Introduction to XML Schemas

Tutorial

XML Europe 2001

21.5.2001, Berlin

Overview

- 1 Introduction
  - Why are Schemas?
- 2 Concepts
  - What are schemas?
- 3 Schema Languages
  - History and Status Quo
- 4 Conclusion
  - Prospects, Examples, Discussion

1 Introduction

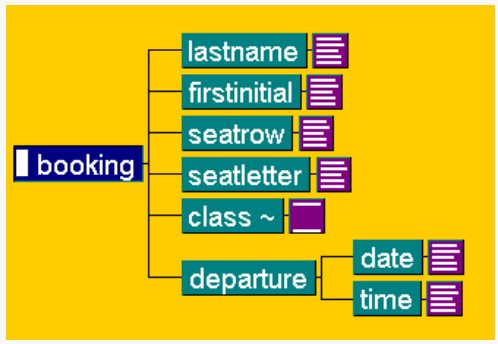
Why are schemas?

INFOTAKT information design

Introduction
Concepts
Languages
Conclusion

Document Type Definition (DTD)

- defines document structures: a tree with nodes and leaves



Database Schema

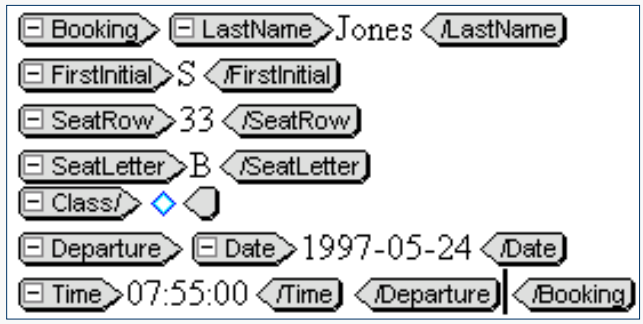
- defines data structures
  - tuples (attribute and domain)
    - customer_id Integer
    - name String
    - zip-code Decimal
- defines a logical structure
  - attributes
  - relations between them
  - constraints

An Example

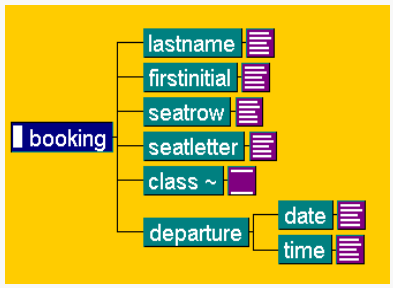
- Online Booking:

Passenger	Jones, S.
Seat	33 B
Departure	1997-05-24T07:55:00
Class	1

XML-ized



The architecture: DTD



„Schemas define the characteristics of classes of objects."

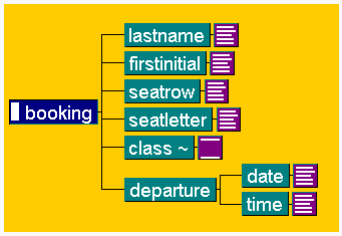
→ A DTD is a schema!

What can you do with DTDs?

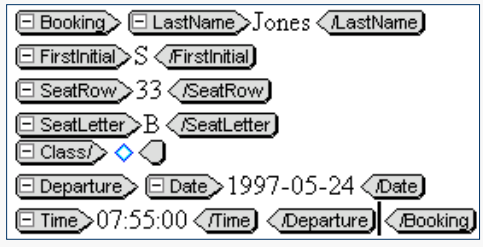
- Descriptive Markup
- An online booking consists of
  - the last name <LastName>
  - the first initial <FirstInitial>
  - the seat <SeatRow>, <SeatLetter>
  - the class <Class>
  - the departure time <Departure>
  - in this order

Components of XML

Schema



XML Document Instance



Requirements

- Example: Online Booking
- An exact description:
  - <FirstInitial> is exactly one letter
  - <SeatRow> is an integer value between 1 and 60
  - <SeatLetter> is A, B, C, or D
  - <Date> is a date
  - <Time> is a time value

Conclusion

- DTDs are limited
  There are new requirements:
  - Dynamic documents => dynamic schemas
  - Not only documents => data modelling
  - more control - more flexibility
- → XML Schemas

2 Concepts

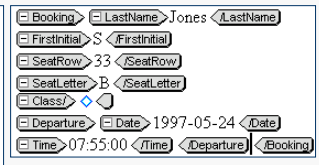
What are XML schemas?
Goals

Classical Components

DTD

```
<!ELEMENT Booking  (LastName, FirstInitial, SeatRow,
                    SeatLetter, Class,Departure)       >
<!ELEMENT LastName  (#PCDATA) >
<!ELEMENT FirstInitial  (#PCDATA) >
<!ELEMENT SeatRow  (#PCDATA) >
<!ELEMENT SeatLetter  (#PCDATA) >
<!ELEMENT Class   EMPTY  >
<!ATTLIST Class
          level (Economy | Business | First)  "Economy" >
<!ELEMENT Departure  (Date,Time) >
<!ELEMENT Date  (#PCDATA) >
<!ELEMENT Time  (#PCDATA) >
```

XML Document



Classical Components

- consist of

XML DTD (Schema)	architecture for "booking"
XML document instance	booking for S. Jones

- Two different languages

XML DTD (Schema)	Markup Syntax
XML Document	Instance Syntax

INFOTAKT information design

Conclusion
Languages
Concepts
Introduction

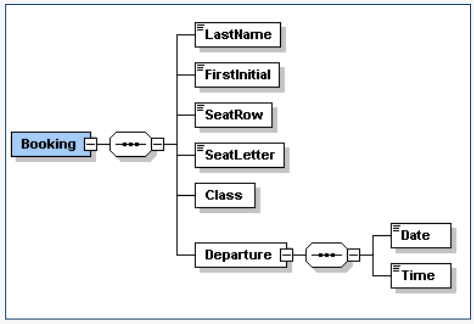
XML once more

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v3.5 NT (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="Booking">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="LastName" type="xsd:string"/>
                <xsd:element name="FirstInitial" type="xsd:string"/>
                <xsd:element name="SeatRow">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:positiveInteger">
                            <xsd:maxInclusive value="60"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="SeatLetter" type="xsd:string"/>
                <xsd:element name="Class">
                    <xsd:complexType>
                        <xsd:attribute name="level" use="default" value="Economy">
                            <xsd:simpleType>
                                <xsd:restriction base="xsd:NMTOKEN">
                                    <xsd:enumeration value="Economy"/>
                                    <xsd:enumeration value="Business"/>
                                    <xsd:enumeration value="First"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:attribute>
                    </xsd:complexType>
                </xsd:element>
```

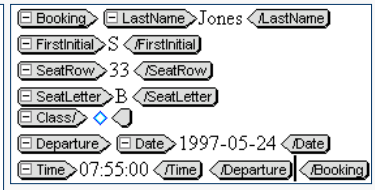
What´s new?

What´s new?

XML Schema



XML document



New syntax & New vocabulary for Schemas

- Schema and document "speak the same language"!
  - Instance Syntax
- More features for defining constraints

What are XML schemas?

- Define and describe a class of XML documents by using the following constructs:
  - data types
  - elements and their content
  - attributes and their values
  - default values
  - documentation vocabulary
- XML Schema: Part 2 "Data types" provides a robust and extensible data type system for document authors and application writers

Goals

- "express syntactic, structural, and value constraints"
- "a useful level of constraint checking to be described and validated for a wide spectrum of XML applications"

Scenarios

- Editing and Publishing
  - Re-used structures for several document types
  - Complex structure within documents
  - Controlled Editing
- E-Commerce
  - Defining standards for data exchange
- Applications
  - Communication between applications
  - Interfaces between applications

INFOTAKT information design

Introduction
Concepts
Languages
Conclusion

Features

- DTD features and much more
- Structured schema documentation
- Constraints for content
- Flexible content models
- Re-Use and modularity
- New schemas by composition

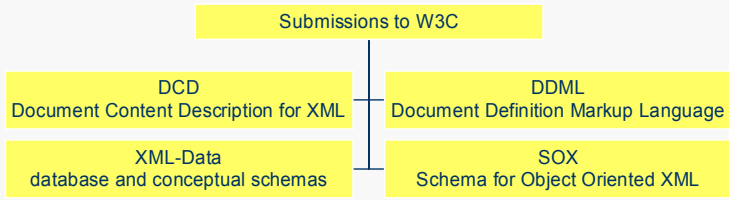
3 XML Schema Languages

History and Status Quo
XML Schema Language in Detail

History and Status Quo

Which Schema Languages did / do exist?

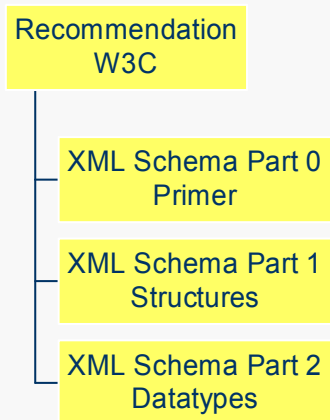
XML Schema Languages Phase 1



INFOTAKT information design

Introduction
Concepts
Languages
Conclusion

XML Schema Language Phase 2



XML Schema Languages Phase 3

Schema Languages

- XML Schema W3C
- RELAX JIS TR
- and others...

- XDR (XML Data Reduced)
- TREX by James Clark

XML Schema Language

Introduction

3.1 Structures

XML Schema Part 1

Overview

- Type definitions
- Element and attribute declarations
- Building content models: Basic features (Connectors, occurrence, re-use of content models)
- Building content models: Advanced features (deriving types, equivalence, uniqueness and relations, "Any")
- Composition of schemas (Namespaces, including / importing schemas)

a) Type Definitions

Simple and complex types

Simple and Complex Types

- Simple Types
  - no elements in the content model
  - no attributes
- Complex Types
  - containing elements and attributes

Simple Type

- Simple Type Definition (in schema "booking"):

```
<xsd:simpleType name="flight_number_type">
   <xsd:restriction base="xsd:string">
         <xsd:pattern value="[A-Z]{2} \d{4}"/>
   </xsd:restriction>
</xsd:simpleType>
```

- Use (in an element declaration of schema "booking"):

```
<xsd:element name="flight_number"
         type="flight_number_type"/>
```


Complex Type

- Complex Type Definition:

```
<xsd:complexType name="address_type">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="city" type="xsd:string"/>
    <xsd:element name="zip" type="xsd:number"/>
    <xsd:element ref="comment" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

- Use (in an element declaration):

```
<xsd:element name="address" type="address_type"/>
```

b) Element and Attribute Declarations

Element Declaration (1)

```
<xsd:element name="booking">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="flight_number"
                   type="flight_number_type"/>
      <xsd:element ref="name"/>
      <xsd:element ref="address"/>
      <xsd:element ref="seat"/>
    </xsd:sequence>
    <xsd:attribute name="booking_code"
                   type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

Element Declaration (2)

```
<xsd:element name="seat">
 <xsd:complexType>
   <xsd:choice>
     <xsd:element ref="reservation"/>
     <xsd:sequence>
       <xsd:element ref="row"/>
       <xsd:element ref="seat_number"/>
     </xsd:sequence>
   </xsd:choice>
 </xsd:complexType>
</xsd:element>
```


Attribute Declaration

- declaration:

```
<xsd:element name="booking">
  <xsd:complexType>
    ...
    <xsd:attribute ref="booking_code"/>
  </xsd:complexType>
</xsd:element>
```

- use within a content model:

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

Global and Local Declarations

- Elements and attributes can be declared globally or locally
  - globally: they are declared on the schema level and can be used in every content model of the schema
  - locally: they are declared within another element declaration and can be used only in this content model.

Global Declaration

- globally declared element:

```
<xsd:element name="address"
             type="address_type"/>
```

- used within another declaration:

```
<xsd:element name="booking">
  <xsd:complexType>
    ...
    <xsd:element ref="address"/>
    ...
  </xsd:complexType>
</xsd:element>
```

Local Declaration

- Locally declared element:

```
<xsd:element name="booking">
  <xsd:complexType>
    ...
    <xsd:element name="flight_number"
                 type="flight_number_type"/>
    ...
  </xsd:complexType>
</xsd:element>
```

- used only within this content model!

Local and Global Type Definitions

- Also simple and complex types can be defined globally and locally:
  - local declaration: see example above
  - global declaration

```
<xsd:complexType name="address_type">
  ...
</xsd:complexType>
```

c) Building Content Models

Basic Features

Basic Features: Overview

- Connectors
- Occurrence constraint
- Re-Use (parts of) content models: Named model groups, attribute groups

Connectors

- Sequence (default connector)
- Choice
- All (simplified SGML's &-Connector)

```
<xsd:complexType name="address_type">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="city" type="xsd:string"/>
    <xsd:element name="zip" type="xsd:positiveInteger"/>
    <xsd:element ref="comment" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```


Occurrence Constraints

- Occurrence of Elements and Attributes:
  - minOccurs
  - maxOccurs
  - fixed
  - default

Occurrence Constraints: Example

```
<xsd:complexType name="person_name">
  <xsd:element name="title"
               minOccurs="0"/>
  <xsd:element name="first_name"
               minOccurs="0"
               maxOccurs="unbounded"/>
  <xsd:element name="last_name"/>
</xsd:complexType>
```


Named model group

- named model groups define a (part of a) content model
- they are defined on the schema level (not within a declaration)
- they can be (re-)used within declarations (similar to parameter entities in SGML and XML DTDs)

Named model group: Definition

```
<xsd:group name="seat_booking">
    <xsd:sequence>
        <xsd:element ref="row"/>
        <xsd:element ref="seat_number"/>
    <xsd:sequence>
</xsd:group>
```


Named model group: Use

- within an element declaration:

```
<xsd:element name="seat">
   <xsd:complexType>
         <xsd:choice>
               <xsd:element ref="reservation"/>
               <xsd:group ref="seat_booking"/>
         </xsd:choice>
   </xsd:complexType>
</xsd:element>
```

Attribute group: Definition

- Reusable attribute sets

```
<xsd:attributeGroup name="BookingCodeGroup">
   <xsd:attribute name="booking_code"
                  type="xsd:string"/>
   <xsd:attribute name="booking_code_internal"
                  type= "xsd:string"/>
</xsd:attributeGroup>
```


INFOTAKT information design

Introduction
Concepts
Languages
Conclusion

Attribute group: Use

```
<xsd:element name="booking">
  <xsd:complexType>
    <xsd:sequence>
    <xsd:element name="flight_number"
                 type="flight_number_type"/>
       ...
    </xsd:sequence>
    <xsd:attributeGroup ref="BookingCodeGroup"/>
  </xsd:complexType>
</xsd:element>
```

d) Building Content Models

Advanced Features

Advanced Features: Overview

- Derive new types
- Equivalence of elements
- Uniqueness constraint and relations
- Flexible content model: "Any"

Types (named content type)

- Definition:

```
<xsd:complexType name="person_name">
   <xsd:element name="title" minOccurs="0"/>
   <xsd:element name="first_name"
            minOccurs="0"
            maxOccurs="unbounded"/>
   <xsd:element name="surname"/>
</xsd:complexType>
```

- Use:

```
<xsd:element name="name" type="person_name"/>
```


Derived types

- Re-use for simpleTypes and complexTypes
  - By extension
  - By restriction

Deriving a Type by Extension

- declaration:

```
<xsd:complexType name="extended_name">
  <xsd:complexContent>
    <xsd:extension base="person_name">
      <xsd:attribute name="gender"
        <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="male"/>
            <xsd:enumeration value="female"/>
         </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexContent>
</xsd:complexType>
```


Deriving a Type by Restriction

- declaration

```
<xsd:complexType name="simple_name">
  <xsd:complexContent>
    <xsd:restriction base="person_name">
       <xsd:element name="title" maxOccurs="0"/>
       <xsd:element name="first_name" minOccurs="1"
                    maxOccurs="1"/>
    </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

Using derived types

- restricted type of example above

```
<xsd:element name="pupils" type="simple_name"/>
```

- extended type of example above

```
<xsd:element name="addressee"
             type="extended_name"/>
```


Abstract elements and Types

- Abstract elements or types
  - can't be used in an instance
  - only used for derivation
- In the document instance types can be substituted by their derivations
  - e.g. a "simple_name" type is valid wherever a "person_name" is valid

```
<xsd:complexType name="simple_name"
   abstract="true">
…
</xsd:complexType>
```

Substitution Groups

- allows elements to be substituted for other elements

```
<xsd:element name="personal_address"
             type="address_type"
             substitutionGroup="address"/>

<xsd:element name="company_address"
             type="address_type"
             substitutionGroup ="address"/>
```

- `<personal_address>` and `<company_address>` are allowed wherever `<address>` is allowed

Uniqueness of elements and attributes

```
<xsd:unique>
   <xsd:selector xpath="region/zip />
      <!-- defines the scope                -->
   <xsd:field xpath="@code"/>
   <xsd:field xpath="part/@number"/>
      <!-- attribute "code" and
           attribute number within part,
           within region/zip:
           combined they must build         -->
</xsd:unique>
```

Key and key references

```
<xsd:key name="pNumKey">
    <xsd:selector xpath="parts/part"/>
       <!-- defines the scope -->
    <xsd:field xpath="@number"/>
       <!-- unique and not nullable -->
</xsd:key>

<xsd:keyref name="pNumKeyRef" refer="pNumKey">
    <xsd:selector xpath="region/zip/part"/>
    <xsd:field xpath="@number"/>
       <!-- there must be a pNumKey with the
            same value (uniqueness not required) -->
</xsd:keyref>
```


Flexible content model: Any element, any attribute

- in general: any well-formed XML is permissible

```
<xsd:complexType name="comments">
   <xsd:any/>
</xsd:complexType>
```

- more detailed: any well-formed XML that belongs to a certain namespace (e.g. HTML)

```
<xsd:complexType name="comments">
   <xsd:any
          namespace="http://www.w3.org/1999/xhtml"/>
</xsd:complexType>
```

e) Composition of Schemas

Include Schemas
Namespaces
Import types

Including Schemas

- Within schemas other schemas can be included:

```
<xsd:schema xmlns=
          "http://www.w3.org/1999/XMLSchema" … >
   <xsd:include schemaLocation=
          "http://www.myschemas.com/address.xsd"/>

   <xsd:element name="booking">…</xsd:element>
   ...
</xsd:schema>
```

Namespaces

- Uniqueness of names (preventing name collisions)
- Composition of schemas
- Namespaces are used e.g. when types are imported from other schemas

Importing Types

- Type definitions of other schemas can be imported
- Example: "flight_number" which could be defined in "Flight_properties.xsd"

Import Types from another Schema

```
<xsd:schema
  xmlns=
    "http://www.w3.org/1999/XMLSchema"
  xmlns:flight=
    "http://www.my.com/flight_properties.xsd" … >

  <xsd:import
    namespace="http://www.my.com/address.xsd"/>

   …

</xsd:schema>
```


Use an Imported Type

```
<xsd:element name="booking">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="flight:flight_number"/>
      ...
    </xsd:sequence>
    <xsd:attribute name="booking_code"
                   type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

3.2 Data Types

XML Schema Part 2

Primitive and Derived Data Types

Primitive	e.g. string, boolean, float, ID, IDREF
Derived	e.g. integer, positiveInteger, time, date

Derived Data Types

- Definition

```
<xsd:simpleType name="positive-integer">
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:minInclusive value="100"/>
  </xsd:restriction>
</xsd:simpleType>
```

- Use

```
<xsd:attribute name="booking_number"
               type="positive-integer"/>
```


Constraining Facets

- length
- minLength
- maxLength
- pattern
- enumeration
- whiteSpace
- minExclusive, minInclusive
- maxExclusive, maxInclusive
- totalDigits
- fractionDigits

Examples

- Enumeration/min-max declarations
  - 3 or 5 or 9
  - 2 to 10
- Length
  - String, maxlength 3
- Regular expressions
  - \d{3}-\d{4} (e.g. 543-1267)

4 Conclusion

Summary, Scenarios

Summary

- More powerful than XML DTDs
- One syntax for documents and for schemas
- More flexible data modelling features

Using XML Schemas

- Constraints not only on the *content models*, but also on the *content*
  - e.g. technical documentation in general
- Intelligent Clients
  - e.g. online booking
  - e.g. external authors
- Standard Schemas for data exchange
  - e.g. product catalogs

Resources

XML Schemas, Applications, Namespaces

XML Schema Specifications (1)

- W3C Early Submissions
  - DDML
    - www.w3.org/TR/NOTE-ddml
  - DCD
    - www.w3.org/TR/NOTE-dcd
  - SOX
    - www.w3.org/TR/NOTE-SOX
  - XML-Data
    - www.w3.org/TR/1998/NOTE-XML-data

XML Schema Specifications (2)

- W3C Proposed Recommendation
  - XML Schema Part 0: Primer
    - www.w3.org/TR/xmlschema-0/
  - XML Schema Part 1: Structures
    - www.w3.org/TR/xmlschema-1/
  - XML Schema Part 2: Data types
    - www.w3.org/TR/xmlschema-2/
- Namespaces in XML
  - www.w3.org/TR/REC-xml-names

INFOTAKT information design

Introduction
Concepts
Languages
Conclusion

INFO TAKT

XML Schema Specifications (3)

- RELAX JIS TR
  - www.xml.gr.jp/relax/
- XDR (XML Data Reduced)
  - www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm
- TREX
  - www.coverpages.org./trex/trexAnnounce.html
  - www.thaiopensource/trex

XML Schema Specifications (3)

- XML Schema Requirements
  - www.w3.org/TR/NOTE-xml-schema-req
- Robin Cover
  - www.oasis-open.org/cover/schemas.html
- Misc
  - www.xml.org
  - www.xmlmag.com
  - www.biztalk.org

Applications

- XMLSpy
  - XML Schema Editor www.xmlspy.com
  - XML Editor with DTD and Schema Support
- XML Authority 1.x (Extensibility)
  - XML Schema Designer www.extensibility.com
- Parser
  - Xerces http://xml.apache.org/
  - Oracle XML Parser www.oracle.com

INFO TAKT

Thank you for your interest!