ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №4

DTD схема XML документа

Виконав:

Ст. Прізвище Ім’я

Група ……..

2022

Тема: DTD схема XML документа.

Мета роботи: практичне ознайомлення з конструкціями DTD схеми XML документа, її створенням і використанням.

Теоретичний матеріал

Основними поняттями XML є *коректність (well formed)* та *валідність (valid):*

- **коректний** документ відповідає всім синтаксичним правилам XML. Документ, що не є коректним, не може називатись XML-документом.
- документ називається **валідним** *(valid)* або ***дійсним***, якщо він є коректним і містить посилання на граматичні правила та повністю відповідає обмеженням, вказаним у цих правилах описаних у схемі - DTD, XML Schema або RELAX NG.

Отже, є кілька способів визначення елементів, які використовуються для представлення даних в XML документі.

Першим було використання схеми **DTD** (Document Type Definition), яка дуже схожа на DTD для метамови SGML. DTD визначає елементи, які можуть з'являтися в XML-документі, порядок їх відображення, як вони можуть бути вкладені один у одного, та інші основні деталі структури документа XML.

Інший метод полягає у використанні **XML Schema**. Вона крім того, що визначає всі структури документів, як і в DTD, також ще може визначати **типи** даних і **більш складні** правила, ніж може DTD. W3C розробив специфікацію схеми XML через кілька років після оригінальної специфікації XML.

DTD (Document Type Definition)

Розглянемо зразок XML документа, що описує поштову адресу США:

```
1  <address>
2    <name>
3      <title>Mrs.</title>
4      <first-name>Mary</first-name>
5      <last-name>McGoon</last-name>
6    </name>
7    <street>1401 Main Street</street>
8    <city>Anytown</city>
9    <state>NC</state>
10   <postal-code>34829</postal-code>
11 </address>
```

DTD схема дозволяє задати структуру XML документа.

Наступний приклад DTD описує структуру XML документа поштової адреси:

```
1  <!-- address.dtd -->
2  <!ELEMENT address (name, street, city, state, postal-code)>
3  <!ELEMENT name (title?, first-name, last-name)>
4  <!ELEMENT title (#PCDATA)>
5  <!ELEMENT first-name (#PCDATA)>
6  <!ELEMENT last-name (#PCDATA)>
7  <!ELEMENT street (#PCDATA)>
8  <!ELEMENT city (#PCDATA)>
9  <!ELEMENT state (#PCDATA)>
10 <!ELEMENT postal-code (#PCDATA)>
```

В даній DTD визначаються всі елементи, використані у XML документі поштової адреси. А саме - три основні речі:

- <*address*> елемент містить <name>, <street>, <city>, <state> і <postal-code>. Усі ці елементи *повинні* бути присутні, і вони повинні вказуватися *у зазначеному порядку*.
- <*name*> елемент містить необов'язковий <title> елемент (*знак питання* означає, що заголовок є *необов'язковим*), а потім <first-name> і <last-name> елементи.
- Усі інші елементи містять текст.
  #PCDATA означає, що парсер здійснюватиме аналіз даних символів - ви не можете включити інший елемент до цих елементів, вони мають містити лише текст. Вміст елементів, визначених, як #CDATA парсером не аналізується.

Хоча DTD схема досить проста, вона дає зрозуміти, які поєднання елементів є допустимими. Наприклад, адресний документ, який містить <postal-code> елемент перед <state> елементом, *не є допустимим*, і жоден документ, який не має <last-name> елемента теж.

Порожні елементи оголошуються за допомогою ключового слова EMPTY:

<!ELEMENT emp EMPTY> визначає в XML документі `<emp />`

Елементи, оголошені за допомогою ключового слова ANY, можуть містити будь-яку комбінацію аналізованих даних:

<!ELEMENT note ANY>

Також потрібно зауважити, що синтаксис DTD *відрізняється* від звичайного синтаксису XML (на відміну від XML Schema, яка сама по собі є XML). Незважаючи на різний синтаксис для DTD, її все одно можна помістити в XML документ.
Якщо DTD оголошено всередині файлу XML, його потрібно загорнути всередину визначення <! DOCTYPE>:

```
<!DOCTYPE НазваКореневогоЕлементаXML […]>
```

Якщо DTD оголошено у зовнішньому файлі, визначення <! DOCTYPE> повинно містити посилання на файл DTD:

```
<!DOCTYPE note SYSTEM "note.dtd">
```

Отже наш XML документ разом з DTD схемою матиме вигляд

```
1  <!DOCTYPE address [
2  <!ELEMENT address (name, street, city, state, postal-code)>
3  <!ELEMENT name (title?, first-name, last-name)>
4  <!ELEMENT title (#PCDATA)>
5  <!ELEMENT first-name (#PCDATA)>
6  <!ELEMENT last-name (#PCDATA)>
7  <!ELEMENT street (#PCDATA)>
8  <!ELEMENT city (#PCDATA)>
9  <!ELEMENT state (#PCDATA)>
10 <!ELEMENT postal-code (#PCDATA)>]>
11 <address>
12 <name>
13 <title>Mrs.</title>
14 <first-name>Mary</first-name>
15 <last-name>McGoon</last-name>
16 </name>
17 <street>1401 Main Street</street>
18 <city>Anytown</city>
19 <state>NC</state>
20 <postal-code>34829</postal-code>
21 </address>
```

Символи в DTD

Існує кілька символів, які використовуються в DTD, щоб вказати, як часто (або чи) щось може з’являтися в XML-документі. Наприклад:

<!ELEMENT address (name, city, state)>
<address> елемент *повинен* містити *по одному* <name>, <city> і <state> елементів у цьому ж порядку. Усі елементи необхідні. Кома розділяє список елементів.

<!ELEMENT name (title**?**, first-name, last-name)>
Знак `?` означає, що <name> елемент містить *необов'язковий* <title> елемент, за яким слідують обов'язкові <first-name> та <last-name> елементи.

Знак *питання* вказує, що необов’язковий елемент може з’явитися один раз або зовсім не з'явитися.

<!ELEMENT addressbook (address**+**)>
<addressbook> елемент містить один або кілька <address>елементів. Ви можете мати стільки <address> елементів, скільки вам потрібно, але має бути принаймні один.

<!ELEMENT private-addresses (address*****)>
<private-addresses> елемент містить нуль або більше <address>елементів.
`*` вказує, що елемент може з’являтися будь-яку кількість разів, включаючи нуль.

<!ELEMENT name (title?, first-name, (middle-initial | middle-name)?, last-name)>
<name> елемент містить необов'язковий <title>елемент, за яким слідує <first-name> елемент, можливо, з подальшим або з <middle-initial> або <middle-name> елементом, за яким слідує <last-name> елемент. Іншими словами, обидва <middle-initial> і <middle-name> *необов'язкові*, і ви можете мати лише одне з двох.

Вертикальна риска позначає перелік варіантів; можна вибрати лише один елемент зі списку. Також цей приклад використовує дужки для групування певних елементів, а також знак питання до групи.

<!ELEMENT name ((title?, first-name, last-name) | (surname, mothers-name))>
<name> елемент може містити одну з двох послідовностей: за бажанням <title>, за яким <first-name> і <last-name> **або** <surname> і <mothers-name>.

Визначення атрибутів

Для елементів XML документа можна визначити атрибути XML. Використовуючи DTD, також можна:

- визначити, які атрибути потрібні
- визначити значення за замовчуванням для атрибутів
- перерахувати всі дійсні значення для даного атрибута

Декларація атрибута має такий синтаксис:

<!ATTLIST `element-name` attribute-name attribute-type attribute-value>

Тип атрибута може бути одним із наступних:

CDATA	Значення - символьні дані
(en1|en2|..)	Значення повинно бути одне зі списку
ID	Значення - це унікальний ID
IDREF	Значення - ID іншого елемента
IDREFS	Значення - це список ID інших елементів
NMTOKEN	Значення - допустиме ім'я XML
NMTOKENS	Значення - це список допустимих імен XML
ENTITY	Значення - це сутність
ENTITIES	Значення - це список сутностей
NOTATION	Значення - це назва позначення
xml:	Значення - це заздалегідь задане значення xml

Атрибут-значення може бути одним з наступних:

Значення	Значення атрибута за замовчуванням
#REQUIRED	Атрибут обов'язковий
#IMPLIED	Атрибут необов'язковий
#FIXED value	Значення атрибута фіксоване

Припустимо, потрібно змінити DTD, щоб зробити state атрибут <city> елемента. Ось як це зробити:

```
1   <!ELEMENT city (#PCDATA)>
2   <!ATTLIST city state CDATA #REQUIRED>
```

Це визначає <city> елемент, як і раніше, але далі використовується ATTLIST декларація для переліку атрибутів елемента. Ім'я city всередині списку атрибутів повідомляє аналізатору, що ці атрибути визначені для <city> елемента. Назва state- це ім'я атрибута, ключові слова CDATA та #REQUIRED скажуть аналізатору, що state атрибут містить текст і він необхідний (якщо це необов'язково - CDATA #IMPLIED, якщо фіксоване значення - CDATA #FIXED "value").

Щоб визначити кілька атрибутів для елемента, застосовують ATTLIST так:

```
1       <!ELEMENT city (#PCDATA)>
2       <!ATTLIST city state CDATA #REQUIRED
3                      postal-code CDATA #REQUIRED>
```

Цей приклад визначає state і postal-code атрибути <city> елемента.

Нарешті, DTD дозволяє визначити значення за замовчуванням для атрибутів та перерахувати всі дійсні значення для атрибута:

```
1       <!ELEMENT city (#PCDATA)>
2       <!ATTLIST city state CDATA (AZ|CA|NV|OR|UT|WA) "CA">
```

Приклад вказує, що підтримуються лише адреси штатів Арізона (AZ), Каліфорнія (CA), Невада (NV), Орегон (OR), Юта (UT) та Вашингтон (WA), і що за замовчуванням штат Каліфорнія. Таким чином можна зробити дуже обмежену форму перевірки даних.

У XML немає правил щодо того, коли використовувати атрибути та коли використовувати дочірні елементи. Дані можуть зберігатися в дочірніх елементах або в атрибутах.

Наприклад:

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>

<person>
  < gender >female</ gender >
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

У першому прикладі стать є атрибутом. В останньому – це дочірній елемент елемента person.

Деякі проблеми з атрибутами:

- атрибути не можуть містити декілька значень (дочірні елементи можуть)
- атрибути не легко розширюються (для майбутніх змін)
- атрибути не можуть описувати структури (дочірні елементи можуть)
- атрибути утруднюють зміни програмного коду
- значення атрибутів непросто перевірити на DTD

Якщо ви використовуєте атрибути як контейнери для даних, ви отримуєте документи, які важко читати та підтримувати. Краще використовувати **елементи** для опису даних. Атрибути використовуйте лише для надання інформації, яка не стосується даних. Тобто, метадані (дані про дані) повинні зберігатися як атрибути, а самі дані повинні зберігатися як елементи.

```
<messages>
  <note id="p501">
    <to>Tove</to>
  </note>
  <note id="p502">
    <to>Jani</to>
    <from>Tove</from>
  </note>
</messages>
```

id у цих прикладах - це лише лічильник або унікальний id для ідентифікації різних приміток у файлі XML, а не частина даних примітки.

Сутності

Сутності використовуються для визначення ярликів якихось характеристик. Вони можуть бути внутрішніми або зовнішніми. Формат оголошення внутрішньої сутності

```
<!ENTITY entity-name "entity-value">
```

Наприклад,

```
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">
```

В XML це відображатиметься як

```
<author>&writer;&copyright;</author>
```

Формат оголошення зовнішньої сутності

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

Наприклад,

```
<!ENTITY writer SYSTEM "https://www.w3schools.com/entities.dtd">
<!ENTITY copyright SYSTEM "https://www.w3schools.com/entities.dtd">
```

В XML це відображатиметься як

```
<author>&writer;&copyright;</author>
```

Сутність складається з трьох частин:
амперсанд (&), ім’я сутності та крапка з комою (;)

Сутності розгортаються, коли документ аналізується аналізатором XML.

Наступні сутності попередньо визначені в XML:

Сутність	Символ
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

Деякі зауваження

Визначаючи схему документа слід пам'ятати про гнучкість щодо схеми документів XML. Розглянемо назву зразка та тип документа. На початку було чітко вказано, що це маються на увазі поштові адреси США. Якщо потрібно DTD або схему, яка визначає правила для інших типів адрес, то доведеться додати до неї набагато більшу складність. Так, <state> елемент може мати сенс в Австралії, але не у Великобританії. Канадська адреса може оброблятися зразком DTD у визначеннях типу документа, але додавання <province> елемента буде кращою ідеєю.

Отже, перед тим як визначити структуру документа XML, слід так само заздалегідь продумати свій DTD або схему, як і коли б ви розробляли схему бази даних або структуру даних вашої програми. Чим більше майбутніх вимог можна передбачити, тим легше і дешевше буде їх виконати пізніше.

Також, на даний час в PostgreSQL тип xml не перевіряє значення що вводяться за схемою DTD, навіть якщо в них присутні посилання на DTD. Немає і вбудованої підтримки інших різновидів схем, наприклад XML Schema.

Матеріали для ознайомлення з теоретичними поняттями теми «DTD схема XML документа»:

1. Beginning XML, 5-edition © 2012 Joe Fawcett, Liam R.E. Quin, Danny Ayers
2. XML DTD W3C.
3. XML DTD Tutorial W3C.

Перелік понять, з якими необхідно ознайомитись для виконання завдання лабораторної роботи:

1. Основні поняття XML (*коректність* та *валідність)*
2. Способи визначення структури та елементів в XML документі
3. Визначення структури, елементів та атрибутів у DTD схемі
4. Символи DTD схеми
5. Визначення атрибутів DTD схеми
   5.1 Типи атрибутів
   5.2 Значення атрибутів
6. Сутності

Хід роботи

1. Опрацювати теоретичний матеріал.
2. Власноруч створити DTD схему для XML документа, який ви створили в попередній ЛР-3.
3. Використати on-line валідатор для перевірки валідності створеного XML документа разом з DTD схемою.
4. Оформити звіт про виконання лабораторної роботи, який має містити:
   - титульну сторінку;
   - тему, мету та завдання лабораторної роботи;
   - опис створення DTD схеми XML документа;
   - навести скріни екрану з кодом DTD схеми та XML документа і результатом перевірки його валідатором (наприклад TRUUGO – за деякий час може вимагати *безкоштовну* реєстрацію).
5. Завантажити в канал “БД. Лабораторна робота” своєї команди в Teams.

Приклад валідації XML документа з DTD