

XML Vorlesung ETHZ, Sommersemester 2006

XML Schema Teil II

Erik Wilde

16.5.2006

<http://dret.net/lectures/xml-ss06/>

16.5.2006

XML Vorlesung ETHZ SS 2006

1

Übersicht

- Identity Constraints
 - ID/IDREF in XML Schema
- Complex Type Derivation
 - Derivation by Restriction
 - Derivation by Extension

16.5.2006

XML Vorlesung ETHZ SS 2006

2

XML Schema Identity Constraints

- ID/IDREF(S) in DTDs
- Identity Constraints
 - Konsistenzbedingungen für Dokumente
- Uniqueness
 - Garantie der Eindeutigkeit
- Keys
 - Garantie von Eindeutigkeit und Existenz
- Key References
 - Garantie von Referenzen auf existierende Keys

16.5.2006

XML Vorlesung ETHZ SS 2006

3

ID/IDREF(S) in XML DTDs

- Referenzen von Elementen zu Elementen
- IDs werden als Attribute vergeben
 - sie müssen gültige XML Namen sein
 - sie müssen im XML Dokument eindeutig sein
- IDREF(S) werden als Attribute vergeben
 - sie müssen existierende IDs referenzieren
- keine Möglichkeit der eigenen Typisierung
 - Identifikation einzig über den Attributtyp ID
 - keine Integers als Keys möglich (nur XML Namen)
 - unabhängig vom Elementtyp

16.5.2006

XML Vorlesung ETHZ SS 2006

4

Identity Constraints

- Erweiterung des ID/IDREF Konzepts
 - aber IDREFS wird nicht unterstützt
- bieten eine ganze Reihe von Vorteilen
 - belassen den Typ der Keys
 - ermöglicht eine Typ-Validierung der Keys
 - ermöglicht verschiedene Key Typen, z.B. Integers/Strings
 - ermöglichen Keys über mehrere Felder
 - ermöglichen scoped Keys (d.h. kontext-abhängig)
 - beruhen auf Wertegleichheit (z.B. +002 = 2)
 - können auch auf Elemente angewendet werden
- sind etwas komplizierter zu benutzen

16.5.2006

XML Vorlesung ETHZ SS 2006

5

Deklaration von "ID" Typen

- eigene Typen für die IDs definieren
 - beliebige XML Schema Typen sind erlaubt
- eigene Typen für Listen von IDs sind möglich
 - allerdings nicht von den Identity Constraints unterstützt
 - Tests müssen von der Anwendung gemacht werden

```
<xs:simpleType name="idType">
  <xs:restriction base="xs:positiveInteger"/>
</xs:simpleType>
<xs:simpleType name="idListType">
  <xs:list itemType="idType"/>
</xs:simpleType>
```

16.5.2006

XML Vorlesung ETHZ SS 2006

6

Constraints Syntax

- enthalten in einer Element-Deklaration
 - definiert den Scope des Constraints
- je nach Typ wird ein Element verwendet
 - `<xs: unique>`, `<xs: key>`, `<xs: keyref>`
 - enthält einen eindeutigen Namen
 - `<xs: keyref>` referenziert einen Namen
- darin enthalten ist ein Selector
 - spezifiziert in XPath Syntax die relevanten Knoten
- und ein oder mehrere Felder
 - XPaths, die die eindeutigen Felder selektieren

16.5.2006

XML Vorlesung ETHZ SS 2006

7

Beispiel Uniqueness Constraint

- Kontext ist das Durchführung Element
 - enthält die Daten für eine Kursdurchführung

```
<xs: unique name="teilnehmerRefUnique">  
  <xs: selector xpath="kurssteilnehmer" />  
  <xs: field xpath="@teilnehmer" />  
</xs: unique>
```

- kontrolliert die Teilnehmer eines Kurses
 - kontrolliert die Eindeutigkeit einer Referenz
 - verhindert die Duplizierung von Einträgen

16.5.2006

XML Vorlesung ETHZ SS 2006

8

XPaths in Identity Constraints

- XML Schema definiert eine XPath Untermenge
 - definiert auf syntaktischer Basis
 - unterstützt die wenigsten der XPath Konstrukte
- XPaths in Identity Constraints sind
 - ein Location Path (oder Kombinationen mit |)
 - immer relative Ausdrücke
 - benutzen keine Prädikate
 - benutzen nur die Child oder Attribute Axis
 - benutzen XPath's *Abbreviated Syntax*
 - `child::` : weglassen und `attribute::` durch `@` ersetzen
 - fangen u.U. mit `..` an (alle Descendants)

16.5.2006

XML Vorlesung ETHZ SS 2006

9

Selectors und Fields

- dienen der Selektion für alle Constraints
- Selectors selektieren den Scope
 - relativ zum Scoping Element (das Element, in dem der Constraint steht)
 - selektieren eine Menge von Knoten
- Fields selektieren die Werte
 - relativ zu den Knoten des Selectors
 - selektieren immer höchstens einen Knoten
 - können Elemente oder Attribute selektieren
 - es können beliebig viele Fields angegeben werden
 - die Fields müssen Simple Content haben

16.5.2006

XML Vorlesung ETHZ SS 2006

10

Keys und Referenzen

- XML Schema berücksichtigt Datentypen
 - Werte mit unrelated Types sind niemals gleich
 - einer Instanz nicht unbedingt anzusehen
 - z.B. gilt `xs:integer 2` ist nicht gleich `xs:string 2`
 - aber `xs:integer 2` ist gleich `xs:integer +0002`
 - und `xs:string 2` ist nicht gleich `xs:string +0002`
 - Identity Constraints sind auf Instanzen definiert, und nicht auf ihren Typen
- d.h. am besten definiert man eigene Typen
 - z.B. Simple Type als Restriction von `xs:integer`
 - eigene Typen für Key und KeyRef

16.5.2006

XML Vorlesung ETHZ SS 2006

11

Uniqueness Constraints

- dienen der Eindeutigkeit
 - überprüfen, dass kein Wert zweimal vorkommt
 - können auch mehrere Fields umfassen
 - z.B. kann man Name/Vorname auf Eindeutigkeit testen
- können entweder Werte kontrollieren
 - keine zwei gleichen IDs vergeben
- oder auch Referenzen
 - praktisch für das Vermeiden von Duplikaten

16.5.2006

XML Vorlesung ETHZ SS 2006

12

Key Constraints

- Unique und müssen existieren
 - Erweiterung der Uniqueness Constraints
 - falls xs: key gegeben ist, dann auch xs: unique
- für Anwendungen, die Schlüssel brauchen
 - stellen Eindeutigkeit sicher
 - stellen Existenz sicher
- Anwendungen auch für andere Werte möglich
 - Duplikate vermeiden
 - es können mehrere Fields angegeben werden

16.5.2006

XML Vorlesung ETHZ SS 2006

13

Beispiel Key Constraint

- alle Teilnehmer haben eine eindeutige id
 - den Attributtyp sollte man sinnvoll setzen
 - z.B. sollte es als required gekennzeichnet werden

```
<xs:key name="teilnehmerKey">
  <xs:selector xpath="teilnehmende/teilnehmer"/>
  <xs:field xpath="@id"/>
</xs:key>
```

- der Constraint hätte auch im teilnehmende Element angegeben werden können
 - aber auf keinen Fall im teilnehmer Element!

16.5.2006

XML Vorlesung ETHZ SS 2006

14

Key Reference Constraints

- müssen auf existierende Keys verweisen
 - verwenden das refer Attribut in keyref
 - Keys müssen im gleichen Element oder einem der Kinder definiert sein
- die Referenz zeigt auf den verwendeten Key
 - dieser sollte eindeutig sein (unique/key)
- Selector+Field zeigen auf die Referenzen
 - analog zu unique/key
 - Selector identifiziert die Knoten
 - Field identifiziert von diesen ausgehend die Referenzen

16.5.2006

XML Vorlesung ETHZ SS 2006

15

Beispiel Key Reference

- Kursteilnehmer müssen existieren
 - referenziert einen Key Constraint
 - der muss im gleichen oder Kinderelementen existieren
 - Selektor/Field geben die Referenz an

```
<xs:keyref name="teilnehmerRef" refer="teilnehmerKey">  
  <xs:selector xpath="kurse/kurs/durchführung/kursteilnehmer"/>  
  <xs:field xpath="@teilnehmer"/>  
</xs:keyref>
```

16.5.2006

XML Vorlesung ETHZ SS 2006

16

XML Schema Identity Constraints

- Identity Constraints sind Einschränkungen
 - genauere Spezifikation des Inhalts
 - Keys/Referenzen nur eine mögliche Anwendung
- genauere Spezifikation des Schemas
 - prüfen deklarativ auf semantische Probleme
- nicht alles kann angegeben werden
 - keine kausalen Abhängigkeiten
 - programmierte Tests sind immer noch notwendig
- leider keine Verbindung mit dem Typ-Konzept
 - Identity Constraints verwenden keine Typ-Namen

16.5.2006

XML Vorlesung ETHZ SS 2006

17

Complex Type Derivation

- was ist Complex Type Derivation?
- Restriction und Extension
- Derivation by Extension
- Derivation by Restriction
- Type Substitution
- Kontrolle über Type Derivation und Type Substitution

16.5.2006

XML Vorlesung ETHZ SS 2006

18

Was ist "Type Derivation"?

- abgeleitete Typen (*derived Types*) sind Typen, die auf einem Basis-Typ aufbauen und diesen entweder erweitern oder einschränken
- Beispiel: Modellieren von Person, Teilnehmer und Referent
 - sowohl ein Teilnehmer wie auch ein Referent sind Personen
 - aber sie haben wahrscheinlich zusätzliche Eigenschaften, die sie beschreiben

Type Extension und Restriction

- Complex Type Extension (Erweiterung):
 - bestehender Typ wird mit zusätzlichen Attributen und/oder zusätzlichen Child Elements erweitert
 - Werte des Basis-Typs sind nicht unbedingt auch gültige Werte des erweiterten Typs
- Complex Type Restriction (Einschränkung):
 - bestehender Typ wird eingeschränkt
 - Wertebereich ist Untermenge des Basis-Typs
 - alle Werte des eingeschränkten Typs gehören auch in den Wertebereich des Basis-Typs
- Complex Types können nicht gleichzeitig erweitert und eingeschränkt werden (nur in 2 Schritten)

Complex Type Extension

- Complex Types können
 - mit zusätzlichen Attributen erweitert werden
 - und/oder über das Content Model erweitert werden

16.5.2006

XML Vorlesung ETHZ SS 2006

21

Erlaubte Typenerweiterungen

		Base Type					
		Simple Type	Complex Type				
			Simple Content	Complex Content			
			Element only	Mixed	Empty		
Derived Type	Simple Type						
	Complex Type	Simple Content	Ja (1)	Ja			
		Complex Content	Element-only			Ja (2)	Ja (4)
			Mixed			Ja (3)	Ja (5)
	Empty				Ja (6)		

16.5.2006

XML Vorlesung ETHZ SS 2006

22

Extension: Attribute

- alle Attribute vom Basis-Typ werden an den erweiterten Typ weitervererbt
- Attribute des Basis-Typs müssen und dürfen nicht wiederholt werden im erweiterten Typ
- Attribute des Basis-Typs können im erweiterten Typ weder gelöscht noch modifiziert werden

16.5.2006

XML Vorlesung ETHZ SS 2006

23

Simple Content Extension (1)

- einziger Nutzen: Attribute hinzufügen
- Beispiel: Complex Type mit Simple Content (integer) wird mit Attribut erweitert

```
<xs:complexType name="TeilnehmerAnzahlTyp">
  <xs:simpleContent>
    <xs:extension base="xs:integer">
      <xs:attribute name="Stand"
        type="xs:date"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<TeilnehmerAnzahl Stand="2002-03-12">20</TeilnehmerAnzahl >
```

16.5.2006

XML Vorlesung ETHZ SS 2006

24

Complex Content Extension (2)

- Element-only Content:
 - Content Model kann erweitert werden
 - zusätzliche Child Elemente am Ende des Content Models hinzufügen
 - braucht eine sequence oder eine choice Model Group
 - nicht erlaubt mit all Model Group (all nur alleine erlaubt)
 - das Content Model vom Basis-Typ muss nicht kopiert werden
 - beide Content Models werden implizit mit einer sequence Group verbunden
 - Attribute können hinzugefügt werden
 - aber existierende Attribute können nicht verändert oder gelöscht werden

16.5.2006

XML Vorlesung ETHZ SS 2006

25

Complex Content Extension (2)

- Element-only Content: Beispiel

```

<xs:complexType name="referentTyp">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Vorname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="eitelReferentTyp">
  <xs:complexContent>
    <xs:extension base="referentTyp">
      <xs:choice>
        <xs:element name="Titel" type="xs:string"/>
        <xs:element name="Alter" type="xs:integer"/>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

16.5.2006

XML Vorlesung ETHZ SS 2006

26

Complex Content Extension (3)

- Mixed Content:
 - abgeleiteter Typ muss auch Mixed Content haben

```

<xs:complexType name="anmeldeTyp" mixed="true">
  <xs:sequence>
    <xs:element name="Kurs" type="xs:string"/>
    <xs:element name="Teilnehmer" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="erweiterterAnmeldeTyp" mixed="true">
  <xs:complexContent>
    <xs:extension base="anmeldeTyp">
      <xs:sequence>
        <xs:element name="Alter" type="xs:integer"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

16.5.2006

XML Vorlesung ETHZ SS 2006

27

Complex Content Extension (4, 5, 6)

- Empty Content:
 - Complex Type mit Empty Content kann erweitert werden, indem Attribute oder Child Elements hinzugefügt werden
 - Empty Content plus Child Elemente wird Element-only (4)
 - Empty Content plus Child Elemente und Zeichenkette zwischen Start- und Endtag wird Mixed (5)
 - Empty Content plus zusätzliche Attribute bleibt Empty Content (6)

16.5.2006

XML Vorlesung ETHZ SS 2006

28

Complex Content Extension (4)

- Empty Content Beispiel (4):

```
<xs:complexType name="IetzterStand">
  <xs:attribute name="Datum" type="xs:date"/>
</xs:complexType>
<xs:complexType name="IetzterStandmitElement">
  <xs:complexContent>
    <xs:extension base="IetzterStand">
      <xs:sequence>
        <xs:element name="Element"
          type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

16.5.2006

XML Vorlesung ETHZ SS 2006

29

Complex Type Restriction

- Complex Types können eingeschränkt werden
 - indem das Content Model eingeschränkt wird
 - indem Attribute eliminiert oder eingeschränkt werden
- alle Elemente, die einem eingeschränkten Typ angehören, sind auch gültige Elemente des Basistyps

16.5.2006

XML Vorlesung ETHZ SS 2006

30

Erlaubte Typeinschränkungen

		Base Type						
		Simple Type	Complex Type					
			Simple Content	Complex Content				
			Element only	Mixed	Empty			
* falls alle Child Elements optional sind								
Derived Type	Simple Type		Ja					
	Complex Type	Simple Content		Ja (1)		Ja* (3)		
		Complex Content	Element-only			Ja (2)	Ja (3)	
			Mixed				Ja (3)	
			Empty			Ja*	Ja* (3)	Ja (4)

16.5.2006

XML Vorlesung ETHZ SS 2006

31

Restriction: Attributes

- alle Attribute werden vererbt
- davon müssen im abgeleiteten Typ nur diejenigen Attribute wieder deklariert werden, die entweder eingeschränkt oder eliminiert werden
- erlaubte Attributeinschränkungen sind:
 - Typeinschränkung
 - Hinzufügen, Ändern oder Löschen eines Default-Wertes
 - einen Wert für fixed hinzufügen, wenn im Basis-Typ noch keiner vorhanden ist
 - optional auf required ändern
 - optional auf prohibited ändern

16.5.2006

XML Vorlesung ETHZ SS 2006

32

Simple Content Restriction (1)

- Nutzen: Simple Content und/oder Attribute des entsprechenden komplexen Typs können eingeschränkt werden
- beispielsweise kann der Wertebereich von Integer auf einen Teilbereich [10. . 30] eingeschränkt werden
 - analog zu Simple Type Restrictions durch Facets
- oder ein Attribut kann z.B. von optional auf required gesetzt werden etc.

16.5.2006

XML Vorlesung ETHZ SS 2006

33

Simple Content Restriction (1)

- Beispiel:

```

<xs:complexType name="teilnehmerAnzahlTyp">
  <xs:simpleContent>
    <xs:extension base="xs:integer" />
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="eingeschaenkterTeilnehmerAnzahlTyp">
  <xs:simpleContent>
    <xs:restriction base="teilnehmerAnzahlTyp">
      <xs:minInclusive value="10" />
      <xs:maxInclusive value="30" />
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>

```

16.5.2006

XML Vorlesung ETHZ SS 2006

34

Complex Content Restriction (2)

- Element-only Content:
 - Content Model kann eingeschränkt werden
 - der zu vererbende Teil des Content Models vom Basis-Typ muss in den eingeschränkten Typ kopiert werden
 - alle Instanzen des eingeschränkten Typs müssen auch gültige Instanzen des Basistyps sein
 - falls alle Child Elements des Basistyps mit Element-only Content optional sind, kann daraus auch ein komplexer Typ mit Empty-Content abgeleitet werden
 - Grundregel: falls Sie eine zulässige Instanz des abgeleiteten Typs finden, die nicht auch im Basistyp gültig ist, ist die Restriction nicht erlaubt

16.5.2006

XML Vorlesung ETHZ SS 2006

35

Complex Content Restriction (2)

- Element-only Content Beispiel:

```

<xs:complexType name="eIerReferentTyp">
  <xs:sequence>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Vorname" type="xs:string"/>
    <xs:element name="Titel" type="xs:string" minOccurs="0"/>
    <xs:element name="Alter" type="xs:integer" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="referentTyp">
  <xs:complexContent>
    <xs:restriction base="eIerReferentTyp">
      <xs:sequence>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Vorname" type="xs:string"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

```

16.5.2006

XML Vorlesung ETHZ SS 2006

36

Complex Content Restriction (3)

- Mixed Content:
 - können auf einen neuen komplexen Typ mit mixed Content eingeschränkt werden oder auf einen komplexen Typ mit Element-only Content
 - falls der Content mixed bleibt, muss in der Typdeklaration des abgeleiteten Typs `mixed="true"` spezifiziert werden (wird nicht vererbt)
 - falls alle Child Elements des komplexen Typs mit Mixed Content optional sind, kann daraus auch ein komplexer Typ mit Empty-Content oder mit Simple Content abgeleitet werden

16.5.2006

XML Vorlesung ETHZ SS 2006

37

Complex Content Restriction (3)

- Mixed Content Beispiel:

```

<xs:complexType name="erweiterterAnmeldeTyp" mixed="true">
  <xs:sequence>
    <xs:element name="Kurs" type="xs:string"/>
    <xs:element name="Teilnehmer" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="anmeldeTyp" mixed="false">
  <xs:complexContent>
    <xs:restriction base="erweiterterAnmeldeTyp">
      <xs:sequence>
        <xs:element name="Kurs" type="xs:string"/>
        <xs:element name="Teilnehmer" type="xs:string"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

```

16.5.2006

XML Vorlesung ETHZ SS 2006

38

Complex Content Restriction (4)

- Empty Content:
 - Einschränkung bezieht sich – zwangsläufig – auf die Attribute (siehe Folie *Complex Type Restriction: Attributes*)
 - der abgeleitete Complex Type hat immer Empty-Content

Kontrolle Type Derivation

- die Ableitung von Typen kann mit folgenden Attributen kontrolliert werden:
 - Attribut `final` in `complexType`-Deklaration:
 - `#all`: Typ kann weder erweitert noch eingeschränkt werden
 - d.h. keinerlei Typableitungen sind mehr möglich
 - `restriction`: Typ kann nicht eingeschränkt werden
 - `extension`: Typ kann nicht erweitert werden
 - Attribut `abstract` in `complexType`-Deklaration:
 - Typ darf nicht instanziiert werden

Zusammenfassung

- es gibt folgende Content Types für komplexe Typen:
 - Simple Content (Simple Type Zeichenkette zwischen Tags)
 - Complex Content:
 - Element-only Content (nur Child Elemente)
 - Mixed Content (Zeichenkette und Child Elemente)
 - Empty Content (kein Content, höchstens Attribut)
- Complex Types können auf drei Arten definiert werden:
 - ohne Type Derivation:
 - Group (all, choice, sequence) plus Attribute
 - ein einzelnes simpleContent Child Element:
 - Ableiten eines komplexen Typs von (a) einem Simple Type oder (b) einem komplexen Typ mit Simple Content
 - ein einzelnes complexContent Child Element:
 - Ableiten eines komplexen Typs von einem anderen komplexen Typ