

# XML

(Very brief introduction)

# Learning Outcomes

- Understand and be able to recognise XML syntax
- Understand and be able to explain what is “well-formed”/”valid” XML
- Understand and be able to explain the purpose of DTD
- Understand and be able to explain the purpose of XML schema (XSD)
- Be able to explain the differences between relational model and XML

# eXtensible Markup Language (XML)

- A meta-language (i.e. a language for describing other languages) that enables designers to create their own customised tags to provide functionality not available with HTML.
- Standard for data representation and exchange
- Also streaming format
- XML has a document format similar to HTML
  - Tags describe content instead of formatting



# HTML

```
<html>
  <head>
    <title>My Title</title>
  </head>
  <body>
    <h1>Heading 1</h1>
    <p>Hello World</p>
  </body>
</html>
```

## Heading 1

Hello World

```
<h1>Bibliography</h1>
<p><i>Foundations of Databases</i>,
Abiteboul, Hull, and Vianu
<br>Addison Wesley, 1995
<p>...
```

 Mozilla Firefox

## Bibliography

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

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

# XML

```
<!--Bookstore with no DTD -->
<Bookstore>
  <Book ISBN="0590353403" Price="20" Edition="1st"
    <Title>Harry Potter and the Philosopher's Stone</Title>
    <Authors>
      <Author>
        <First_Name>J.k.</First_Name>
        <Last_Name>Rowling</Last_Name>
      </Author>
    </Authors>
  </Book>
</Bookstore>
```

# XML - Elements

```
<!--Bookstore with no DTD -->
<Bookstore>
  <Book ISBN="0590353403" Price="20" Edition="1st"
    <Title>Harry Potter and the Philosopher's Stone</Title>
    <Authors>
      <Author>
        <First_Name>J.k.</First_Name>
        <Last_Name>Rowling</Last_Name>
      </Author>
    </Authors>
  </Book>
</Bookstore>
```

- Elements, or tags, are most common form of markup.
- First element must be a root element, which can contain other (sub)elements.
- XML document must have one root element **<Bookstore>**.
  - start-tag **<Bookstore>**
  - end-tag **</Bookstore>**
- XML elements are case sensitive.
- An element can be empty, in which case it can be abbreviated to **<EMPTYELEMENT/>**
- Elements must be properly nested.

# XML - Attributes

```
<!--Bookstore with no DTD -->
<Bookstore>
  <Book ISBN="0590353403" Price="20" Edition="1st"
    <Title>Harry Potter and the Philosopher's Stone</Title>
    <Authors>
      <Author>
        <First_Name>J.k.</First_Name>
        <Last_Name>Rowling</Last_Name>
      </Author>
    </Authors>
  </Book>
</Bookstore>
```

- Attributes are name-value pairs that contain descriptive information about an element.
- Attribute is placed inside start-tag after corresponding element name with the attribute value enclosed in double quotes.

```
<Book ISBN="0590353403" Price="20" Edition="1st" >
```

- A given attribute may only occur once within a tag, while sub-elements with same tag may be repeated.

# XML – Other Sections

```
<!--Bookstore with no DTD -->
<Bookstore>
  <Book ISBN="0590353403" Price="20" Edition="1st"
    <Title>Harry Potter and the Philosopher's Stone</Title>
    <Authors>
      <Author>
        <First_Name>J.k.</First_Name>
        <Last_Name>Rowling</Last_Name>
      </Author>
    </Authors>
  </Book>
</Bookstore>
```

- *XML declaration*: optional at start of XML document.

E.g., `<?xml version="1.0" encoding="UTF-8" standalone="no" ?>`

- *Entity references*: serve various purposes, such as shortcuts to often repeated text or to distinguish reserved characters from content.

E.g., `<` is the same as `&lt;`

- *Comments*: enclosed in `<!--` and `-->` tags.

E.g., `<!--this is a comment -->`

- XML – Ordering
  - elements are ordered.
  - attributes are unordered.



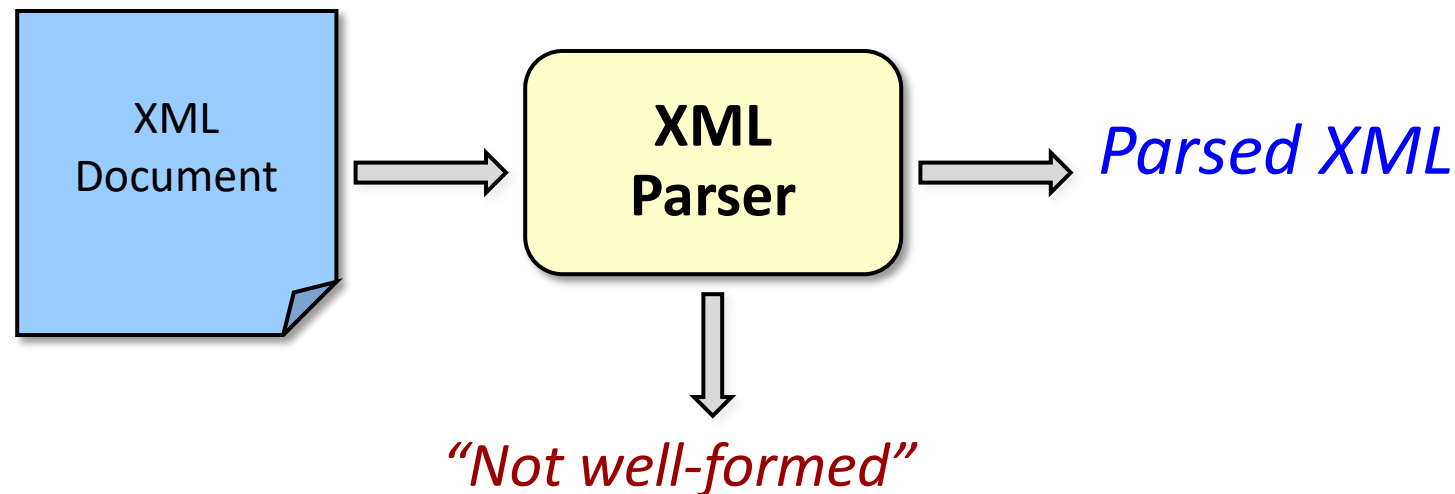
- **C, C++ , FORTRAN, Pascal, Visual Basic, Java...**
  - are *general-purpose programming languages*
  - You can: specify calculations, actions, and decisions to be carried out in order.
- **SQL**
  - is a *special-purpose programming language*
  - You can: manage data in a relational databases.
- **XML**
  - is a *markup specification language*
  - You can: design ways of describing information (text or data), usually for storage, transmission, or processing by a program (you can use it in combination with a programming language).
  - It says nothing about what you should do with the data (although your choice of element names may hint at what they are for).

# What does XML do??

- XML doesn't *do* anything!
- It's a data format which just sits there until you run a program (e.g., Java, C, ...) which does something *with* it.

# “Well-Formed” XML

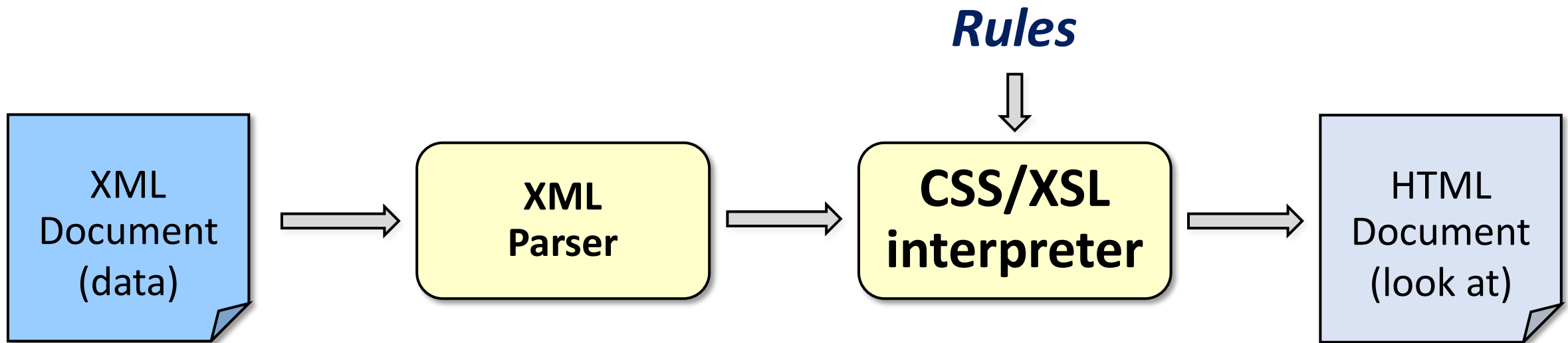
- Adheres to basic structural requirements
  - Single root element
  - Matched tags, proper nesting
  - Unique attributes within elements



# Displaying XML

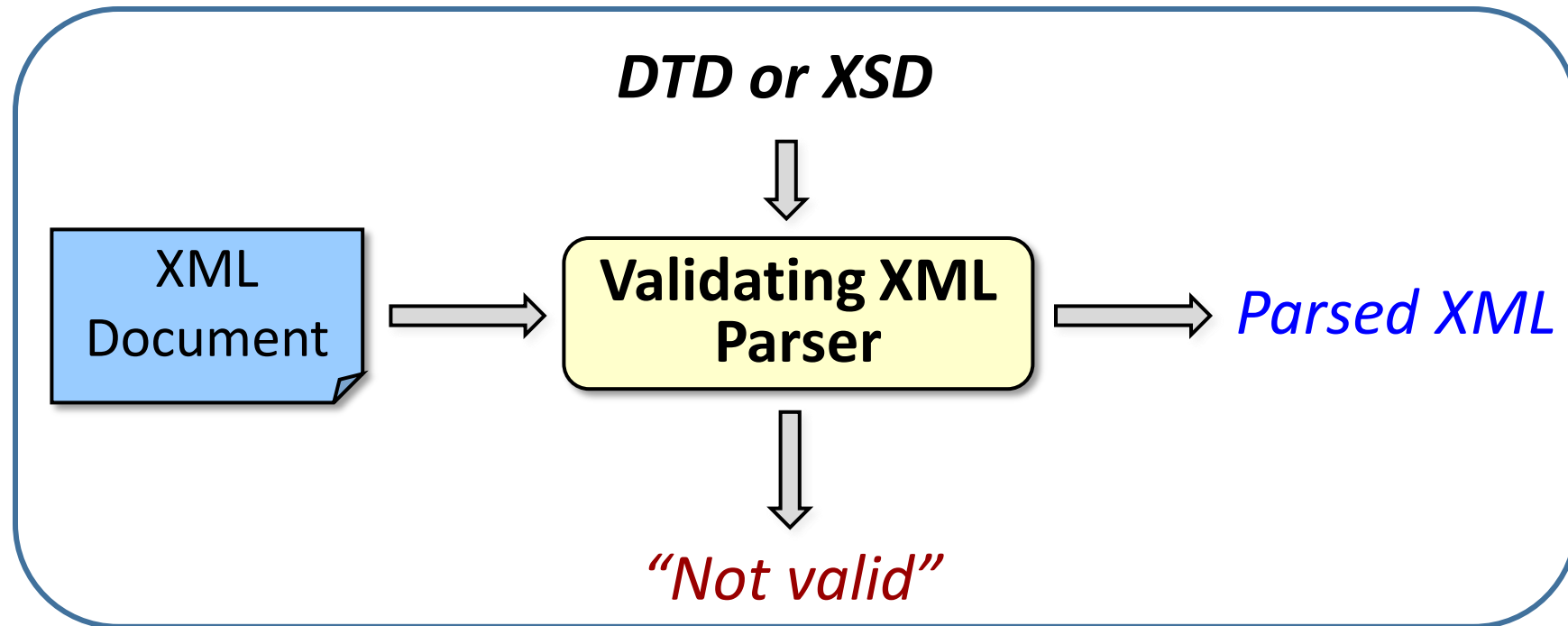
Use rule-based languages to translate to HTML

- *Cascading StyleSheets* (CSS): alternative rendering to the tags
- *eXtensible Stylesheet Language* (XSL): defines how data is rendered



# Document Type Definitions (DTDs) “Valid” XML

- **Defines the valid syntax of an XML document.**
- Specifies elements, attributes, nesting, ordering, number of occurrences, also special attribute types ID and IDREF(S).
- Term *vocabulary* sometimes used to refer to the elements used in a particular application.
- Grammar is specified using EBNF, not XML.
- Although optional, DTD is recommended for document conformity.



# Document Type Definitions (DTDs) – example

```
<!DOCTYPE Bookstore [  
  <!ELEMENT Bookstore (Book*, Author*)>  
  <!ELEMENT Book (Title, Remark?)>  
  <!ATTLIST Book ISBN ID #REQUIRED  
    Price CDATA #REQUIRED  
    Authors IDREFS #REQUIRED>  
  <!ELEMENT Title (#PCDATA)>  
  <!ELEMENT Remark (#PCDATA | BookRef)*>  
  <!ELEMENT BookRef EMPTY>  
  <!ATTLIST BookRef book IDREF #REQUIRED>  
  <!ELEMENT Author (First_Name, Last_Name)>  
  <!ATTLIST Author Ident ID #REQUIRED>  
  <!ELEMENT First_Name (#PCDATA)>  
  <!ELEMENT Last_Name (#PCDATA)>  

```

```
<Bookstore>
  <Book ISBN="0590353403" Price="20" Edition="1st" Authors="JR">
    <Title>Harry Potter and the Philosopher's Stone</Title>
    <Authors>
      <Author>
        <First_Name>J.k.</First_Name>
        <Last_Name>Rowling</Last_Name>
      </Author>
    </Authors>
  </Book>
  <Book ISBN="9781292061184" Price="35" Edition="6th" Authors="TC CB">
    <Title>Database Systems</Title>
    <Remark>Buy this book with
      <BookRef book="0590353403"/>
      - A great deal!
    </Remark>
    <Authors>
      <Author ID="TC">
        <First_Name>Thomas</First_Name>
        <Last_Name>Connolly</Last_Name>
      </Author>
      <Author ID="CB">
        <First_Name>Carolyn</First_Name>
        <Last_Name>Begg</Last_Name>
      </Author>
    </Authors>
  </Book>
</Bookstore>
```

# DTDs: Element Type Declarations

- Identify the rules for elements that can occur in the XML document. Options for repetition are:
  - \* zero or more occurrences for an element;
  - + one or more occurrences for an element;
  - ? zero occurrences or (exactly) one occurrence for an element.
- Name with no qualifying punctuation must occur exactly once.
- Commas between element names indicate they must occur in succession; if commas are omitted, elements can occur in any order.
- #PCDATA: *parsable character data*, declares the base elements.

```
<!DOCTYPE Bookstore [  
  <!ELEMENT Bookstore (Book*, Author*)>  
  <!ELEMENT Book (Title, Remark?)>  
  <!ATTLIST Book ISBN ID #REQUIRED  
    Price CDATA #REQUIRED  
    Authors IDREFS #REQUIRED>  
  <!ELEMENT Title (#PCDATA)>  
  <!ELEMENT Remark (#PCDATA | BookRef)*>  
  <!ELEMENT BookRef EMPTY>  
  <!ATTLIST BookRef book IDREF #REQUIRED>  
  <!ELEMENT Author (First_Name, Last_Name)>  
  <!ATTLIST Author Ident ID #REQUIRED>  
  <!ELEMENT First_Name (#PCDATA)>  
  <!ELEMENT Last_Name (#PCDATA)>  
>
```



# DTDs: **Attribute** List Declarations

- Identify which elements may have attributes, what attributes they may have, what values attributes may hold, plus optional defaults.
- Some types:
  - CDATA: character data, containing any text.
  - whether an attribute must occur (#REQUIRED) or not (#IMPLIED).
  - ID: used to identify individual elements in document (ID is an element name).
  - IDREF/IDREFS: must correspond to value of ID attribute(s) for some element in document.
  - List of names: values that attribute can hold.

```
<!DOCTYPE Bookstore [  
  <!ELEMENT Bookstore (Book*, Author*)>  
  <!ELEMENT Book (Title, Remark?)>  
  <!ATTLIST Book ISBN ID #REQUIRED  
    Price CDATA #REQUIRED  
    Authors IDREFS #REQUIRED>  
  <!ELEMENT Title (#PCDATA)>  
  <!ELEMENT Remark (#PCDATA | BookRef)*>  
  <!ELEMENT BookRef EMPTY>  
  <!ATTLIST BookRef book IDREF #REQUIRED>  
  <!ELEMENT Author (First_Name, Last_Name)>  
  <!ATTLIST Author Ident ID #REQUIRED>  
  <!ELEMENT First_Name (#PCDATA)>  
  <!ELEMENT Last_Name (#PCDATA)>  
>
```

# DTDs: Element Identity, IDs, IDREFs

- ID allows unique key to be associated with an element.
- IDREF allows an element to refer to another element with the designated key.
- Attribute type IDREFS allows an element to refer to multiple elements.
- Example – to loosely model relationship ***Branch Has Staff***:
  - <!ATTLIST STAFF staffNo ID #REQUIRED>
  - <!ATTLIST BRANCH staff IDREFS #IMPLIED>

# XML Schema Definition (XSD) – 1/2

- DTDs have number of limitations:
  - it is written in a different (non-XML) syntax;
  - it has no support for namespaces;
  - it only offers extremely limited data typing.
  - **OLD!**
- XML Schema is a more comprehensive method of defining content model of an XML document.
- Tries to overcome XML deficiencies.
- Additional expressiveness allows web applications to exchange XML data more robustly without relying on ad hoc validation tools.

# XML Schema Definition (XSD) – 2/2

- XML schema is the definition (both in terms of its organization and its data types) of a specific XML structure.
- Like DTDs, can specify elements, attributes, nesting, ordering, occurrences.
- Also data types, keys, (typed) pointers, and more ...
- Schema is an **XML** document, and so can be edited and processed by same tools that read the XML it describes.
- XSD aspects (and syntax)
  - Type
  - Key declaration
  - References
  - Occurrence constraints

# XSD Types

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Example of type definition:

XSD:

```
<xsd:attribute name="Price" type="xsd:integer" use="required"/>
```

XML:

```
<Book ISBN="0590353403" Price="100">
```

If `Price="100"` is changed to `Price="foo"`, it would not validate

# XSD Key declaration <xsd:key>

- Specifies that an attribute/element value (or set of values) must be a key within the containing element in an instance document.
- A key means that data should be unique within a specified scope, non-nillable, and always present.

## XSD:

```
<xsd:key name="BookKey">
  <xsd:selector xpath="Book"/>
  <xsd:field xpath="@ISBN"/>
</xsd:key>
<xsd:key name="AuthorKey">
  <xsd:selector xpath="Author"/>
  <xsd:field xpath="@Ident"/>
</xsd:key>
```

## XML:

```
<Author Ident="HG">
  <First_Name>Hector</First_Name>
  <Last_Name>Garcia-Molina</Last_Name>
</Author>
<Author Ident="TC">
  <First_Name>Thomas</First_Name>
  <Last_Name>Connolly</Last_Name>
</Author>
<Author Ident="CB">
  <First_Name>Carolyn</First_Name>
  <Last_Name>Begg</Last_Name>
</Author>
```

```
<Book ISBN="9781292061184" Price="35">
  <Title>Database Systems</Title>
  <Authors>
    <Auth authIdent="TC"/>
    <Auth authIdent="CB"/>
  </Authors>
</Book>
```

# XSD References <xsd:keyref>

- Specifies that an attribute or element value (or set of values) corresponds to those of the specified **key** or **unique** element.

## XSD:

```
<xsd:keyref name="AuthorKeyRef" refer="AuthorKey">
  <xsd:selector xpath="Book/Authors/Auth"/>
  <xsd:field xpath="@authIdent"/>
</xsd:keyref>
<xsd:keyref name="BookKeyRef" refer="BookKey">
  <xsd:selector xpath="Book/Remark/BookRef"/>
  <xsd:field xpath="@book"/>
</xsd:keyref>
```

## XML:

```
<Book ISBN="9781292061184" Price="35">
  <Title>Database Systems</Title>
  <Authors>
    <Auth authIdent="TC"/>
    <Auth authIdent="CB"/>
  </Authors>
  <Remark>
    Buy this book with
    <BookRef book="0590353403"/>
    - a great deal!
  </Remark>
</Book>
```

# XML Schema – Simple and Complex Types



- Elements that do not contain other elements or attributes are known as being of type **simpleType**.

```
<xsd:element name="STAFFNO" type = "xsd:string"/>
<xsd:element name="DOB" type = "xsd:date"/>
<xsd:element name="SALARY" type = "xsd:decimal"/>
```

- Attributes must be defined last:

```
<xsd:attribute name="branchNo" type = "xsd:string"/>
```

- Elements that contain other elements are known as being of type **complexType**.
- List of children of **complexType** are described by the **sequence** element.

```
<xsd:element name = "STAFFLIST">
  <xsd:complexType>
    <xsd:sequence>
      <!-- children defined here -->
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



# XSD Cardinality and Occurrences

- Cardinality of an element can be represented using attributes **minOccurs** and **maxOccurs**.
- To represent an optional element, set **minOccurs** to **0**; to indicate there is no maximum number of occurrences, set **maxOccurs** to **unbounded**.

```
<xsd:element name="DOB" type="xsd:date" minOccurs = "0"/>
```

```
<xsd:element name="NOK" type="xsd:string" minOccurs = "0" maxOccurs = "3"/>
```

- Another example:

```
<xsd:element name="Book" type="BookType" minOccurs="0" maxOccurs="unbounded"/>
```

```
<xsd:element name="Author" type="AuthorType" minOccurs="0" maxOccurs="unbounded"/>
```

```
<xsd:element name="Auth" maxOccurs="unbounded">
```

```
<xsd:element name="Remark" minOccurs="0">
```

# Querying XML

- Not nearly as mature as Querying a Relational Model
  - Newer
  - No underlying algebra
- Steps required to query XML (**out of scope in this course**)
  1. XPath (path expressions + conditions)
  2. XSLT (XPath + transformations, output formatting)
  3. XQuery (XPath + full-featured query language)

# Relational Model versus XML



	<b>Relational</b>	<b>XML (Semi-Structured data)</b>
Structure	Tables, columns, rows	Hierarchical, tree
Schema	Fixed in advance	Self-describing, flexible
Queries	SQL, simple language, standard	Not so simple
Ordering	None	Ordered
Implementation	Mature, native	Add-on

# What have we learned?

- What is XML, its purpose and XML basic syntax
- What is “well-formed” XML
- Purpose of DTD
- Purpose of XML schema (XSD) and advantages over DTD
- Basic differences between relational model and XML