УДК 004.652.4

П.І. Жежнич
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

ГЕНЕРУВАННЯ SQL-СЦЕНАРІЇВ ЗА ДОПОМОГОЮ XSLT-ПЕРЕТВОРЕНЬ

© Жежнич П.І., 2010

Розроблено методи генерування крос-платформних відносно СКБД SQL-сценаріїв за допомогою XML-технологій.

Ключові слова: реляційні бази даних, СКБД, SQL, XML, XSLT.

This paper considers methods of SQL-scripts generation with XML-technologies that are crossplatform by DBMS.

Keywords: relational databases, DBMS, SQL, XML, XSLT.

Вступ

Тенденції розвитку сучасних комп'ютерних технологій у напрямку формування інформаційного суспільства змінюють акценти щодо застосування баз даних: вони стають важливими продуктами виробництва і споживання інформації, а не лише інструментами автоматизації різних видів діяльності людини. Це, своєю чергою, вимагає певних засобів, які б забезпечували можливість використання баз даних «незалежно» від засобів їхнього створення та формування, тобто від систем керування базами даних (СКБД).

На ринку програмного забезпечення доступні комерційні та вільновживані реляційні СКБД (наприклад, Oracle Database Server, Microsoft SQL Server, PostgreSQL, MySql тощо), що підтримують стандартну мову доступу до даних SQL [1]. Однак важливим недоліком цих систем є саме відхилення від стандарту SQL (тобто підтримка різних діалектів SQL) для досягнення високої продуктивності опрацювання даних. А це приводить до «залежності» баз даних (БД) від конкретної СКБД у випадках, коли продуктивність системи не є критичною.

Зв’язок висвітленої проблеми із науковими завданнями

Метою статті є побудова XML-засобів генерування SQL-сценаріїв, здатних налаштовуватися на задану СКБД.

Наукова новизна статті полягає у побудові методу генерування SQL-сценаріїв за допомогою XSLT-перетворень.

Практична цінність статті полягає у побудові XSLT-перетворень для генерування SQL-сценаріїв, здатних налаштовуватися на задану СКБД.

Аналіз останніх досліджень

XML (Extensible Markup Language) – мова розширюваної розмітки – це стандарт для створення мов розмітки, що описують структури даних [2]. Оскільки XML, зокрема, розглядається як структура даних в описі документа, то логічно його використовувати як засіб опису структурованих даних. Як формат даних XML має декілька переваг. Зокрема, він самоописуваний, що забезпечує легке його опрацювання за допомогою різних програмних засобів. Однак він має і недоліки, наприклад, XML є занадто громіздким в описі даних, а тому неефективним у швидкому доступі до даних через велику кількість текстових перетворень. З іншого боку, XML та XML-технології не можна розглядати як повноцінну СКБД. Хоча XML дає змогу реалізувати достатньо елементів звичайних баз даних (збереження у форматі XML-документів, схеми даних за допомогою DTD та XML Schema, мови запитів як XQuery і XPath, XQL, XML-QL, QUILT, SQL4X тощо, програмні інтерфейси як SAX, DOM, JDOM), він має недоліки, що не дають змоги вважати XML "справжньою" СКБД. Зокрема, XML не підтримує таких важливих елементів БД, як економічність

збереження, індекси (ефективний пошук даних), безпеку, транзакції та інтегрованість даних, багатокористувацький доступ, тригери, перегляди тощо. Отже, XML-документи можна використовувати як БД, однак це доцільно робити лише в окремих випадках (у середовищах з невеликою кількістю даних або невисокими вимогами до продуктивності). Переважно це стосується задач перенесення (реплікації, збереження тощо) даних у розподілених в просторі і часі інформаційних системах (розподіленість у просторі означає наявність фізично розділених вузлів БД, розподіленість в часі означає потребу тимчасового збереження даних незалежно від поточного використання БД з метою їхнього опрацювання в майбутньому).

Існує декілька методів подання даних у XML-документах [3]: відображення на основі шаблону; відображення на основі моделі. В разі відображення на основі шаблону між документом і БД не існує наперед визначеної однозначної відповідності. Кожного разу за потреби перенесення даних у XML-документ створюється шаблон, у який вставляються спеціальні команди, що опрацьовуються програмами перенесення даних. У відображеннях на основі моделей дані у XML-документі подаються відповідно до наперед визначеної моделі даних. Цей підхід має значні переваги за простотою – документ містить лише структуровані дані і нічого зайвого. Гнучкість використання для різноманітних задач досягається інтеграцією із засобами перетворення XML-документів як XSLT [4]. Сучасні реляційні СКБД переважно підтримують об'єктно-реляційну модель відображення даних в XML. У цьому відображенні дані у XML-документі моделюються деревом об'єктів, специфічних для цього типу документів. Типи елементів з атрибутами та вміст елементів переважно моделюються класами. Класи відображаються у таблиці (відношення), скалярні властивості – в колонки (атрибути).

Наукові дослідження у напрямі подання реляційних БД за допомогою XML-документів переважно зводяться лише до відображення даних (наприклад, [5]), тоді як методи опису схем БД, обмежень цілісності та запитів до баз даних фактично не розглядаються. З іншого боку, така мова, як XML Schema, є придатною для відображення схем реляційних баз даних [6], однак вона є недостатньою для відображення зв'язків між даними (наприклад, часово-залежні дані містять причинно-наслідкові зв'язки в часі [7]), оскільки за допомогою неї описують схеми XML-документів. Крім того, XML Schema орієнтована передусім на валідацію XML-документів, а не на відображення схеми БД загалом, а при перенесенні даних з БД у XML-формат така валідація не завжди є критичною, оскільки ті самі сучасні СКБД дають змогу генерувати «правильні» XML-документи. Оскільки метадані (тобто схема БД) в СКБД реляційного типу переважно зберігаються безпосередньо у базі даних у вигляді спеціального словника даних [8], то XML-формат можна використовувати для опису не тільки даних, але й метаданих. А це означає, що XML-технологія дає змогу створити крос-платформні відносно СКБД засоби перенесення БД (тобто даних і метаданих).

Постановка задачі

Задача перенесення реляційних БД в XML-формат і навпаки зводиться до таких дій: 1) відображення даних і метаданих за допомогою спеціальних XML-документів (рис. 1); 2) розроблення необхідних XSLT-перетворень, які б забезпечували генерування SQL-сценаріїв з XML-документів відповідно до діалекту SQL (рис. 2). Необхідно вибрати єдину XML-мову для опису даних і метаданих у спеціальних XML-документах (це пов'язано з тим, що різні СКБД можуть генерувати різне об'єктно-реляційне відображення даних, і для збереження крос-платформності ці відображення треба перетворити в єдиний XML-формат).

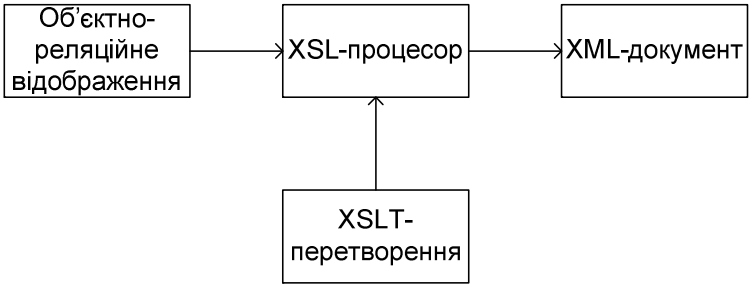

Рис. 1. Схема відображення даних і метаданих за допомогою спеціальних XML-документів

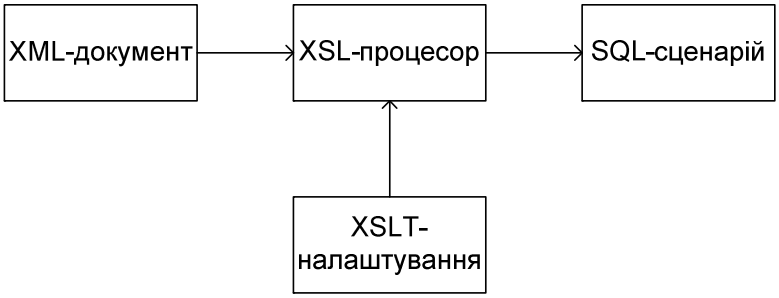

Рис. 2. Схема генерації SQL-сценаріїв зі спеціальних XML-документів

Основний матеріал

Виберемо за єдину XML-мову подання даних та метаданих XTDQML (eXtended Time-Depended Query Markup Language) – XML-мову розмітки часово-залежних запитів і даних [7]. Основні її структурні елементи подано на рис. 3–5.

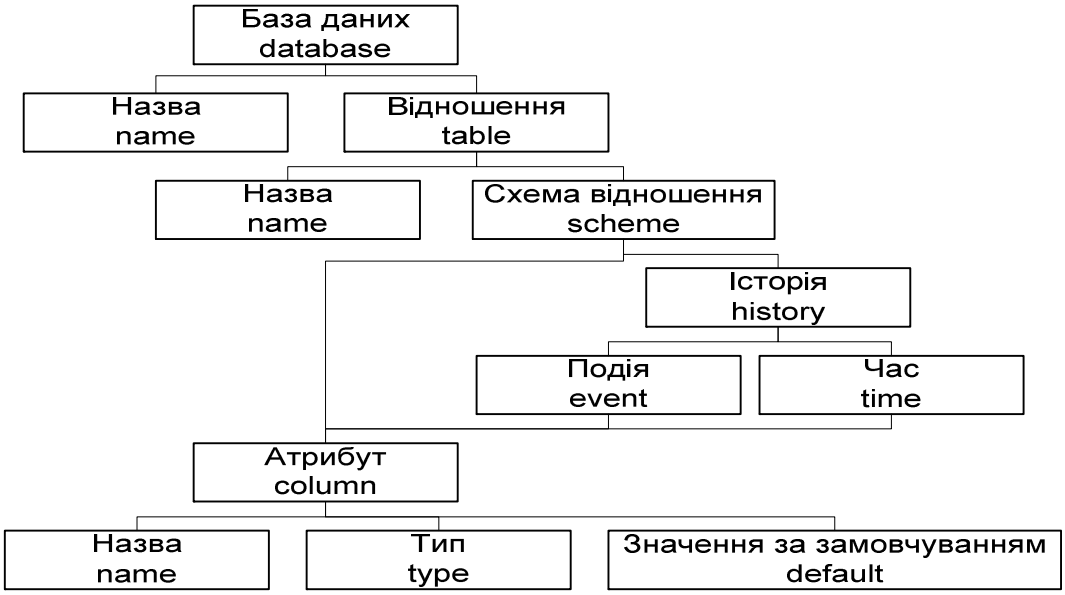

Рис. 3. Співвідношення тегів XTDQML для відображення схеми БД

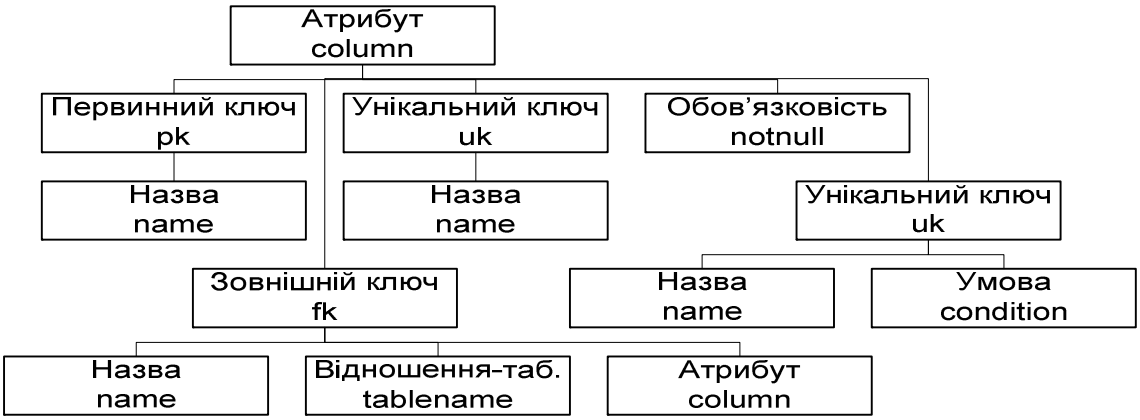

Рис. 4. Співвідношення тегів XTDQML для відображення обмежень цілісності БД

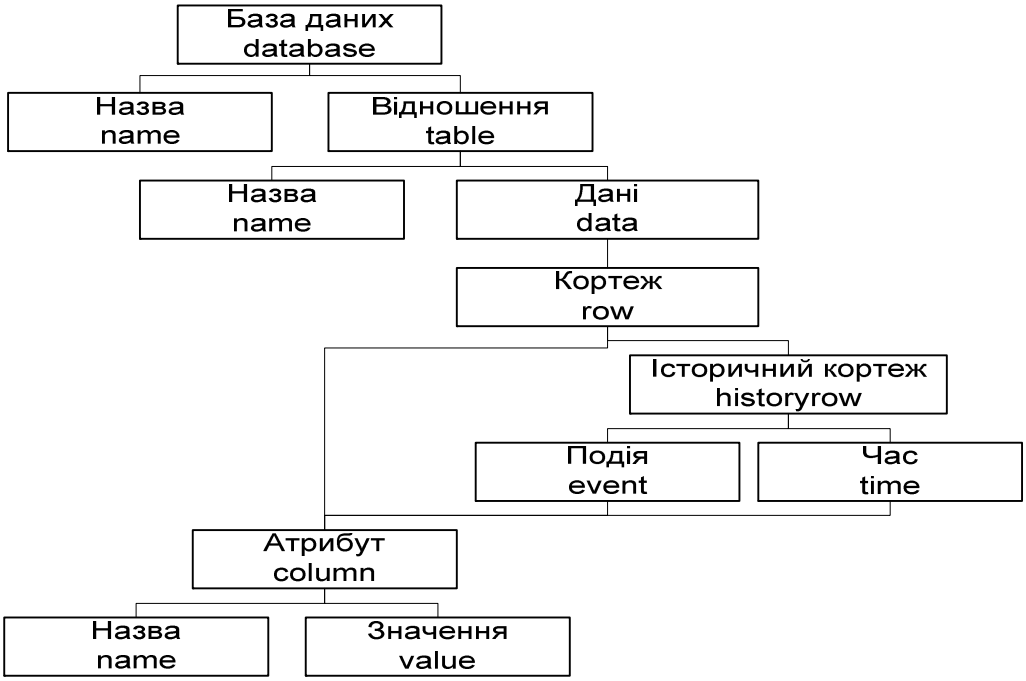

Рис. 5. Співвідношення тегів XTDQML для вмісту БД

Відображення даних і метаданих за допомогою XTDQML-документів полягає у встановленні відповідності між тегами об'єктно-реляційного відображення даних, отриманого засобами реляційної СКБД, і тегами XTDQML. Оскільки це відображення полягає у перетворенні одного XML-документа на інший і є типовою процедурою для XSLT, то ми не будемо розглядати його.

Генерація SQL-сценаріїв на основі XTDQML-документів полягає у встановленні відповідності між тегами та елементами відповідних SQL-запитів за допомогою спеціальних XSLT-перетворень, які встановлюють відповідність між тегами XTDQML і запитами (елементами запитів) мови SQL.

Для ілюстрації розглянемо лише метод генерації SQL-сценаріїв створення таблиць. Методи генерації інших SQL-сценаріїв за допомогою XTDQML розглядати не будемо, оскільки вони будуються за аналогією до генерації SQL-сценаріїв створення таблиць.

Для побудови SQL-сценаріїв створення таблиць розглянемо ієрархію XTDQML-тегів для опису схеми РБЧЗД, подану на рис. 3, де:

- *database* відповідає SQL-сценарію;
- *table* відповідає SQL-запиту на створення таблиці.

Побудова SQL-сценарію починається з опрацювання тега <database> як кореневого елемента XTDQML-документа:

```
<xsl:stylesheet version = '1.0'
 xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
<xsl:template match="database">
-- SQL-сценарій створення бази даних
-- <xsl:apply-templates select="name"/>
 <xsl:apply-templates select="table"/>
</xsl:template>
<xsl:template match="name">
 <xsl:value-of select="."/>
</xsl:template>
<!--table-->
</xsl:stylesheet>
```

Рядком "<!--table-->" позначено місце, куди записуються усі наступні перетворення, що стосуються таблиць бази даних.

Зазначимо, що, якщо для бази даних визначено назву <name>, то вона відобразиться як коментар SQL-сценарію за допомогою відповідного правила опрацювання тега <name>. Також це правило діятиме для усіх елементів бази даних (таблиці, атрибути тощо), які мають назви.

Кожен тег <table> послідовно опрацьовується у парі з тегом <scheme> за допомогою відповідних правил. Наявність окремого правила для елемента <scheme> забезпечує опрацювання лише схем таблиць, оскільки тег <column> може бути як у схемі, так і у вмісті таблиці:

```
<xsl:template match="table">
-- Створення таблиці <xsl:apply-templates select="name"/>
CREATE TABLE <xsl:apply-templates select="name"/> (
 <xsl:apply-templates select="scheme"/>
);
</xsl:template>
<xsl:template match="scheme">
 <xsl:for-each select="column">
  <xsl:apply-templates select="name"/>
  <xsl:text> </xsl:text>
  <xsl:apply-templates select="type"/>
  <xsl:text> </xsl:text>
  <xsl:apply-templates select="default"/>
  <xsl:text> </xsl:text>
  <xsl:apply-templates select="constraint"/>
  <xsl:if test="not (position()=last())">
   <xsl:text>,</xsl:text>
  </xsl:if>
 </xsl:for-each>
);
</xsl:template>
<!--type-->
<!--default-->
<!--constraint-->
```

Рядками "<!--type-->", "<!--default-->" та "<!--constraint-->" позначено місця, куди записуються наступні перетворення, що стосуються типу атрибута, значення за замовчуванням та обмежень цілісності.

Для опрацювання тега <column> існує єдине правило незалежно від того, до якої групи належить атрибут (фактично для SQL-запиту на створення таблиці елементи <history>, <event> і <time> є зайвими, а тому XSLT-перетворення ігнорують).

Тег <type> відображає тип даних атрибута, який допускає п'ять основних можливих значень: символьний з фіксованою довжиною (*char*), символьний з плаваючою довжиною (*varchar*), числовий (*numeric*), часовий (*date*), а також інтервал (*interval*):

```
<xsl:template match="type">
 <xsl:apply-templates select="char"/>
 <xsl:apply-templates select="varchar"/>
 <xsl:apply-templates select="numeric"/>
 <xsl:apply-templates select="date"/>
 <xsl:apply-templates select="interval"/>
</xsl:template>
<!--typedefs-->
```

Рядком "<!--typedefs-->" позначено місце, куди записуються перетворення щодо визначення типів даних. Типи даних у різних реляційних СКБД (і відповідних версіях SQL) реалізуються неоднаково. Наприклад, символьний тип з плаваючою довжиною може записуватися як VARCHAR, VARCHAR2 чи CHARACTER VARYING, а числовий тип – як NUMERIC чи NUMBER. Деякі СКБД підтримують усі варіанти запису, а деякі – лише один з них. Це приводить до наявності різних діалектів SQL, що відображається певними відмінностями у синтаксисі SQL-запитів, які необхідно враховувати у XSLT-перетвореннях. Наступне XSLT-перетворення відображає загальну схему визначення типу даних:

```
<xsl:template match="typename">
 SQLTYPE<xsl:text>(</xsl:text>
  <xsl:value-of select="."/><xsl:text>)</xsl:text>
</xsl:template>
```

Тут замість рядка *typename* записується назва XTDQML-тега, який відображає один з типів даних. Рядок *SQLTYPE* позначає відповідний тип даних за стандартом SQL.

Опрацьовуючи значення за замовчуванням, необхідно замінити тег *default* на його вміст:

```
<xsl:template match="default">
 DEFAULT <xsl:value-of select="."/>
</xsl:template>
```

Тег <constraint> відображає обмеження цілісності одного з п’яти видів: первинний ключ (*pk*), унікальний ключ (*uk*), обов’язковість значень атрибута (*notnull*), зовнішній ключ (*fk*), контроль значень атрибута (*check*):

```
<xsl:template match="constraint">
 <xsl:apply-templates select="name"/>
 <xsl:variable name="columnname"/>
  <xsl:apply-templates select="columnname"/>
 </xsl:variable>
 <xsl:variable name="columnlist"/>
  <xsl:apply-templates select="columnlist"/>
 </xsl:variable>
 <xsl:apply-templates select="pk"/>
 <xsl:apply-templates select="uk"/>
 <xsl:apply-templates select="notnull"/>
 <xsl:apply-templates select="fk"/>
 <xsl:apply-templates select="check"/>
</xsl:template>
<xsl:template match="name">
 CONSTRAINT <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="columnname">
 <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="columnlist">
 <xsl:for-each select="columnname">
  <xsl:value-of select="."/>
  <xsl:if test="not (position()=last())">
   <xsl:text>,</xsl:text>
  </xsl:if>
 </xsl:for-each>
</xsl:template>
<!--constraintdefs-->
```

Рядком "<!--constraintdefs-->" позначено місце, куди записуються перетворення щодо окремих обмежень цілісності.

Наступне XSLT-перетворення відображає загальну схему визначення для первинного та унікального ключа:

```
<xsl:template match="key">
 SQLKEY ({$columnname}{$columnlist})
</xsl:template>
```

Тут *key* позначає *pk* або *uk*, а SQLKEY – відповідні ключові слова мови SQL – PRIMARY KEY та UNIQUE. Зазначимо, що змінні $columnname та $columnlist визначаються на рівні опрацювання тега <constraint>.

Наступні XSLT-перетворення відображають визначення обов’язковості та контролю значень атрибута:

```
<xsl:template match="notnull">
 NOT NULL
</xsl:template>
<xsl:template match="check">
 CHECK
 <xsl:text>(</xsl:text><xsl:value-of select="."/><xsl:text>)</xsl:text>
</xsl:template>
```

Наступні XSLT-перетворення відображають загальну схему визначення для первинного та унікального ключа:

```
<xsl:template match="fk">
 FOREIGN KEY ({$columnname}{$columnlist})
 REFERENCES <xsl:value-of select="tablename"/>(
  <xsl:apply-templates select="columnname"/>
  <xsl:apply-templates select="columnlist"/>
 )
</xsl:template>
```

Висновки

Потреба створення засобів створення SQL-сценаріїв з урахуванням особливостей діалектів SQL у різних СКБД зумовлює використання XML-технологій, які, з одного боку, придатні для відображення даних та метаданих, а з іншого – дають можливість їхнього структурного перетворення. Такі особливості XML використано для побудови крос-платформних відносно СКБД SQL-сценаріїв. Для генерації SQL-сценаріїв з урахуванням особливостей діалектів SQL розроблено відповідні XSLT-перетворення об'єктно-реляційного відображення даних в XTDQML-документи і XSLT-перетворення XTDQML-документів на SQL-запити.

1. Грабер М. SQL. Справочное руководство. Пер. с англ. – М.: Лори, 1997. – 291 с. 2. XML-DEV. – OASIS. – http://www.xml.org/xml/xmldev.shtml. 3. Буре Р. XML и базы даннях // Открытые системы. – 2000. – №10. – http://www.osp.ru/os/2000/10/178269/. 4. World Wide Web Consortium. XSL Transformations (XSLT). Version 1.0 – W3C Recommendation, 16 November 1999. – http://www.w3.org/TR/xslt. 5. Пахчанян А. Технологии электронного документооборота //Открытые системы. – 2002. – № 10. – С.17–21. – http://www.osp.ru/os/2002/10/181977/. 6. Thompson H.S. World Wide Web Consortium. XML Schema, Parts 0, 1, and 2 (Second Edition) /Thompson H.S., Beech D., Maloney M., Mendelsohn N. – W3C Recommendation, 28 October 2004. – http://www.w3.org/TR/xmlschema-0/, http://www.w3.org/TR/xmlschema-1/, http://www.w3.org/TR/xmlschema-2/. 7. Жежнич П. Часові бази даних (моделі та методи реалізації): Монографія. – Львів: Вид-во Нац. ун-ту "Львівська політехніка", 2007. – 260 с. 8. Дейт К.Дж. Введение в системы баз данных. 8-е изд. Пер. с англ. – М.: Вильямс, 2006. – 1328 с.