

从扩展 ER 模型到 XML 模式自动转换的算法研究

李思莉, 刘 耀

(成都理工大学 工程技术学院 计算机科学与技术系, 四川 乐山 614007)

摘 要: XML 已经成为 Internet 环境下数据表示、数据交换以及数据集成的标准。它独立于现有的数据库和编程语言, 具有异构性、可扩展性以及灵活性等优势。由于 XML 不是一个概念化的模型, 因此, 如何把从现实世界中抽象出的数据模型转换成 XML 逻辑模型, 进而设计出优良的 XML 文档成为研究热点。文中主要提出了一系列从扩展 ER 的概念模型到 XML 模式的转换规则, 并设计了转换算法, 以此实现自动操作。实验证明, 该算法自动生成的 XML Schema 文档能够真实反应扩展 E-R 模型, 并具有一定的通用性。

关键词: 扩展 ER; XML Schema; 算法

中图分类号: TP301.6

文献标识码: A

文章编号: 1673-629X(2010)11-0059-05

Auto-Conversion Algorithm From Extended Entity Relationship to XML Schema

LI Si-li, LIU Yao

(Department of Computer Science and Technology, Engineering & Technical College of
Chengdu University of Technology, Leshan 614007, China)

Abstract: XML has become data representation, data exchange and data integration standards under Internet environment. This is independent of databases and programming languages, with heterogeneity, scalability and flexibility advantages. As XML is not a conceptual model, therefore, how to transfer the data model that extracted from the real world to a logistic model and furthermore, devise a delicate XML document, has been become a hot spot nowadays. Provide a series of transformation rule from the extended ER conceptual model to XML Schema. Also, it has designed a conversion algorithm in order to achieve automatic operation. Experimental results show that automatically generated XML Schema by this algorithm can provide a distinct reflection about the extended E-R model with a certain degree of generality.

Key words: extended ER; XML Schema; algorithm

0 引言

XML 事实上已经成为 Internet 环境下数据表示、数据交换以及数据集成的标准。它独立于现有的数据库和编程语言, 具有异构性、可扩展性以及灵活性等优势。相比传统的关系型数据库而言, XML 文档结构更灵活。同时, 好的 XML 文档设计直接关系到应用程序数据的访问、修改、集成效率。由于 XML 不是一个概念化的模型, 因此, 如何把从现实世界中抽象出的数据模型转换成 XML 逻辑模型, 进而设计出优良的

XML 文档成为研究热点。文中专注于研究如何从扩展 ER 模式自动生成 XML Schema 的算法, 即根据已有的扩展 ER 图自动生成一个数据结构正确, 满足各种数据约束条件的 XML Schema 逻辑模型。

1 相关研究

早期, 从关系模式到 XML 模式有代表性的研究是 V. Turau 的 DB2XML^[1], 但它几乎是以一对一的方式把关系映射成了 XML 元素。在此基础上, 又有了 D. Lee 的 NeT&CoT^[2], 它考虑到了实体间的部份约束, 并通过在每个关系元组上重复应用套操作符(nest operator)推导出元素间的嵌套结构。由于这种推导并不是在类型的层次上, 因此推导出的嵌套结构可能是没有用处的。C. F. Liu 提出的文献[3], 在总结了以前

收稿日期: 2010-03-16; 修回日期: 2010-06-18

基金项目: 成都理工大学科研发展基金(C122010003)

作者简介: 李思莉(1974-), 女, 讲师, CCF 会员, 研究方向为计算机安全、数据库管理。

的研究成果后,提出了从关系模式生成 XML Schema 的转换规则,并在一定程度上保证了信息无损。即保留了 E-R 模型中的部分约束关系,同时也在一定程度上也保证了数据的完整性和一致性^[4,5]。对面向对象 XML 的存储模式进行了研究^[6-8]。直接研究了 XML 文档到关系数据的映射策略。在研究中,选择了 XML Schema 为 XML 文档的设计、验证模式。同时,由于 E-R 模型主要针对结构化数据建模,对于半结构化的 XML 并不完全适合。文献[9]指出了半结构化数据模型与 E-R 模型之间的差异。

文中主要采用扩展 ER 模型作为 XML 文档设计的概念模型,设计并实现了一个从概念模型到 XML Schema 逻辑模式设计的自动转换算法 EtoXS。实验证明,依据算法产生的 XML Schema 而设计的 XML 文档不仅满足了实体关系间存在的约束关系,也保证了数据的一致性和完整性。

2 扩展 ER 模型

文中的转换算法采用扩展 ER 模型(以下简称 EER)作为数据概念模型。它包括了 ER 模型中的所有概念,并在此基础上延伸出子类型、超类型、归纳、演绎以及属性继承等概念,增强了 ER 模型的语义表达形式。同时,采用 EER 作为概念模式,意味着提供的转换算法也同样适用于使用 UML 等其它概念模式的抽象模型。为了具体描述所使用的扩展 ER 模型,给出了如下正式的定义。

定义:设 $K = (E, R, A, n_d, \rho, s, \kappa)$ 。其中:

(1) E 是实体的集合,对 $\forall e(n_e, t) \in E$, n_e 表示实体名称, $t \in (\text{regular}, \text{weak}, \text{high-level})$ 表示实体类型。 regular 代表实体类型为一般实体, weak 代表为弱实体, high-level 代表拥有 isa 关系的超类型实体。

(2) R 是实体关系的集合。对 $\forall r(n_r, \{(e, \text{card}, \text{part}, \text{cons}, \text{role})\}, t) \in R$ 。其中 n_r 表示关系名称,集合中的每个元组 $(e, \text{card}, \text{part}, \text{role})$ 用来描述所涉及到的实体集。 $e \in E$, $\text{card} \in (1, n)$, $\text{part} \in (\text{total}, \text{partial})$, $\text{cons} \in (\text{disjoint}, \text{overlap})$, $t \in (\text{generalization}, \text{specialization}, \text{aggregation})$, part 和 cons 的结合可用来描述演绎或归纳过程的正交约束和完全性约束。 role 代表实体在关系集合中所扮演的角色。相同的实体在所涉及的关系中可以成为不同的角色。

(3) A 是属性的集合,对 $\forall a(n_a, vt, st) \in A$ 表示属性名, $vt \in (\text{single-value}, \text{multi-value})$, $st \in (\text{automatic}, \text{composite})$ 。

(4) n_d 表示扩展 ER 图的名称。

(5) $\rho: E \rightarrow E$ 定义了对每一个弱实体 $e \in E$,

$\rho(e)$ 表示了弱实体 e 所依赖的实体。

(6) $s: E \rightarrow E$ 定义一个 ISA 关系。对每一个 $e \in E$, $s(e)$ 表示 e 的子类型。

(7) $\kappa: E \rightarrow 2^A$ 定义了实体集的键值集合。

3 扩展 ER 到 XML Schema 转换映射规则

在设计中,把整个转换过程分为两个部分。第一部分为基本转换规则,负责处理一般实体和弱实体;第二部分为语义转换规则,负责处理实体间各种关系以及约束,包括 ISA,归纳、演绎、聚合等。

3.1 基本转换规则

规则 1 对 EER 图 $K = (E, R, A, n_d, \rho, s, \kappa)$, n_d 转换为所有元素的根元素。

规则 2 对 $e(n_e, t)$, 如果 $t(e) = \text{"regular"}$, 则把实体直接映射成元素。对 $k(e) = \{k_1, k_2, \dots, k_n\}$, 直接用 $\langle \text{key} \rangle$ 来标记。

```
< xs:element name = "n_d" >
```

```
.....
```

```
< /xs:element >
```

```
< xs:key name = "key_n_e" >
```

```
< xs:selector xpath = "path_n_e" />
```

```
< xs:field xpath = "k_1" /> ..... < xs:field xpath = "k_n" />
```

```
< /xs:key >
```

规则 3 对 $e(n_e, t)$, 如果 $t(e) = \text{"weak"}$, 并且 $\rho(e) = e_1$, 则把元素 n_e 放在元素定义 n_{e_1} 下, 作为 n_{e_1} 的子元素。对 $k(e) = \{k_1, k_2, \dots, k_n\}$, $k(e_1) = \{k_{11}, k_{21}, \dots, k_{n1}\}$, 有:

```
< xs:key name = "key_n_e" >
```

```
< xs:selector xpath = "path_n_e" />
```

```
< xs:field xpath = "../k_11" /> ..... < xs:field xpath = "../k_n1" />
```

```
< xs:field xpath = "k_1" /> ..... < xs:field xpath = "k_n" />
```

```
< /xs:key >
```

3.2 语义转换规则

规则 4 对 $e(n_{e_1}, t)$, 如果 $t(e_1) = \text{"high-level"}$, 存在 $s(e_1) = e_2$, 则在元素 n_e 中使用 "extension" 标记来继承 e_1 的特性。

规则 4.1 对 $e(n_{e_1}, t)$, 如果 $t(e_1) = \text{"high-level"}$, 存在 $s(e_1) = \{e_2, e_3, \dots, e_n\}$, 当 $r(n_r, \{(e_1, \text{card}, \text{part}, \text{cons}, \text{role})\}, t)$, $t(r) \in (\text{generalization}, \text{specialization})$, $\text{cons} = \text{"disjoint"}$, 则把 e_2, e_3, \dots, e_n 作为子元素加入 e_1 的定义中, 并用 "choice" 来表示这些子类型是正交的, 当 $\text{cons} = \text{"overlap"}$, 则扩展规则 4, 添加对子类型的引用。演绎与归纳互为逆过程, 因此规则 4.1 同样适用于 ($\text{specialization}, \text{disjoint}$) 和 ($\text{specialization},$

overlap) 的情况。

规则 4.2 对 $e(n_{e_1}, t)$, 如果 $t(e_1) = \text{"high-level"}$, 存在 $s(e_1) = \{e_2, e_3, \dots, e_n\}$, 对关系 $r(n_r, \{(e_1, \text{card}, \text{part}, \text{cons}, \text{role})\})$, 如 $t(r) = \text{"aggregation"}$, 则创建名为 n_r 的元素, 对 $\{e_2, e_3, \dots, e_n\}$ 使用基本转换规则转换成元素, 在元素 n_r 中添加对 $\{e_2, e_3, \dots, e_n\}$ 的引用, 如图 1 所示。

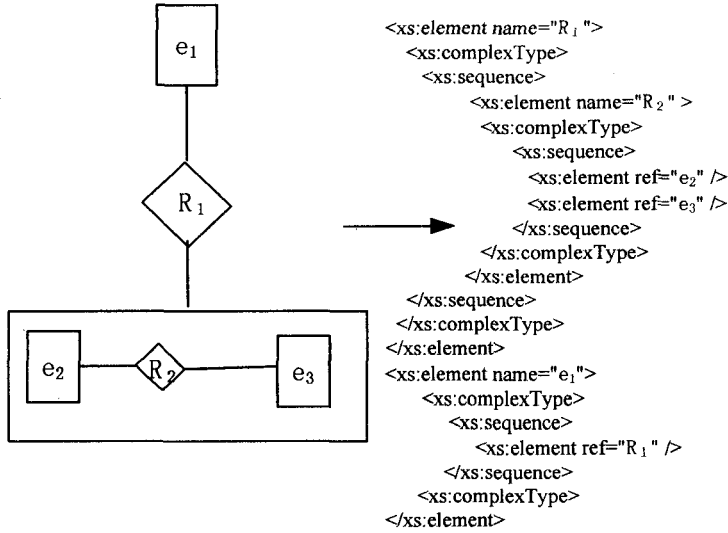


图 1 聚合关系的转换

规则 5 对 $r(n_r, \{(e_1, 1, p_1, -, -), (e_2, 1, p_2, -, -)\}, t)$, 首先创建 n_r 元素, 然后根据 p_1 和 p_2 的不同, 采用不同的规则。当 p_1 和 p_2 都等于“total”, $k(e_1)$ 是 e_2 的外键, 则把 n_r 作为 e_2 的子元素, e_2 放在 e_1 元素定义下。添加 $\text{minOccurs} = \text{"1"}$ 和 $\text{maxOccurs} = \text{"1"}$ 属性给 e_2 ; 当 p_1 和 p_2 都等于“partial”, 假设 e_1 的主键是 e_2 的外键之一。则对 $k(e_2) = \{k_1, k_2, \dots, k_n\}$, 用“keyref”来联接 e_1 和 e_2 , 如图 2 所示。

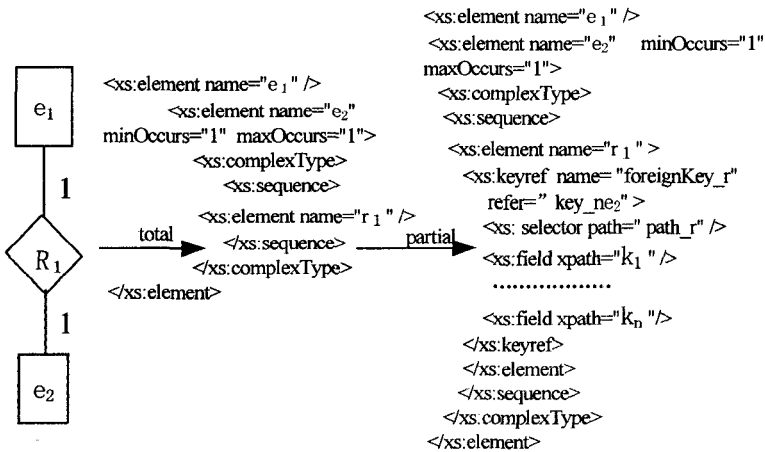


图 2 基约束与参与约束转换图

规则 6 对 $r(n_r, \{(e_1, 1, p_1, -, -), (e_2, n, p_2, -, -)\}, t)$, 令 $\text{maxOccurs} = \text{"unbounded"}$, 其余与规

则 5 同理。

规则 7 对 $r(n_r, \{(e_1, n, -, -, -), (e_2, n, -, -, -)\}, t)$, 按基本规则转换 e_1 和 e_2 , 创建 n_r 元素, 加入 $\text{maxOccurs} = \text{"unbounded"}$ 属性, 并添加对 e_1 和 e_2 的引用。对“keyref”的定义同规则 5, 如图 3 所示。

规则 8 对形如 $r(n_r, \{(e_1, c_1, -, -, r_1), (e_1, c_2, -, -, r_2)\}, t)$ 的一元关系, 根据基约束 $c_1:c_2$, 如果 $c_1 = c_2 = 1$ 或 $c_1 \neq c_2$, 假定 r_1 是主要角色, 则创建元素 n_{r_1} , 并把它置于 e_1 之下, 用 keyref 声明限定 r_2 的外键属性; 如果 $c_1 = c_2 = n$, 则只需要在 n_{r_1} 元素中添加 $\text{maxOccurs} = \text{"unbounded"}$ 属性。

规则 9 对形如 $r(n_r, \{(e_1, c_1, -, -, -) \dots (e_n, c_n, -, -, -)\}, t)$ 的 n 元关系, 按 1:1, 1:n 和 m:n 的不同, 设置不同的 minOccurs 和 maxOccurs 值。余下的转换与规则 7 同理。

规则 10 对实体 e , 关系 r 或组合属性 a' 的属性 $a(\text{na}, \text{vt}, \text{st})$, 如果 $\text{st}(a) = \text{"composite"}$, 则创建元素 n_a , 并放置在实体 e , 关系 r 或组合属性 a' 的定义之下。如果 $\text{vt}(a) = \text{"multi-valued"}$, 则需要添加 $\text{maxOccurs} = \text{"unbounded"}$ 属性。

规则 11 对实体 e , 关系 r 或组合属性 a' 的属性 $a(n_a, \text{vt}, \text{st})$, 如果 $\text{st}(a) = \text{"automatic"}$, 当 $\text{vt}(a) = \text{"multi-valued"}$ 时, 运用规则 10; 当 $\text{vt}(a) = \text{"single-valued"}$, 则直接作为与之相关元素的属性。

4 自动转换算法 EtoXS

为了使上述规则能够自动转换成 XML Schema 文件, 设计了一个自动转换算法。该算法是在文献[10]的基础上, 加入 EER 概念修改实现的。

4.1 算法描述

输入: 一个扩展 ER 图 $K = (E, R, A, n_d, \rho, s, \kappa)$

输出: 一个完整的 XML Schema 步骤:

(1) 对 $K = (E, R, A, n_d, \rho, s, \kappa)$ 应用规则 1 产生根元素;

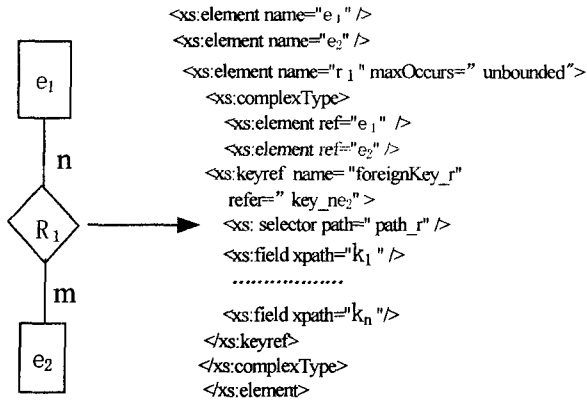
(2) $E_1 = \{e \mid e \in E \wedge t(e) = \text{"regular"}\}$;

(3) $E_2 = E - E_1$;

(4) for each $e \in E_1$ {

(5) if $s(e) = \Phi$

(6) {应用规则 2 产生元素 e ;

图 3 $n:m$ 基约束转换

```

(7)  $E_1 = E_1 - \{e\}$ ;
(8) }
(9) }
(10) while  $E_2 \neq \Phi$  do {
(11) get  $e \in E_2$ ;
(12) if  $t(e) = \text{"weak"}$  and  $p(e) \in E_2$ 
(13) { 应用规则 3 创建元素  $e$ ;
(14)  $E_2 = E_2 - \{e\}$ ;
(15) }
(16) while  $E_2 \neq \Phi$  do {
(17) get  $e \in E_2$ ;
(18) if  $t(e) = \text{"high-level"}$ 
(19) { if  $s(e_1).count = 1$ 
(20) { 应用规则 4 创建元素  $e$ ;
(21) EtoXS( $k$ );
(22)  $E_2 = E_2 - \{e\}$ ;
(23) }
(24) else if  $r.e.t(r)$  in ("generalization",
"specialization")
(25) { 应用规则 4.1 创建元素  $e$ ;
(26) EtoXS( $k$ );
(27)  $E_2 = E_2 - \{e\}$ ;
(28) }
(29) else if  $r.e.t(r) = \text{"aggregation"}$ 
(30) { 应用规则 4.2 创建元素  $e$ ;
(31) EtoXS( $k$ );
(32)  $E_2 = E_2 - \{e\}$ ;
(33) }
(34)  $R_1 = R$ ;
(35) for each  $r \in R_1$ 
(36) { if  $\text{nary}(r) = 1$ 
(37) { 应用规则 8 来产生元素  $r$ ;
(38)  $R_1 = R_1 - \{r\}$ ;

```

(39) }

(40) else if $\text{nary}(r) > 2$ (41) { 应用规则 9 来为 r 产生元素以及嵌套结构;(42) $R_1 = R_1 - \{r\}$;

(43) }

(44) else if $\text{card}(r.e_1) = "n"$ and $\text{card}(r.e_2) = "n"$ (45) { 应用规则 7 来为 r 产生元素以及嵌套结构;(46) $R_1 = R_1 - \{r\}$;

(47) }

(48) else if $\text{card}(r.e_1) = "n"$ or $\text{card}(r.e_2) = "n"$ (49) { 应用规则 6 来为 r 产生元素以及嵌套结构;(50) $R_1 = R_1 - \{r\}$;

(51) }

(52) }

(53) for each $r \in R_1$ (54) { 应用规则 5 来为 r 产生元素以及嵌套结构;(55) $R_1 = R_1 - \{r\}$;

(56) }

(57) for each $a \in A$ (58) { if $\text{st}(a) = \text{"composite"}$ (59) { 应用规则 10 来产生 a 的元素和属性以及嵌套结构;(60) $A = A - \{a\}$

(61) }

(62) else

(63) { 应用规则 11;

(64) $A = A - \{a\}$;

(65) }

(66) }

4.2 一个实例

为了验证算法,文中建立了一所学校简单的数据模型。由于篇幅所限,该模型简化了其中大部份的属性和关系,并只提供了部分实验代码。如图 4 所示,各实体间拥有超类、子类、聚合以及演绎等关系,如实体 employee 和 Student 的聚合形成了实体 person,并且 Employee 又正交演绎成了 Staff 和 Faculty 等。接着,使用前述的转换规则和算法把该 EER 模型转换为 XML Schema 文档。

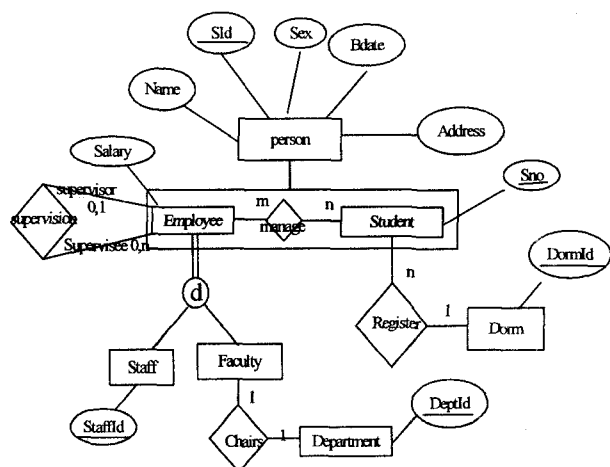


图 4 扩展 ER 图

转换以后的 XML Schema 文档:

```

<xs:element name="university" type="university_type" />
<xs:complexType name="university_type">
  <xs:sequence>
    <xs:element name="people" type="people_type" />
  </xs:sequence> </xs:complexType>
<xs:complexType name="people_type">
  <xs:sequence>
    <xs:element name="manage">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="employee" />
          <xs:element ref="student" />
        </xs:sequence>
      </xs:complexType>
      <xs:keyref name="foreignKey_stu" refer="key_Student">
        <xs:selector xpath="//manage" />
        <xs:field xpath="@manage" />
      </xs:keyref>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="SId" type="xs:string" />
  <xs:attribute name="Name" type="xs:string" />
  <xs:attribute name="Sex" type="xs:string" />
  <xs:attribute name="Bdate" type="xs:date" />
  <xs:attribute name="Address" type="xs:date" />
</xs:complexType>
<xs:element name="employee" type="employee_type maxOccurs="unbounded" />
<xs:complexType name="employee_type">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="staff" />
      <xs:element ref="department" />
    </xs:choice>
    <xs:element name="supervise_emp">

```

```
<xs:complexType>
  <xs:attribute name="supervision" type="xs:int"/>
</xs:complexType>
<xs:keyref name="foreignKey..supervise" refer="key..employee">
<xs:selector xpath="//supervise..emp"/><xs:field xpath="@supervision"/>

```

5 结束语

文中提供了一系列从扩展 ER 的概念模型到 XML 模式的转换规则,并设计了转换算法 EtoXS,以此实现自动操作。该 XML Schema 作为 XML 文档设计的逻辑模型,具有一定的通用性。由于实体间存在的关系大多通过引用来实现,因此,在一定程度上会降低生成的 XML 文档的处理效率。在以后的研究中,将考虑引入工作负载作为实体转换的条件之一,以期提高处理效率。也会在文献[11,12]的基础上,研究 XML 的查询语言。

参考文献:

- [1] Turau V. Making Legacy Data Accessible for XML Applications[EB/OL]. 2001. <http://www.informatik.fh-wiesbaden.de/turau/DB2XML/>.
- [2] Lee D, Mani M, Chiu F, et al. NeT & CoT: Translating Relational Schemas to XML Schemas Using Semantic Constraints [C]. McLean, VA, USA: [s. n.], 2002: 282 - 291.
- [3] Liu C, Vincent M W, Liu J. Constraint Preserving Transformation from Relational Schema to XML Schema[J]. World Wide Web Journal, 2006, 9(1): 93 - 110.
- [4] 张晓琳, 谭跃生, 周 健. 基于面向对象 XML 的存储模式的设计与实现[J]. 计算机应用, 2005, 25(9): 1995 - 1998.
- [5] 周 健, 孙丽艳. 面向对象 XML 的存储模式的研究[J]. 计算机技术与发展, 2009, 19(3): 114 - 117.
- [6] 朱珊珊, 李书琴, 安福定. XML 文档到关系数据库的转换研究[J]. 计算机工程与设计, 2008, 29(21): 5507 - 5509.
- [7] 耿 飏, 宋余庆, 梁成全, 等. XML 文档到关系数据库映射方法的研究[J]. 计算机应用研究, 2010, 27(3): 951 - 954.
- [8] 李思莉, 李 娟. XML 文档到关系数据库的映射策略[J]. 计算机工程, 2010, 36(5): 40 - 42.
- [9] 刘洪星, 陈 明, 张学敏. 几种基于 EER 的 XML 概念模式的研究[J]. 武汉理工大学学报, 2006, 28(5): 25 - 28.
- [10] Liu C, Li J. Designing quality xml schemas from e - r diagrams [M]//Advances in Web - Age Information Management Heidelberg: Springer, 2006: 508 - 159.
- [11] 孟小峰, 王 宇, 王小峰. XML 查询优化研究[J]. 软件学报, 2006, 17(10): 2069 - 2086.
- [12] 华珊珊, 谢铨洋. XML 查询语言 XQuery 的研究与实现[J]. 计算机技术与发展, 2009, 19(4): 48 - 50.