УДК 004.413+004.415.25

Ю.В. Ришковець
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

МЕТОД ІНТЕГРАЦІЇ ДАНИХ З ІНФОРМАЦІЙНИХ СИСТЕМ МУЗЕЙНИХ ПРЕДМЕТІВ

Запропоновано метод інтеграції даних з інформаційних систем, реалізованих на певних системах керування базами даних, який дає змогу за короткий час значно збільшити обсяги даних об'єднаної системи музейних предметів для забезпечення повноти інформації про музейні предмети.

Ключові слова: інтеграція, СКБД, XSLT, XML.

In the paper method of integrating data from museum information systems, implemented on certain database management system are proposed that allows for short time to increase the volume of data combined system of museum objects to provide complete information about museum objects.

Key words: integration, DBMS, XSLT, XML.

Вступ

Збереження культурної спадщини та ознайомлення з нею широкого кола людей – одне з головних завдань музейних закладів [1]. Інтеграція музеїв в мережу Інтернет є обов'язковою вимогою подальшого розвитку музейної сфери. Наслідком постійного зростання кількості користувачів Інтернету є зростання ролі Веб-галерей, які за своєю суттю є відображенням звичайних музеїв та виставок в мережі Інтернет. Веб-галереї дають змогу переглядати експозиції не виходячи з дому, і надають можливість реальним музеям створювати повні експозиції, які не обмежуються ні часом перегляду, ні площею приміщення музею. Відвідуваність та авторитетність Веб-галереї значною мірою залежать від якісного інформаційного наповнення Веб-галереї. Якісна інформація передусім консолідується в музейних установах у вигляді певних інформаційних систем музейних предметів (ІСМП) або у вигляді Веб-галерей мережі Інтернет.

У цій роботі розглядаються проблеми інтеграції даних з інформаційних систем музейних предметів, реалізованих на певних системах керування базами даних (СКБД).

Зв'язок висвітленої проблеми із науковими завданнями

Мета роботи – проаналізувати та розробити методи консолідації даних з інформаційних систем музейних предметів для забезпечення повноти інформації про музейні чи інші предмети, зв'язки між ними, що уможливить створювати цікавіші та інформативніші Веб-виставки.

Наукова новизна роботи полягає у побудові методу інтеграції даних з інформаційних систем музейних предметів, реалізованих на певних СКБД.

Практична цінність роботи полягає у розробленні методів консолідації даних з певних СКБД інформаційних систем музейних предметів.

Аналіз останніх досліджень

Значне збільшення обсягу даних в інформаційних системах музейних предметів можливе лише за рахунок інтеграції даних з інших інформаційних музейних систем, до яких належать Веб-галереї.

Надалі поняття “інформаційна музейна система”, “інформаційна система музейних предметів” та “Веб-галерея” вважатимемо синонімами.

Необхідність інтеграції даних виникає через неоднорідність програмного середовища, розподілений характер організації, підвищені вимоги до безпеки даних, необхідність наявності багаторівневих довідників метаданих, потребу в ефективному зберіганні й опрацюванні дуже великих обсягів інформації [2].

Під інтеграцією даних розумітимемо процес об’єднання даних з різноплатформенних джерел даних, попередньо перетворивши їх з різних форматів у формат цільового місця зберігання даних.

Сьогодні найпоширенішою технологією інтеграції неоднорідних даних є XML [3, 4].

Мова XML (Extensible Markup Language – розширювана мова розмітки) надає гнучкі та ефективні можливості описанню даних розміткою за допомогою тегів. Однак вона не забезпечує можливостей для пошуку визначених частин структурованих даних всередині документа.

Будь-яка програма, що підтримує XML, може читати та опрацьовувати будь-які XML-дані, незалежно від операційної системи та апаратного забезпечення комп'ютера.

Звичайна XML-система складається з файлів трьох типів:

§ XML-дані – це корисна інформація, а теги XML описують значення та структуру цих даних;

§ XML-схеми визначають правила стосовно того, що може, а що не може міститися у файлах даних. Наприклад, схема повинна забороняти введення слів тексту у поле дати;

§ XML-перетворення роблять дані придатними для використання у різних програмах або файлах. Наприклад, одне перетворення може вставляти дані продажу в електронну таблицю, а інше – вставляти ці самі дані в XML-документ.

Ключовою компонентою у системі XML є дані. Файли даних XML містять дані та набір кодів (тегів), які описують значення даних. Можна створити будь-які теги, необхідні для ваших даних. Саме з цієї причини формат XML є таким корисним і гнучким у використанні.

Другим основним компонентом системи XML є схема. Слово “схема” може бути незрозумілим, але це лише набір правил, у яких говориться, що може бути і чого не може бути в різних частинах файла даних XML. Схема використовується для перевірки даних. Наприклад, вона допомагає перевірити, чи не введено текст у поля, призначені для введення чисел.

На відміну від цього, правила схеми визначають, що може, а що не може міститися у певній структурі даних.

Мова XSL використовується для перетворення неоднорідних та складних даних, розміщених в XML-документах. XSL має дві основні функції: форматування XML-документів та перетворення їх в інші формати даних. До складу XSL входить мова перетворення розширених таблиць стилів (Extensible Stylesheet Language Transformations – XSLT), яка використовується для перетворення XML-документів з одного формату в інший. Перетворення містить сукупність правил для перетворення даних, що описуються одним набором тегів, на дані, що описуються іншим набором тегів. XSLT використовує XPath для зіставлення вузлів перетворюваного та результуючого документів [3–5].

XML Path Language (XPath) – це мова виразів для адресації частин XML-документа, або для обчислення величин (рядкових, числових або булевих) на основі вмісту XML-документа. XPath використовується для організації доступу до елементів документа XML з файла стилів XSLT. Він надає зручний та ефективний синтаксис для визначення місцезнаходження визначених частин XML-документа. У цій мові документ розглядається як дерево, в якому кожна частина документа подається у вигляді вузла.

На рис. 1 показано процес консолідації даних з різних музейних систем. Кожна музейна система переважно за допомогою СКБД здійснює експорт даних у XML-формат. Структури отриманих в такий спосіб XML-документів відрізняються, тому що, по-перше, в музейних системах використовуються різні схеми баз даних, а по-друге, різні засоби експорту даних. З цього випливає, що для інтеграції цих даних в об’єднану музейну систему необхідно привести структуру цих XML-документів до певного вигляду. За такі перетворення відповідає XSLT-процесор, в якому

визначаються необхідні XSLT-правила. Залежно від структури вхідного XML-документа застосовується відповідне XSLT-правило. Після опрацювання XSLT-процесором XML-документа отримаємо XML-документ потрібного вигляду, який імпортується у кінцеву музейну систему.

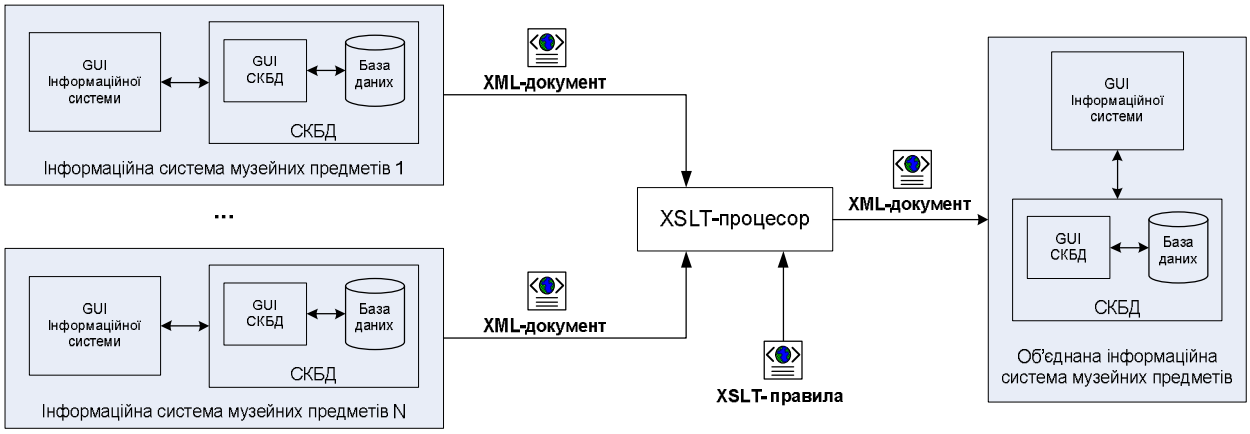

Рис. 1. Загальна схема інтеграції даних інформаційних систем музейних предметів

Перевагою такого методу консолідації даних з інформаційних систем обліку музейних предметів є можливість адаптації XSLT-правил перетворень XML-документів безпосередньо для кожної системи без необхідності розробляти спеціальні алгоритми імпорту даних в об'єднану музейну систему.

Виділення невирішених частин проблеми

Переважна більшість ІСМП для збереження та опрацювання введеної інформації використовує певну СКБД, побудовану на деякій моделі даних – реляційній, ієрархічній, об'єктно-орієнтованій, часовій чи іншій. Для перенесення інформації між СКБД, що ґрунтуються на однаковій моделі даних, наприклад, реляційній, достатньо використати звичайний SQL. У такому разі проблем не виникає, тому що ці СКБД є сумісні, тобто вони містять сумісні засоби зберігання та перенесення інформації з одного місця в інше без втрати цілісності. Крім того, кожна з таких систем має свої функціональні можливості і реалізована на певній програмній платформі. Наслідком цього є невідповідність схем баз даних і неоднорідність даних у таких системах, що стає серйозною проблемою за консолідації даних з цих систем.

Основний матеріал

Проведемо аналіз багатьох популярних СКБД (Microsoft Access, MySQL, PostgreSQL, Oracle та Microsoft SQL Server) на наявність стандартної функціональної можливості експортування даних бази даних у XML-формат.

Розглянемо детальніше можливості експортування кожної з наведених вище СКБД. Експортування в XML-формат здійснюватимемо на прикладі відношення ob_exhibit (id, Name, Collection_Id, Material, Technique, Height, Width, crtdate), яке містить інформацію про експонат. Після цього проведемо аналіз структури XML-документів кожної із розглянутих СКБД.

До складу СКБД Microsoft Access входять вбудовані засоби для здійснення експортування даних у XML-документи. Можливі такі види експорту [6]:

- § експорт даних у файл XML і за необхідності використання мови XSLT для перетворення даних в інший формат;
- § експорт схеми даних з використанням стандарту схем XML (XSD);
- § експорт у файл XML даних із форм та звітів.

Експериментально встановлено, що в Microsoft Access відсутня стандартна можливість експортування в XML-документ одразу усієї бази даних, що є значним недоліком цієї СКБД.

Приклад 1. Розглянемо XML-документ, що описує структуру таблиці ob_exhibit в MS Access:

```
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
generated="2010-04-12T17:12:19">
     <ob_exhibit>
          <id>6289</id>
          <Name>Богоматір(ікона)</Name>
          <Collection_Id>Музика</Collection_Id>
          <Material>фанера</Material>
          <Technique>темпера</Technique>
          <Height>84</Height>
          <Width>62</Width>
          <crtdate>2006-02-23T13:23:55</crtdate>
     </ob_exhibit>
...
</dataroot>
```

Проаналізувавши отриманий за допомогою експорту певного відношення XML-документ, побудуємо його DTD-схему:

```
<!ELEMENT dataroot (table+)>
<!ATTLIST dataroot
          xmlns:od CDATA #REQUIRED
          generated CDATA #REQUIRED>
<!ELEMENT table (column+)>
          <!ELEMENT column (#PCDATA)>
```

Тут головним тегом є <dataroot>...</dataroot>, він містить атрибути з інформацією про XML-схему (xmlns: od), що використовується, та час створення XML-документа (generated). Тегами <table>...</table> позначається назва відношення, а тегами <column>...</column> – назви атрибутів цього відношення. У XML-документі тег <table>...</table> зустрічається стільки разів, скільки є кортежів у відповідному відношенні.

Розглянемо СКБД MySQL, вона містить засоби експортування у формат XML як цілої бази даних, так і окремих її відношень.

Різниця синтаксису XML-документів СКБД MySQL та MS Access не істотна і полягає лише в описанні головного тегу, а уся інша структура XML-документа ідентична. Головний тег позначається за допомогою <database>...</database> і не містить атрибутів.

Приклад 2. Розглянемо XML-документ, що описує структуру таблиці ob_exhibit в MySQL:

```
<museum>
     <ob_exhibit>
          <id>6289</id>
          <Name>Богоматір(ікона)</Name>
          <Collection_Id>Музика</Collection_Id>
          <Material>фанера</Material>
          <Technique>темпера</Technique>
          <Height>84</Height>
          <Width>62</Width>
          <crtdate>2006-02-23T13:23:55</crtdate>
     </ob_exhibit>
     ...
</museum>
```

Після аналізу поданого у прикладі 2 XML-документа побудуємо його DTD-схему:

```
<!ELEMENT database (table+)>
<!ELEMENT table (column+)>
          <!ELEMENT column (#PCDATA)>
```

СКБД PostgreSQL, крім експортування у формат XML цілої бази даних або окремих її відношень, здійснює експортування в XML-документ схеми бази даних.

Починаючи з PostgreSQL 8, з'явились функції для роботи з XML. Включаючи функції xmlelement, xmlforest та xmlroot до запитів вибору реляційних даних, можна формувати XML-документи з різною структурою.

Приклад 3. Розглянемо XML-документ, що описує структуру таблиці ob_exhibit в PostgreSQL:

```
<database>
   <name>museum</name>
   <table>
       <name>ob_exhibit</name>
       <declaration>
           <field>
              <name>id</name>
              <type>Текстовий</type>
              <description>Код</description>
           </field>
           ...
           <index>
              <name>id_primarykey</name>
              <field>
                  <name>id</name>
                  <sorting>ascending</sorting>
              </field>
           </index>
       </declaration>
       <initialization>
           <insert>
              <field>
                  <name>id</name>
                  <value>6289</value>
              </field>
              <field>
                  <name>Name</name>
                  <value>Богоматір(ікона)</value>
              </field>
              <field>
                  <name>Collection</name>
                  <value>Музика</value>
              </field>
              <field>
                  <name>Material</name>
                  <value>фанера</value>
              </field>
          </insert>
          ...
       </initialization>
   </table>
   <table>
   ...
   </table>
</database>
```

DTD-схему розглянутого у прикладі 3 XML-документа подано нижче:

```
<!ELEMENT database (name, table+)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT table (name, declaration, initialization)>
      <!ELEMENT declaration (field+, index+)>
          <!ELEMENT field (name, type, description)+>
               <!ELEMENT type (#PCDATA)>
               <!ELEMENT description (#PCDATA)>
          <!ELEMENT index (name, field)>
               <!ELEMENT field (name, sorting)>
                    <!ELEMENT sorting (#PCDATA)>
      <!ELEMENT initialization (insert+)>
          <!ELEMENT insert (field+)>
               <!ELEMENT field (name, value)>
                    <!ELEMENT value (#PCDATA)>
```

Тут тег <database>...</database> містить два підтеги <name>...</name> та <table>...</table>, що відображають інформацію про базу даних. Відповідно перший тег відображає ім'я бази даних, а другий – опис відношення, до якого входить опис назви відношення (тег <name>...</name>), опис його схеми – атрибутів, індексів (тег <declaration>...</declaration>) та опис даних, що містяться у цьому відношенні (тег <initialization>...</initialization>). Кожний кортеж відношення позначається тегом <insert>...</insert>, а кожний атрибут – тегом <field>...</field>. Тег атрибуту має свої два підтеги – назву атрибуту (тег <name>...</name>) та його значення у відповідному кортежі відношення (тег <value>...</value>). Хоч у цьому XML-документі наводиться опис схеми відношення, але для нашої подальшої інтеграції він нам не буде потрібний.

Включно до восьмої версії у своєму функціоналі СКБД Oracle не мала стандартних засобів експортування даних у XML-формат. Для експортування даних у XML-формат необхідно було розширити функціонал Oracle, встановивши певне програмне забезпечення, наприклад, XML Developer's Kit (XDK) [5]. Починаючи з дев'ятої версії, до складу СКБД Oracle входить набір XML-функцій, використовуючи які, в запитах до реляційних даних можна отримати дані у XML-форматі [5]. Використовуючи XML-функції XMLELEMENT та XMLFOREST, в Oracle можна сформувати XML-документ. Враховуючи схожість структур XML-документів СКБД Oracle та MySQL, вважатимемо, що все, що стосується MySQL, стосується і Oracle. Тоді розроблені нижче засоби можна використати для інтеграції даних і із СКБД Oracle.

Microsoft SQL Server 2000 підтримує використання XML через Microsoft SQLXML, що дає змогу конвертувати реляційні дані у формат XML. Починаючи з версії Microsoft SQL Server 2000, для генерації XML з реляційних даних і, навпаки, використовуються конструкції FOR XML і OPENXML. При експорті в XML можливі два види структур XML-файлів, перша з них відповідає структурі XML-файла СКБД MySQL, тому цей варіант окремо не розглядатимемо. Враховуючи це, розглянемо лише другу структуру XML-файла.

Приклад 4. Розглянемо XML-документ, що описує структуру таблиці ob_exhibit в MS SQL Server 2005:

```
<dbo.ob_exhibit id="0001" Name="Богоматір(ікона)"
Collection_Id="Музика" Material="фанера" Technique="темпера"
Height="84" Width="62" crtdate="2006-02-23T13:23:55"/>
```

Аналізуючи розглянутий у прикладі 4 XML-документ, побудуємо його DTD-схему:

```
<!ELEMENT dbo.table EMPTY>
<!ATTLIST dbo.table
column CDATA #REQUIRED
... >
```

Тут table відображає назву відношення бази даних, що експортується. Атрибутами елемента dbo.table є назви атрибутів відношення (column) з відповідними значеннями. За допомогою тегу dbo.table відображається лише один кортеж відношення.

Проаналізуємо формат XML-документа, який ми використовуватимемо у нашій системі.

Приклад 5. Розглянемо XML-документ, що описує структуру таблиці ob_exhibit в об'єднаній інформаційній системі музейних предметів:

```
<web-gallery>
     <ob_exhibit>
          <row>
             <id>6289</id>
             <Name>Богоматір(ікона)</Name>
             <Collection_Id>Музика</Collection_Id>
             <Material>фанера</Material>
             <Technique>темпера</Technique>
             <Height>84</Height>
             <Width>62</Width>
             <crtdate>2006-02-23T13:23:55</crtdate>
          </row>
          ...
     </ob_exhibit>
     ...
</web-gallery>
```

Відповідно DTD-схема розглянутого у прикладі 5 XML-документа така:

```
<!ELEMENT database (table+)>
<!ELEMENT table (row+)>
          <!ELEMENT row (column+)>
           <!ELEMENT column (#PCDATA)>
```

Тут як database вказується назва бази даних, table відображає назву відношення. Кожний кортеж відношення позначається тегом <row>...</row> і містить елементи column, які відображають назви атрибутів з відповідними значеннями.

Аналіз XML-документів, отриманих за допомогою експорту даних з певних СКБД, дав можливість побудувати відповідні DTD-схеми цих документів, а аналіз DTD-схем – розробити XSLT-перетворення, які здійснюють структурне перетворення певного XML-документа у структуру XML-документа об'єднаної ІСМП.

Процес інтеграції даних є дуже складним процесом, тому що передбачає узгодження форматів даних та структур XML-документів.

Тому розглянемо такі випадки інтеграції даних:

1) схеми баз даних і структури XML-документів джерела та приймача даних збігаються;

2) схеми баз даних збігаються, а структури XML-документів джерела та приймача даних не збігаються;

3) ні схеми баз даних, ні структури XML-документів джерела та приймача даних не збігаються.

Перший випадок не розглядатимемо, тому що тут інтеграція даних зводиться до звичайного імпорту XML-документа за допомогою засобів СКБД без внесення у нього будь-яких змін.

Другий випадок складніший, тому що схеми баз даних однакові, а за інтеграції даних враховуються тільки відмінності у структурах XML-документів джерела та приймача даних.

У наступному прикладі подамо розроблені нами XSLT-перетворення для певних СКБД.

Приклад 6. Розглянемо XSLT-перетворення, які дають змогу привести XML-документи неоднорідних баз даних (MS Access, MySQL, PostgreSQL, Microsoft SQL Server 2005) до формату інтегрованої системи музейних даних (приклад 5). Усі перетворення будуть наведені на прикладі таблиці ob_exhibit.

Кожне XSLT-перетворення починається з правила опрацювання кореневого елемента XML-документа:

```
<xsl:template match="/">
```

```
     <web-gallery>
          <xsl:apply-templates/>
     </web-gallery>
</xsl:template>
```

Тут елемент template зіставляє визначений вузол XML-документа, використовуючи вираз XPath, заданий в атрибуті match із заданим деревом XML-документа. У цьому випадку, коли в заданому дереві зустрічається кореневий елемент /, то в результуюче дерево заноситься вміст елемента template, а саме: створюється кореневий елемент web-gallery, в який заноситься вміст елемента apply-templates. Елемент apply-templates використовується для застосування шаблону XSLT-документа до визначених вузлів XML-документа.

1) XSLT-перетворення для MS Access та MySQL:

```
<xsl:template match="/">
     <web-gallery>
          -- Застосування шаблону XSLT-документа
          <xsl:apply-templates/>
     </web-gallery>
</xsl:template>
-- Шаблон, що застосовується до всіх дочірніх вузлів XML-
документа
<xsl:template match="//element()">
     <xsl:element name=“ob_exhibit”>
          <xsl:for-each select=“ob_exhibit”>
               <row><xsl:copy-of select="*"/></row>
          </xsl:for-each>
     </xsl:element>
</xsl:template>
```

Тут визначено шаблон опрацювання усіх дочірніх тегів <xsl: template match="//element()"> у заданому XML-документі.

Рядок <xsl: element name=“ob_exhibit”> використовується для створення елемента з ім’ям, визначеним в атрибуті name, тобто ob_exhibit.

Елемент XSLT for-each застосовує вміст свого елемента до кожного вузла, що збігаються зі значенням атрибута select. У нашому випадку зіставлення здійснюється з усіма елементами ob_exhibit. За збігу елементів створюється тег <row>, в який за допомогою шаблону <xsl: copy-of select="*"/> із заданого XML-файла копіюються усі дочірні вузли разом зі своїми атрибутами.

2) XSLT-перетворення для PostgreSQL:

```
<xsl:template match="/">
     <web-gallery>
          -- Застосування шаблону XSLT-документа
          <xsl:apply-templates/>
     </web-gallery>
</xsl:template>
-- Шаблон, що застосовується до всіх дочірніх вузлів XML-
документа
<xsl:template match="//element()">
     <xsl:for-each select=“table”>
          <xsl:element name="{child::name}">
               <xsl:for-each select=“initialization/insert”>
                    <row>
                         <xsl:for-each select=“field”>
                              <xsl:element name="{child::name}">
                                   <xsl:value-of select=“value”/>
```

```
                    </xsl:element>
                </xsl:for-each>
                </row>
            </xsl:for-each>
        </xsl:element>
    </xsl:for-each>
</xsl:template>
```

Шаблон <xsl: template match="//element()"> опрацьовує усі дочірні теги у заданому XML-документі.

Елемент XSLT for-each застосовує вміст свого елемента до кожного вузла, що збігається зі значенням атрибута select. У нашому випадку зіставлення здійснюється з усіма елементами table. За збігу елементів за допомогою шаблону <xsl: element name="{child:: name}"> в результуючому дереві створюється тег <name>, дочірній тегу <table>, значення name якого зчитується із однойменного тегу заданого XML-файла.

Наступний елемент XSLT for-each застосовує вміст свого елемента до кожного вузла <insert>, що входить до тегу <initialization>. Якщо елементи збіглись, то в результуючому XML-документі для кожного тегу <insert> створюється тег <row>.

Ще один елемент XSLT for-each застосовує вміст свого елемента до кожного вузла <field>, що входить до тегу <insert>. Наступний рядок <xsl: element name="{child:: name}"> використовується для створення елемента з ім'ям, визначеним в атрибуті name за допомогою XPath-виразу {child:: name}, що відповідає дочірньому вузлу <field>. За допомогою рядка <xsl: value-of select="value"/> із заданого XML-документа зчитується значення відповідного тегу і записується у відповідний елемент, створений перед цим в результуючому файлі.

3) XSLT-перетворення для MS SQL Server 2005:

```
<xsl:template match="/">
-- Формування результуючого XML-документа
<web-gallery>
        <xsl:element name="ob_exhibit">
            <row>
                <xsl:element name="id">
                    <xsl:value-of select="@id"/>
                </xsl:element>
                <xsl:element name="Name">
                    <xsl:value-of select="@Name"/>
                </xsl:element>
                <xsl:element name="Collection">
                    <xsl:value-of select="@Collection"/>
                </xsl:element>
                <xsl:element name="Material">
                    <xsl:value-of select="@Material"/>
                </xsl:element>
                <xsl:element name="Technique">
                    <xsl:value-of select="@Technique"/>
                </xsl:element>
                <xsl:element name="Height">
                    <xsl:value-of select="@Height"/>
                </xsl:element>
                <xsl:element name="Width">
                    <xsl:value-of select="@Width"/>
                </xsl:element>
                <xsl:element name="crtdate">
```

```
                        <xsl:value-of select="@crtdate"/>
                    </xsl:element>
                </row>
            </xsl:element>
        </web-gallery>
</xsl:template>
```

Тут спочатку створюється кореневий елемент <web-gallery>, після цього створюється дочірній йому елемент <ob_exhibit>, що позначає відношення. Потім створюється вкладений тег <row>, у якому створюються дочірні елементи з назвою. вказаною в атрибуті name, і значенням, що зчитується за допомогою XPath-виразу з відповідного атрибута. Причому атрибути позначаються символом @.

За інтеграції можливий ще третій випадок, коли ні схеми баз даних, ні структури XML-документів джерела та приймача даних не збігаються. Він є найскладнішим, бо для того, щоб привести XML-документи джерел даних до структури XML-документа інтегрованої системи музейних даних, потрібно, крім відмінностей у схемах баз даних, врахувати ще й відмінності у структурах XML-документів джерела та приймача даних. Ця проблема вимагає додаткового дослідження методу інтеграції музейних даних.

Використання розроблених XSLT-перетворень дає змогу швидко та якісно консолідувати дані з різних інформаційних систем обліку музейних предметів та Веб-галерей шляхом інтеграції цих даних за допомогою технології XML.

Висновки

Сьогодні завдання інтеграції та консолідації даних в інформаційних системах музейних предметів, зокрема у Веб-галереях, є доволі актуальним завданням, оскільки його виконання дасть змогу реалізувати централізоване сховище даних музейних предметів, що забезпечить оперативність доступу до музейних даних та забезпечить їх доступність і повноту. Для розв'язання цієї задачі у роботі запропоновано використати метод інтеграції даних з певних СКБД інформаційних систем музейних предметів, що уможливило розробити засоби інтеграції даних з цих систем. В основу використаних методу та засобів інтеграції покладено технологію XML, що на практиці дає змогу консолідувати інформацію про музейні предмети в єдину інформаційну систему музейних предметів.

1. Основи музеєзнавства, маркетингу та рекламно-інформаційної діяльності музеїв: посібник / П. Горішевський, М. Дейнега, М. Ковалів, В. Мельник, Н. Рега, С. Оришко, О. Соколова / за ред. В. Великочого, Н. Гасюк. – Івано-Франківськ : Плай, 2005. – 64 с. 2. Шаховська Н. Б. Сховища та простори даних : монографія / Н. Б. Шаховська, В. В. Пасічник. – Львів: Видавництво НУ "Львівська політехніка", 2009. – 244 с. 3. Дейтел Х. М. Как программировать на XML / Х. М. Дейтел, П. Дж. Дейтел, Т. Р. Нието, Т. М. Лин, П. Садху; пер. с англ. – 2-е изд. – М.: ООО "Бином-Пресс", 2008. – 944 с. 4. Кэй М. XSLT. Справочник программиста / М. Кэй. – 2-е изд. – Спб.: Символ-Плюс, 2002. – 1016 с. 5. World Wide Web Consortium. XSL Transformations (XSLT). Version 1.0 – W3C Recommendation, 16 November 1999. [Електронний ресурс]. – Режим доступу : http://www.w3.org/TR/xslt. – 23.02.2010 р. – Назва з титул. екрана. 6. Вивчення основ XML і розгляд використання XML у деяких програмах Microsoft Office System. [Електронний ресурс] – Режим доступу: http://office.microsoft.com/training/training.aspx? AssetID=RC011304651058. – 28.02.2010 р. – Назва з титул. екрана. 7. MySQL 5.1 Reference Manual. Mysql Options. [Електронний ресурс]. – Режим доступу: http://dev.mysql.com/doc/refman/5.1/en/mysql-command-options.html# option_ mysql_ xml. – 15.09.2010 р. – Назва з титул. екрана. 8. PostgreSQL 8.3.14 Documentation. Chapter. 9. Functions and Operators. [Електронний ресурс] – Режим доступу: http://www.postgresql.org/docs/ 8.3/static/functions-xml.html. – 15.09.2010 р. – Назва з титул. екрана. 10. Чанг Б. Oracle 9i XML / Б. Чанг, М. Скардина, С. Киритцов; пер. с англ. А. Головко. – М.: Лори, 2003. – 476 с. 11. Каленик А.И. Использование новых возможностей Microsoft SQL Server 2005 / А. И. Каленик. – СПб.: Русская Редакция, 2006. – 334 с.