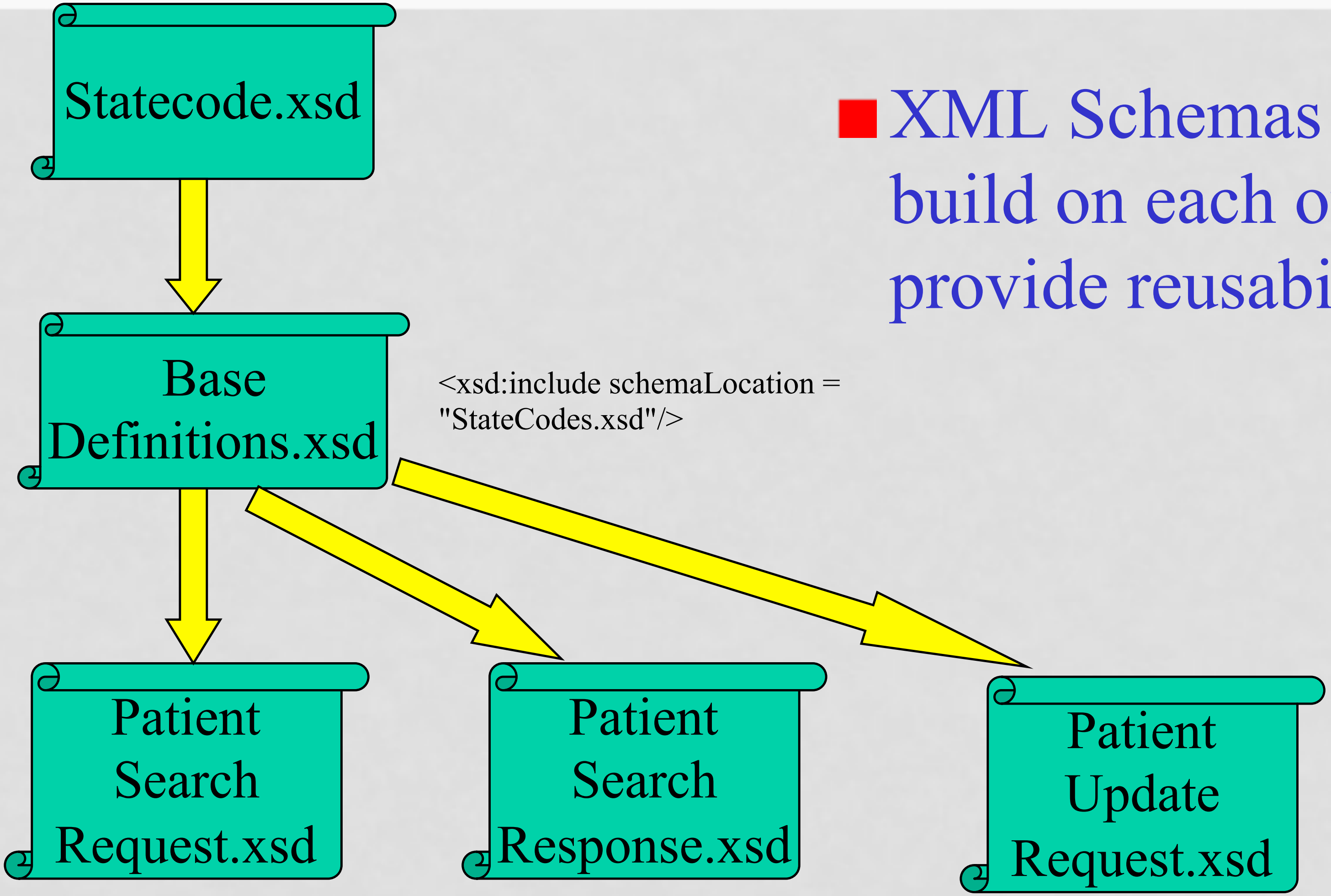


Putting the data in XML documents following the given schema

Parsing the document and validating it against the schema

REUSING XML SCHEMAS

- XML Schemas can build on each other to provide reusability.


Statecode.xsd
Base Definitions.xsd
Patient Search Request.xsd
Patient Search Response.xsd
Patient Update Request.xsd

<xsd:include schemaLocation = "StateCodes.xsd"/>

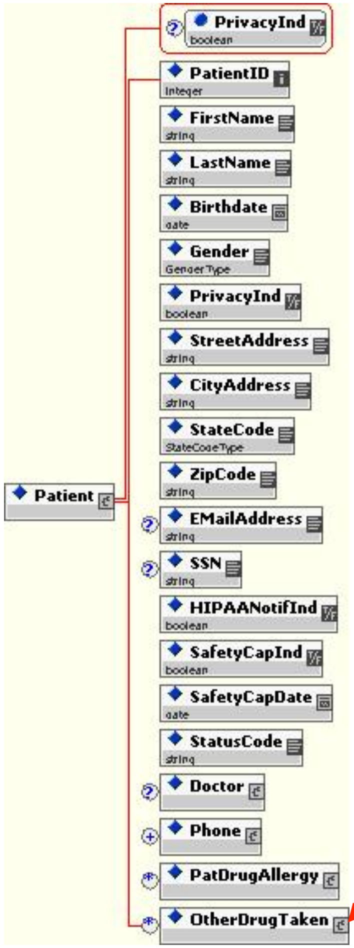
<xsd:include schemaLocation = "BaseDefinitions.xsd"/>

<xsd:include schemaLocation = "BaseDefinitions.xsd"/>

<xsd:include schemaLocation = "BaseDefinitions.xsd"/>

GUI FOR MANAGING XML SCHEMA

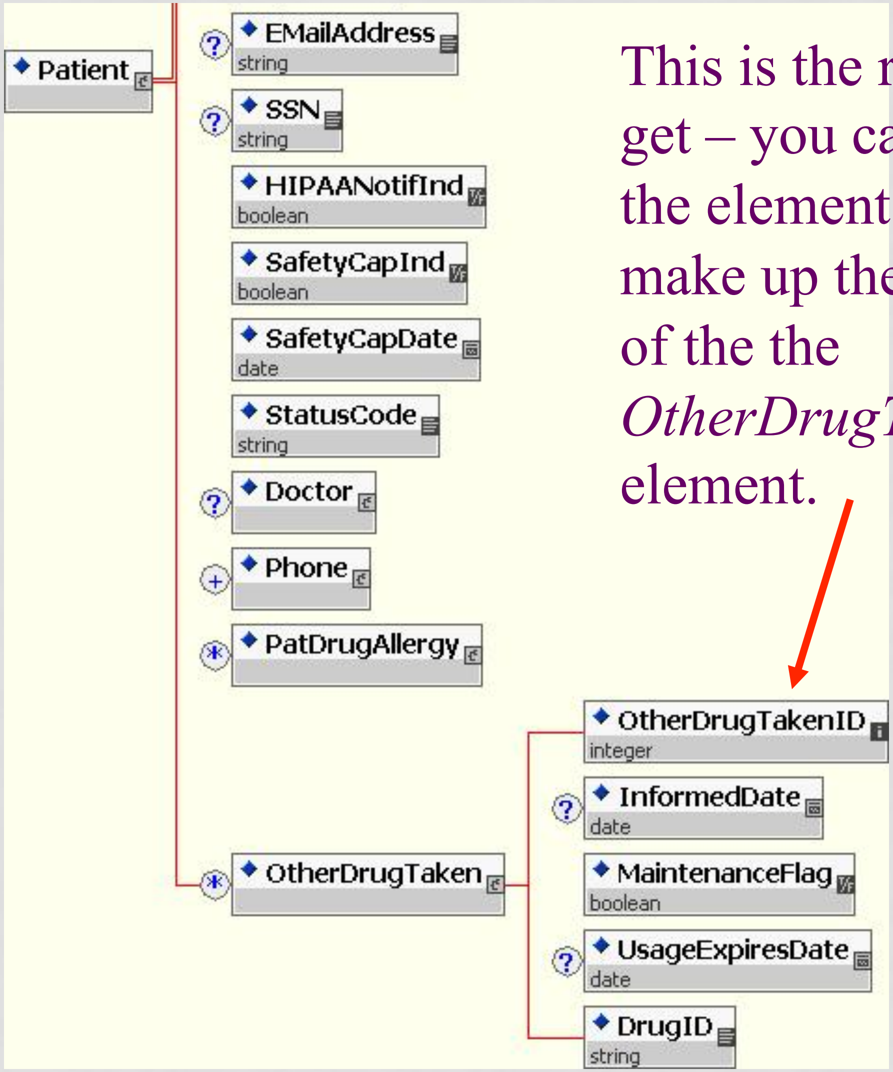
The Structure of an XML Schema



•Elements in an XML Schema are hierarchical.

•To expand the hierarchy with this tool (Tibco’s XML Authority), click here.

EXPANDING ELEMENTS

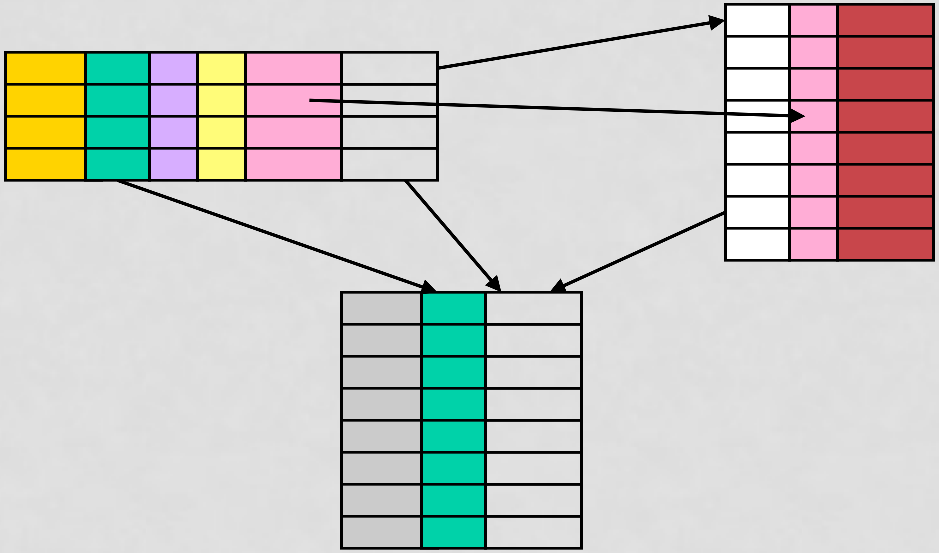


This is the result you get – you can now see the elements that make up the structure of the the *OtherDrugTaken* element.

XML Model
vs.
Relational Model

DATABASE ARCHITECTURE

- Database architecture is relational:
  - Normalized to eliminate data redundancy
  - Join on any two columns that have the same data type.
  - Foreign keys can enforce data integrity



RELATIONAL METADATA – THE SCHEMA

- Relational metadata is stored in the database
  - Database control tables fully define the structure of the database.
  - Without the DBMS metadata the contents of the database are worthless.
  - Completely self-contained (not reusable)
  - Tables are structured, each column is a “bucket” for a specific kind of data
  - In most databases, the metadata does not include descriptions, so a Data Dictionary is necessary.

XML METADATA – THE DOCUMENT

- Metadata built into the document
  - Every element has a tag to tell you where the data is stored in the document.
  - Descriptive tags give structure to the document and tell you what the data means (sort of).
  - Document cannot be parsed for storage on its own. What else is needed?…

XML METADATA – THE SCHEMA

- An XML Schema (or DTD) is needed to:
  - Provide standardization (basis of agreement)
  - Allow meaningful parsing and data storage
  - Specify agreement on document structure
- A data dictionary is still necessary to provide definition for Elements and Attributes

COMPARISON

RDBMS	XML
Relationships among items is explicitly defined	Relationships among items inferred by position
General-purpose storage and processing systems	Used for data exchange and with XSLT for web visualization
Good for general-purpose queries asking for different objects	Good for partitioned data and for retrieving objects with their all sub-components
Easy to optimize for storage and querying	Harder to optimize for storage and querying
Straightforward to export to XML	Usually not straightforward