XML SCHEMA
Prezentare

1. Despre XML Schema – scurta istorie

XML Schema a inceput ca o initiativa a companiei Microsoft. Intre timp insa, consortiul W3C a preluat aceasta initiativa si a dezvoltat un set mai larg de cerinte si caracteristici pentru documente ce descriu tipuri de documente. Acestea sunt reunite sub numele de **Document Content Description** si informatii despre acest proiect sunt disponibile la http://w3.org/TR/NOTE-dcd. XML Schema a devenit o recomandare W3C pe data de 02 Mai 2001.

Spre deosebire de (sau in plus fata de) DTD-uri, XML Schema permite definirea regulilor si relatiilor intre elemente si atribute folosind un fisier XML (cu alte cuvinte, o schema XML este descrisa intr-un fisier XML). In plus, se adauga support pentru *spatii de nume* (namespaces), tipuri de date si caracteristici mai avansate cum ar fi *constrangerile* (constraints).

XML Schema este o alternativa la DTD care este bazata pe standardul XML. Ca si DTD, XML Schema este folosita pentru a descrie structura documentelor XML. Acest standard mai este cunoscut si sub numele de XML Schema Definition (XSD).

2. XML Schema vs. DTD

Inca de la primele utilizari ale XML-ului in aplicatii a devenit evident ca adeseori definirea tipurilor de documente folosind fisiere DTD are cateva neajunsuri, dintre care enumeram:

- Limbajul DTD foloseste o alta sintaxa nu cea de XML Schema ceea ce presupune ca:
  - Utilizatorii sa invete doua “limbaje” (desi nici unul nu este dificil)
  - Aplicatiile care folosesc validarea de DTD sa contina componente separate de procesare a DTD-urilor
- Sintaxa DTD este relativ restransa. In cadrul DTD ar fi dificil, de exemplu, de declarat un tip de document XML care contine rapoarte organizate pe luni si zile. Astfel, o regula evidenta ar fi ca un element *luna* ar trebui sa contina intre 28 si 31 de elemente de tip *zi*, lucru care se poate specifica in DTD insiruind de 28 do ori *zi* si apoi definindu-le pe restul ca fiind optionale (?). Un alt exemplu este acela de verificare a corectitudinii structurii CNP-urilor (numere alcatuite din 13 cifre). XML Schema face ca astfel de structuri sa poata fi definite mult mai usor.

- DTD-urile au o capacitate foarte limitata de tipuri de date (suporta doar 10 tipuri de date). Pentru a exemplifica, in DTD nu putem spune ca un atribut este de tip intreg pozitiv, ci doar il putem defini ca sir de caractere, identificator sau fragment de nume.
- DTD-urile suporta attribute identificatori si referinte catre acestia (ID si IDREF). Nu se pot insa trasa relatii clare cum ar fi: atributul *zi_ref* este o referinta catre un atribut de tip identificator din elementul *zi*.
- DTD-urile nu suporta spatii de nume (namespaces).

Aceste neajunsuri au condus la aparitia XML Schema care prezinta urmatoarele avantaje:

- Sunt scrise folosind aceeasi sintaxa ca si fisierele pe care le descrie, ceea ce presupune mai putina sintaxa de memorat.
- Suporta peste 44 de tipuri de date si permite, de asemenea, definirea propriilor tipuri de date.
- Sunt orientate pe obiecte (se poate restrictiona sau extinde un anumit tip, derivand un nou tip pe baza unuia vechi).
- Se pot defini mai multe elemente care sa aiba acelasi nume dar continut diferit.
- Suporta spatii de nume (namespaces).
- Sunt extensibile ceea ce permite utilizarea unei Scheme in cadrul alteia, se pot referi mai multe scheme in cadrul aceluiasi document si se pot crea tipuri de date proprii derivate din tipurile standard

3. Sintaxa

Pentru o mai usoara intelegere a XML Schema, vom utiliza un document XML pe care vom arata cum se defineste si utilizeaza o Schema.

```
<?xml version="1.0"?>
<student>
        <nume>Ion</nume>
        <prenume>George</prenume>
        <adresa>undeva in Romania</adresa>
        <grupa>352C1</grupa>
</student>
```

REFERIREA unei XML Schema intr-un document XML

Pentru a putea referi o XML Schema intr-un document XML se adauga cateva atribute elementului radacina a documentului:

Ex:

```
<student
        xmlns="http://www.acs.ro/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
        xs:schemaLocation="http://www.acs.ro/ student.xsd">
```

Aceste atribute au urmatoarea semnificatie:

- xmlns="http://www.acs.ro/" – defineste namespace-ul default, specificand validatorului ca toate elementele folosite in documentul XML sunt declarate in namespaceul de la adresa respectiva
- xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" – instanta Schemei utilizate pentru respectivul namespace

- xs:schemaLocation="http://www.acs.ro/ student.xsd" – defineste namespace-ul utilizat (http://www.acs.ro/) si locatia XML Schemei ce va fi folosita pentru namespace-ul respectiv (student.xsd)

STRUCTURA unei XML Schema

Orice XML Schema are ca radacina elementul <schema> care poata sa contina niste atribute:

Ex:

```
<?xml version="1.0"?>
<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.acs.ro/"
        xmlns="http://www.acs.ro/"
        elementFormDefault="qualified">
...
...
</xs:schema>
```

Aceste atribute au urmatoarea semnificatie:

- xmlns:xs="http://www.w3.org/2001/XMLSchema" – indica faptul ca elementele si tipurile de date utilizate in Schema provin din namespace-ul definit la adresa respectiva. Totodata, se mentioneaza si faptul ca pentru a folosi elementele si tipurile de date din acest namespace, ele trebuiesc prefixate cu xs (xs este un nume aleator pe care il dam noi)
- targetNamespace="http://www.acs.ro/" – indica faptul ca elementele definite de aceasta Schema (student, nume, prenume, adresa, grupa) provin din namespace-ul de la adresa respectiva
- xmlns="http://www.acs.ro/" – indica faptul ca namespace-ul default e cel de la adresa specificata
- elementFormDefault="qualified" – indica faptul ca fiecare element utilizat in documentul XML si care au fost definite in Schema trebuie sa fie calificate de namespace-ul respectiv

Observatie: Namespace-ul default este cel care nu are asociat nici un sufix.

In continuare vom prezenta intreaga schema a documentului XML definit mai sus si vom continua discutia pe baza acestei scheme:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.acs.ro/"
xmlns="http://www.acs.ro/"
elementFormDefault="qualified">
<xs:element name="student">
   <xs:complexType>
    <xs:sequence>
            <xs:element name="nume" type="xs:string"/>
            <xs:element name="prenume" type="xs:string"/>
            <xs:element name="adresa" type="xs:string"/>
            <xs:element name="grupa" type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
</xs:element>
</xs:schema>
```

Elementul student este definit ca avand un **tip complex** deoarece contine alte elemente. Celelalte elemente (nume, prenume, adresa, grupa) au **tipul simplu** pentru ca nu contin alte elemente in interiorul lor. Acestea pot contine numai text in interiorul lor, dar acest text poate fi de unul dintre tipurile existente in XML Schema sau poate fi de un tip definit de utilizator si i se pot impune anumite restrictii sau se poate specifica faptul ca datele trebuie sa respecte un anumit pattern.

DEFINIREA ELEMENTELOR SIMPLE se face folosind structura:

<element name="nume_element" type="tip_element"/>

In cazul in care namespace-ul default este “http://www.w3.org/2001/XMLSchema” sau <xs: element name="xxx" type="yyy"/> in cazul in care acest namespace este definit ca in schema de mai sus. Nume_element este numele elementului din documentul XML iar tip_element poate fi unul din tipurile definite in XML Schema (ex: string, decimal, integer, boolean, date, time).

Observatie: Daca “http://www.w3.org/2001/XMLSchema” nu este namespace-ul default, tipurile de date trebuiesc prefixate cu prefixul specific acestui namespace (ex: xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time).

Elementele pot avea valori default sau fixate:

<xs:element name="color" type="xs:string" default="red"/>
<xs:element name="color" type="xs:string" fixed="red"/>

In primul caz, daca lipseste continutul elementului, acesta va fi adaugat automat ca fiind “red”. In cel de-al doilea caz, daca continutul exista si este diferit de “red”, se va furniza o eroare, iar daca nu exista, se va adauga automat ca fiind “red”.

Elementele simple **NU** pot sa contina elemente. Daca un element are atribut/e, atunci acesta trebuie sa fie definit ca un element complex. In schimb, atributele sunt tot timpul definite ca fiind atribute simple.

DEFINIREA ATRIBUTELOR se face similar cu definirea elementelor simple, avand aceleasi semnificatii si aceeasi observatie:

<attribute name="nume_atribut" type="tip_atribut"/>

De asemenea, se pot asocia valori default sau fixate:

<xs:attribute name="lang" type="xs:string" default="EN"/>
<xs:attribute name="lang" type="xs:string" fixed="EN"/>

In plus fata de elemente, se poate specifica si daca respectivul atribut poate sau nu sa lipseasca. In mod default, atributul este optional, adica atributul poate sa lipseasca. Pentru a specifica faptul ca atributul este obligatoriu, se adauga atributul use:

<xs:attribute name="lang" type="xs:string" use="required"/>

Observatie: In momentul in care se specifica un anumit tip de date pentru un element sau pentru un atribut din XML, se impune o restrictie asupra continutului acestuia (de exemplu, daca tipul este „date”, se va genera o eroare in momentul in care se gaseste informatia „Buna ziua”). De asemenea, se pot adauga restrictii suplimentare asupra elementelor sau atributelor, restrictii care se numesc fatete (facets).

Restrictiile sunt folosite pentru a defini acceptate pentru elementele si atributele XML. Restrictiile pot fi de maimulte feluri:

- **Restrictii asupra valorilor continute in interiorul tagurilor**

Folosind tipuri implicite	Folosind tipuri explicite	Explicatie
<xs:element name="varsta"> <xs:simpleType> <xs:restriction base="xs:integer"> <xs:minInclusive value="0"/> <xs:maxInclusive value="80"/> </xs:restriction> </xs:simpleType> </xs:element>	<xs:element name="varsta" type="tip_v"> <xs:simpleType name="tip_v"> <xs:restriction base="xs:integer"> <xs:minInclusive value="0"/> <xs:maxInclusive value="80"/> </xs:restriction> </xs:simpleType> </xs:element>	Impunerea faptului ca elementul „varsta” sa aiba valori cuprinse intre 0 si 80

Observatie: Cele 2 expresii sunt echivalente din punct de vedere ar definirii elementului de tip varsta. In cazul celei de-a doua forme, tipul „tip_v” poate fi refolosit in cadrul altor elemente sau extins/restrictionat, deoarece nu este o parte a elementului varsta ca in primul caz.

- **Restrictii asupra seturilor de valori posibile in interiorul tagurilor**

Folosind tipuri implicite	Folosind tipuri explicite	Explicatie
<xs:element name="varsta"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:enumeration value="20"/> <xs:enumeration value="30"/> <xs:enumeration value="40"/> </xs:restriction> </xs:simpleType> </xs:element>	<xs:element name="varsta" type="tip_v"> <xs:simpleType name="tip_v"> <xs:restriction base="xs:string"> <xs:enumeration value="20"/> <xs:enumeration value="30"/> <xs:enumeration value="40"/> </xs:restriction> </xs:simpleType> </xs:element>	Impunerea faptului ca elementul „varsta” sa aiba una din valorile 20, 30 sau 40

- **Restrictii asupra seriilor de valori – folosind patterne**

Folosind tipuri implicite	Folosind tipuri explicite	Explicatie
<xs:element name="cifra"> <xs:simpleType> <xs:restriction base="xs:integer"> <xs:pattern value="[0-9]"/> </xs:restriction> </xs:simpleType> </xs:element>	<xs:element name="cifra" type="tip_c"> <xs:simpleType name="tip_c"> <xs:restriction base="xs:integer"> <xs:pattern value="[0-9]"/> </xs:restriction> </xs:simpleType> </xs:element>	Descrierea unei cifre ca fiind orice intreg cuprins intre 0 si 9
<xs:element name="numar"> <xs:simpleType> <xs:restriction base="xs:integer"> <xs:pattern value="0|[1-9]([0-9])*"/> </xs:restriction> </xs:simpleType> </xs:element>	<xs:element name="numar" type="tip_n"> <xs:simpleType name="tip_n"> <xs:restriction base="xs:integer"> <xs:pattern value="[1-9]([0-9])*"/> </xs:restriction> </xs:simpleType> </xs:element>	Descrierea unui numar ca fiind numarul 0 sau o insiruire de oricate cifre cu conditia ca prima cifra sa fie cuprinsa intre 1 si 9
<xs:element name="CNP"> <xs:simpleType> <xs:restriction base="xs:integer"> <xs:pattern value="(1|2)[0-9]{12}"/>	<xs:element name="CNP" type="tip_cnp"> <xs:simpleType name="tip_cnp">	Descrierea unui CNP ca fiind un numar ce incepe cu cifra 1 sau 2 si care are

</xs:restriction> </xs:simpleType> </xs:element>	<xs:restriction base="xs:integer"> <xs:pattern value="(1|2)[0-9]{12}"/> </xs:restriction> </xs:simpleType> </xs:element>	inca 12 cifre dupa aceasta prima cifra

- **Restrictii asupra caracterelor „whitespace”** – pentru a specifica modul in care sunt interpretate caracterele "whitespace" se foloseste restrictia whiteSpace. Aceasta poate lua 3 valori posibile: preserve, replace si collapse. In primul caz, caracterele whitespace (line feed, tab, space, si carriage return) sunt afisate asa cum sunt intatlnite. In cazul replace, acestea sunt inlocuite cu spatii, iar in ultimul caz caracterele whitespace sunt inlocuite cu un singur caracter spatiu.

preserve	**replace**	**collapse**
<xs:element name="address"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:whiteSpace value="preserve"/> </xs:restriction> </xs:simpleType> </xs:element>	<xs:element name="address"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:whiteSpace value="replace"/> </xs:restriction> </xs:simpleType> </xs:element>	<xs:element name="address"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:whiteSpace value="collapse"/> </xs:restriction> </xs:simpleType> </xs:element>

- **Restrictii supra lungimii** – pentru a limita lungimea valorii continutului unui element, se vor folosi restrictiile: length, maxLength, si minLength. In cazul restrictiei length, lungimea continutului este exact cea specificata de valoarea data lui length. In cazul maxLength/minLength, lungimea continutului textului trebuie sa fie mai mare/mica decat valoarea specificata**.**

Sintaxa	**Explicatie**
<xs:element name="password"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:length value="8"/> </xs:restriction> </xs:simpleType> </xs:element>	Campul "password" trebuie sa aiba exact 8 caractere
<xs:element name="password"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:minLength value="5"/> <xs:maxLength value="8"/> </xs:restriction> </xs:simpleType> </xs:element>	Campul "password" trebuie sa aiba intre 5 si 8 caractere

- **Restrictii pentru tipurile de date**

Constrangere	**Descriere**
enumeration	Defineste o lista de valori ce pot fi acceptate.
fractionDigits	Specifica numarul maxim de decimale permise. Trebuie sa fie mai mare sau egal cu zero.
length	Specifica numarul exact de caractere sau de itemuri dintr-o lista. Trebuie sa fie mai mare sau egal cu zero.
maxExclusive	Specifica limita superioara pentru valorile numerice. Valoarea trebuie sa fie

	strict mai mica decat valoarea specifica astfel.
maxInclusive	Specifica limita superioara pentru valorile numerice. Valoarea trebuie sa fie mai mica sau egala cu valoarea specifica astfel.
maxLength	Specifica numarul maxim de caractere sau de itemuri dintr-o lista. Trebuie sa fie mai mare sau egal cu zero.
minExclusive	Specifica limita inferioara pentru valorile numerice. Valoarea trebuie sa fie strict mai mare decat valoarea specifica astfel.
minInclusive	Specifica limita inferioara pentru valorile numerice. Valoarea trebuie sa fie mai mare sau egala cu valoarea specifica astfel.
minLength	Specifica numarul minim de caractere sau de itemuri dintr-o lista. Trebuie sa fie mai mare sau egal cu zero.
pattern	Specifica un patter pe care trebuiesa il respecte datele respective
totalDigits	Specifica numarul exact de digiti permisi. Trebuie sa fie mai mare sau egal cu zero
whiteSpace	Specifica cum sunt tratate whitespaceurile

Inainte de a arata cum se definesc **elemente complexe** trebuie mai intai sa specificam ce sunt acestea. Un element complex este un element complex care contine alte elemente sau/si atribute. Sunt 4 tipuri de elemente complexe si oricare din aceste tipuri poate sa aiba si atribute:

a) elemente ce contin numai alte elemente in interiorul lor – au numai copii, fara text (ex: <employee><firstname>John</firstname><lastname>Smith</lastname></employee>)
b) elemente vide – ce nu contin nimic in interiorul lor (ex: <product pid="1345"/>)
c) elemente care contin in interiorul lor numai text (ex: <mancare tip="desert">tort</mancare>)
d) elemente mixte – care contin in interiorul lor atat elemente cat si text (ex: <scrisoare>Stimate Domnule <nume>Ion</nume>, Scrisoarea dumneavoastra <sid>102</sid> va ajunge pe data <data>15-11-2008</data>.</scrisoare>)

DEFINIREA ELEMENTELOR COMPLEXE:

a) elemente ce au doar copii in componenta lor

Folosind tipuri implicite	**Folosind tipuri explicite**	**Explicatie**
<xs:element name="employee"> <xs:complexType> <xs:sequence> <xs:element name="firstname" type="xs:string"/> <xs:element name="lastname" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:element>	<xs:element name="employee" type="personinfo"/> <xs:complexType name="personinfo"> <xs:sequence> <xs:element name="firstname" type="xs:string"/> <xs:element name="lastname" type="xs:string"/> </xs:sequence> </xs:complexType>	Se specifica faptul ca elementul employee are doar doi copii (firstname si lastname) care trebuie sa apara strict in aceasta ordine datorita indicatorului <xs:sequence>. Folosirea tipului implicit il face inutilizat in alta parte spre deosebire de folosirea tipurilor explicite care permite refolosirea acestuia precum si crearea de noi elemente complexe pe baza acestuia

Exemplu de refolosire si extindere a tipului explicit „personinfo”:

Sintaxa	**Explicatie**
<xs:element name="employee" type="personinfo"/> <xs:element name="student" type="personinfo"/> <xs:element name="member" type="personinfo"/>	Tipul personinfo este utilizat atat la definirea elementului employee cat si la definirea elementelor student si member

<xs:complexType name="fullpersoninfo"> <xs:complexContent> <xs:extension base="personinfo"> <xs:sequence> <xs:element name="address" type="xs:string"/> <xs:element name="city" type="xs:string"/> <xs:element name="country" type="xs:string"/> </xs:sequence> </xs:extension> </xs:complexContent> </xs:complexType>	In acest fel s-a definit tipul fullpersoninfo care are un continut complex (<xs:complexContent>) si porneste de la tipul personinfo extinzandu-l (<xs:extension base="personinfo">) cu o secventa de alte elemente (<xs:sequence>… </xs:sequence>)

b) elemente vide (ex: <product pid="1345"/>) – pentru a defini un tip vid, trebuie sa definim un tip care sa contina numai elemente in interiorul sau, dar caruia sa nu ii adaugam, de fapt, nici un element. Astfel, el este definit ca fiind un tip complex, cu un continut complex, ceea ce arata faptul ca intentionam sa extindem sau sa restrangem continutul unui element de tip complex.

Sintaxa	**Sintaxa simplificata**	**Sintaxa simplificata folosind tipuri explicite**
<xs:element name="product"> <xs:complexType> <xs:complexContent> <xs:restriction base="xs:integer"> <xs:attribute name="prodid" type="xs:positiveInteger"/> </xs:restriction> </xs:complexContent> </xs:complexType> </xs:element>	<xs:element name="product"> <xs:complexType> <xs:attribute name="prodid" type="xs:positiveInteger"/> </xs:complexType> </xs:element>	<xs:element name="product" type="prodtype"/> <xs:complexType name="prodtype"> <xs:attribute name="prodid" type="xs:positiveInteger"/> </xs:complexType>

c) elemente care contin in interiorul lor numai text (ex: <mancare tip="desert">tort</mancare>) – acest tip contine numai continut simplu (text sau/si atribute). Pentru a defini un astfel de tip, se foloseste „simpleContent” si trebuie utilizata o extensie sau o restrictie.

Restrictie	**Extensie**
<xs:element name="nume"> <xs:complexType> <xs:simpleContent> <xs:restriction base="tip"> .... </xs:restriction> </xs:simpleContent> </xs:complexType> </xs:element>	<xs:element name="nume"> <xs:complexType> <xs:simpleContent> <xs:extension base="tip"> .... </xs:extension> </xs:simpleContent> </xs:complexType> </xs:element>

Observatie: Extensia/restrictia se foloseste pentru a extinde sau limita tipul de baza.

Exemplu:

Sintaxa	**Sintaxa cu tip explicit**
<xs:element name="mancare"> <xs:complexType> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="tip" type="xs:string" /> </xs:extension>	<xs:element name="mancare" type="tip_mancare"> <xs:complexType name="tip_mancare"> <xs:simpleContent> <xs:extension base="xs:string"> <xs:attribute name="tip" type="xs:string" />

</xs:simpleContent> </xs:complexType> </xs:element>	</xs:extension> </xs:simpleContent> </xs:complexType>

e) elemente mixte (ex: <scrisoare>Stimate Domnule <nume>Ion</nume>, Scrisoarea dumneavoastra <sid>102</sid> va ajunge pe data <data>15-11-2008</data>.</scrisoare>)

Folosind tipuri implicite	Folosind tipuri explicite	Explicatie
<xs:element name="scrisoare"> <xs:complexType mixed="true"> <xs:sequence> <xs:element name="nume" type="xs:string"/> <xs:element name="sid" type="xs:positiveInteger"/> <xs:element name="data" type="xs:date"/> </xs:sequence> </xs:complexType> </xs:element>	<xs:element name="scrisoare" type="tip_s"/> <xs:complexType name="tip_s" mixed="true"> <xs:sequence> <xs:element name="nume" type="xs:string"/> <xs:element name="sid" type="xs:positiveInteger"/> <xs:element name="data" type="xs:date"/> </xs:sequence> </xs:complexType>	Pentru a permite sa apara texte intre copiii elementului scrisoare, trebuie ca atributul „mixed” sa aiba valoarea „true”. Tagul <xs:sequence> specifica faptul ca elementele definite in cadrul ei (nume, sid, data) trebuie sa apara in ordinea specificata in elementul scrisoare.

INDICATORI – sunt sapte indicatori grupati in 3 categorii:

- **indicatori de ordine (Order indicators)**: all, choice, sequence – sunt utilizati pentru a specifica ordinea elementelor.
  - **All** – toti copii trebuie sa apara o singura data in cadrul elementului parinte, dar ca pot aparea in orice ordine.
  - **Choice** – orice copil poate sa apara in cadrul elementului parinte
  - **Sequence** – toti copiii trebuie sa apara in cadrul elementului parinte in ordinea specificata

All	Choice	Sequence
<xs:element name="person"> <xs:complexType> <**xs:all**> <xs:element name="firstname" type="xs:string"/> <xs:element name="lastname" type="xs:string"/> </**xs:all**> </xs:complexType> </xs:element>	<xs:element name="person"> <xs:complexType> <**xs:choice**> <xs:element name="employee" type="employee"/> <xs:element name="member" type="member"/> </**xs:choice**> </xs:complexType> </xs:element>	<xs:element name="person"> <xs:complexType> <**xs:sequence**> <xs:element name="firstname" type="xs:string"/> <xs:element name="lastname" type="xs:string"/> </**xs:sequence**> </xs:complexType> </xs:element>

- **indicatori de aparitie (Occurrence indicators)**: maxOccurs, minOccurs – sunt utilizati pentru a specifica de cate ori poate sa apara un element copil in cadrul parintelui sau
  - **maxOccurs** – specifica numarul maxim de aparitii a respectivului element
  - **minOccurs** – specifica numarul minim de aparitii a respectivului element

Sintaxa	Explicatie
<xs:element name="persoana"> <xs:complexType> <xs:sequence> <xs:element name="liceu" type="xs:string"/> <xs:element name="facultate" type="xs:string" maxOccurs="5" minOccurs="0"/>	Sintaxa defineste un element „persoana” care are 2 tipuri de copii: un copil „liceu”

</xs:sequence> </xs:complexType> </xs:element>	si un copil „facultate” care poate sa apara de un numar de ori cuprins intre 0 si 5

Observatii:

1) Pentru toti ceilalti indicatori (de ordine sau de grup), valorile implicite pentru maxOccurs si minOccurs sunt 1.
2) Pentru a permite unui element sa apara de oricate ori, se foloseste maxOccurs=”unbounded”.

- **indicatori de grup (Group indicators)**: Group name, attributeGroup name – folosite pentru a defini grupuri de elemente/atribute inrudite
  - **Group name** – se defineste folosind <group name="groupname">...</group>. In interiorul grupului trebuie definit unul din indicatorii de ordine. Dupa definirea grupului, acesta poate fi referit in alta structura:

Definirea unui grup de elemente	**Referirea acestui grup in alt element**
<xs:group name="persongroup"> <xs:sequence> <xs:element name="firstname" type="xs:string"/> <xs:element name="lastname" type="xs:string"/> <xs:element name="birthday" type="xs:date"/> </xs:sequence> </xs:group>	<xs:element name="person" type="personinfo"/> <xs:complexType name="personinfo"> <xs:sequence> <xs:group ref="persongroup"/> <xs:element name="country" type="xs:string"/> </xs:sequence> </xs:complexType>

  - attributeGroup name – se defineste folosind <attributeGroup name="groupname">...</attributeGroup>. Dupa definirea grupului, acesta poate fi referit in alta structura:

Definirea unui grup de atribute	**Referirea acestui grup in alta structura**
<xs:attributeGroup name="personattrgroup"> <xs:attribute name="firstname" type="xs:string"/> <xs:attribute name="lastname" type="xs:string"/> <xs:attribute name="birthday" type="xs:date"/> </xs:attributeGroup>	<xs:element name="person"> <xs:complexType> <xs:attributeGroup ref="personattrgroup"/> </xs:complexType> </xs:element>

Pentru a permite **extinderea Schemelor XML**, sintaxa permite folosirea unor elemente de tip whildcard (**<any> si <anyAttribute>**) ce permit imbogatirea Schemelor cu elemente/atribute a caror definitie nu este data in Schema respectiva.

Sintaxa	Explicatie
<xs:element name="person"> <xs:complexType> <xs:sequence> <xs:element name="firstname" type="xs:string"/> <xs:element name="lastname" type="xs:string"/> <xs:any minOccurs="0"/> </xs:sequence> </xs:complexType> </xs:element>	Prin folosirea elementului any, continutul elementului „person” poate fi extins cu un alt element, care poate sa nu fie declarat in Schema curenta. Astfel, daca se declara un element „child” intr-o alta schema si apoi se specifica in documentul XML ca se vor folosi ambele scheme, atunci elementul persoana poata ca contina ca si copil si elementul „child” si astfel documentul XML sa fie valid.

Pentru o mai buna intelegere a elementelor <any> si <anyAttribute>, se pot urmari exemplele de la adresele:

http://www.w3schools.com/schema/schema_complex_any.asp respectiv
http://www.w3schools.com/schema/schema_complex_anyattribute.asp

In unele cazuri, se doreste ca in documentul XML sa existe alternative. De exemplu, se poate defini un element adresa in care unul dintre campuri sa fie judetul, pentru locuitorii din provincie sau sectorul pentru locuitorii din capitala. Pentru astfel de situatii, XML Schema pune la dispozitie atributul **substitutionGroup**. Astfel se pot defini doua elemente in care unul sa fie un substitut pentru celalalt:

```
<xs:element name="judet" type="xs:string"/>
<xs:element name="sector" substitutionGroup="judet"/>
```

Astfel, daca avem o XML Schema care defineste o adresa in felul urmator:

```
<xs:element name="judet" type="xs:string"/>
<xs:element name="sector" substitutionGroup="judet"/>
<xs:complexType name="adresa">
  <xs:sequence>
    <xs:element ref="judet"/>
  </xs:sequence>
</xs:complexType>
```

Atunci, urmatorul document XML este valide:

```
<adress book>
     <adresa><judet>Calarasi</judet></adresa>
     <adresa><sector>trei</sector></adresa>
</adress book>
```

Observatii:

1) Tipul elementelor care se doresc a substitui trebuie sa fie acelasi sau, in cel mai rau caz, derivate din tipul de baza (care se doreste a putea fi substituit). Daca tipul este acelasi, atunci nu mai este nevoie sa fie definit tipul elementelor care vor substitui elementul de baza
2) Pentru a putea folosi substituirea este necesar ca elementele implicate in aceasta sa fie declarate global (adica sa fie copii directi ai elementului „schema”)
3) Este posibila impiedicarea incercarii de substitutie, prin folosirea atributului block care trebuie sa aiba valoarea "substitution". Astfel, daca in exemplul de mai sus, elementului judet i se asociaza block="substitution", atunci documentul XML nu mai e valid:

```
<xs:element name="judet" type="xs:string" block="substitution"/>
```

Tipuri de date:

- String – folosit pentru date ce au valori ce contin siruri de caractere. Din acest tip sunt derivate urmatoarele tipuride date: normalizedString, token, ENTITIES, ENTITY, ID, IDREF, IDREFS, language, Name, NCName, NMTOKEN, NMTOKENS, Qname
- Date – folosit pentru date ce contin informatii legate de timp sau de date. Din acest tip sunt derivate: date, time, dateTime, duration, gDay, gMonth, gMonthDay, gYear, gYearMonth

- Decimal - folosit pentru date ce au valori de tip numeric. Din acest tip sunt derivate: byte, decimal, int, integer, long, negativeInteger, nonNegativeInteger, nonPositiveInteger, positiveInteger, short, unsignedLong, unsignedInt, unsignedShort, unsignedByte
- Boolean
- Binare:
  - base64Binary (Base64-encoded binary data)
  - hexBinary (hexadecimal-encoded binary data)
- Altele: anyURI, double, float, NOTATION, Qname

4. Cazuri de utilizare

XML Schema poate fi utilizat oriunde se folosesc documente XML pentru a specifica structura acestora.

5. Tooluri

In prezent, exista o serie de unelte dezvoltate pentru XML Schema: XMLSpy/Altova, Stylus Studio 2007, oXygen XML, Xerces-C++ 2.8.0, xnsdoc 1.2 - XML Schema documentation generator samd. O lista mai completa se gaseste la adresa: http://www.w3.org/XML/Schema. In general, aceste instrumente ofera posibilitatea validarii Schemei XML, validarea documentului XML cu Schema asociata, generarea automata a XML Schema pe baza unui XML, precum si conversia DTD-urilor in XML Schema.

6. Concluzii

XML Schema este un standard utilizat pentru definirea structurii fisierelor XML la fel ca si DTD-ul. Din motive prezentate in capitolul al doilea, credem ca XML Schema tinde sa inlocuiasca in viitor DTD-ul, reprezentand principala metoda de a defini structura acestuia. De asemenea, s-a observat si faptul ca XML Schema este un limbaj foarte bogat si mai ales preocuparea dezvoltatorilor acestui limbaj pentru reutilizare si extensie.

7. Bibliografie

http://www.w3.org/2001/03/webdata/xsv - validator XML Schema
http://www.validome.org/grammar/ - validator XML Schema
http://www.w3.org/TR/xmlschema11-1/
http://www.w3.org/XML/Schema
http://www.w3schools.com/schema/default.asp - tutorial XML Schema
http://www.w3.org/TR/xmlschema-0/
http://www.w3.org/TR/xmlschema-1/
http://www.w3.org/TR/xmlschema-2/
http://www.xfront.com/xml-schema.html - tutorial XML Schema
www.utdallas.edu/~lkhan/Fall2003/**xml-schema**_2.pdf - tutorial XLM Schema