DOI:10.3969/j.issn.1673-4785.201111032
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.TP.20121116.1702.012.html

# 一种基于领域知识的XML数据模糊查询

孟祥福[1], 张霄雁[1], 马宗民[2], 彭晏飞[1]
(1.辽宁工程技术大学 电子与信息工程学院,辽宁 葫芦岛 125105; 2.东北大学 信息科学与工程学院,辽宁 沈阳 110819)

**摘 要**:为了解决普通用户对XML数据的模糊查询问题,提出了一种基于领域知识的XML数据模糊查询方法.以模糊集理论为基础,首先介绍了XML数据模糊查询的构成形式;然后提出了将领域知识和模糊集的隶属函数相结合的方法实现XML数据的模糊查询条件转换,转换过程考虑了查询谓词的重要程度和用户偏好;最后按结果元素对模糊查询的满足程度对模糊查询结果进行排序.该方法无需改变传统的XML查询语言和XDBMS就能够实现模糊查询,从而提高了用户与系统之间的交互能力.实验结果表明,提出的模糊查询方法具有较高的查全率和准确率.

**关键词**:XML;模糊查询;领域知识;用户偏好;排序
**中图分类号**:TP311.13 **文献标志码**:A **文章编号**:1673-4785(2012)06-0525-11

# An XML fuzzy query answering approach based on domain knowledge

MENG Xiangfu[1], ZHANG Xiaoyan[1], MA Zongmin[2], PENG Yanfei[1]
(1. College of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China; 2. College of Information Science and Engineering, Northeastern University, Shenyang 110819, China)

**Abstract**: To deal with the problem of XML fuzzy query for common users, this paper proposes a domain knowledge-based XML data fuzzy query approach. Based on fuzzy sets theory, this paper firstly introduces the form of XML fuzzy query. And then, an approach which leverages the domain knowledge and membership function of fuzzy set to realize the fuzzy query translation of XML data, is presented. The importance of querying predicates and user preferences are taken into consideration when the fuzzy query is translated. Finally, for the fuzzy query results, the elements of them are ranked according to their satisfaction degree to the original fuzzy query. This approach can realize the fuzzy query without modifying XML query language and the XDBMS, and thus it can help users to improve their interaction with the system. Results of experiments demonstrate that the fuzzy query approach proposed in this paper has high recall ratio and precision.

**Keywords**: XML; fuzzy query; domain knowledge; user preference; ranking

随着WWW的迅速膨胀和Internet的普遍应用,访问Web已成为人们获取信息的重要手段.XML规范是当今Web上信息表示与交换的标准,大量的异构数据因而也就集成在XML文档中[1].然而在现实中,普通Web用户的查询意图本身就可能是模糊或不精确的,他们所提交的查询大多数情况下也只是其查询意图的模糊描述.因此,用户希望系统能够支持其模糊查询要求的表达,通过直接含有模糊词或模糊关系的模糊语言查询XML数据.

来看一个例子,在一个存储房产信息的XML文档(HouseDB.xml)中,假设用户想查找"价格至多$300 000,建筑时间比较近,面积在130~200 $m^2$"的房产,可能会输入如下XPath查询语句:

$\widetilde{Q}$: -/HouseDB/House[Price at most 300 000]
[Buildyear ='Recent'][SqFt between 130 and 200].

其中,基于内容的查询约束在XPath中用谓词的形

收稿日期:2011-11-30. 网络出版日期:2012-11-16.
基金项目:国家青年科学基金资助项目(61003162).
通信作者:孟祥福. E-mail:marxi@126.com.

式表示. 很明显,该查询包含了 3 个查询谓词,分别是“Price at most 300 000”、“Buildyear = 'Recent'”和“SqFt between 130 and 200”,其中前 2 个查询谓词分别包含了模糊关系“at most”和模糊词“Recent”. 这里,将包含模糊关系或模糊词的查询谓词称为模糊查询谓词. 如果一个以 XPath 形式表示的 XML 查询中包含了一个或多个模糊查询谓词,则称之为 XML 模糊查询. 然而,目前的 XML 数据查询处理模式仅支持精确查询匹配,还不能直接处理这样的模糊查询. 因此,系统首先需要将模糊查询条件转换为精确查询条件,然后再提交给 XML 数据管理系统执行,进而获取满足模糊查询要求的结果.

在关系数据库领域,Tahani[2] 最先基于 Zadeh 提出的模糊集[3] 来处理关系数据库中含有不精确查询条件的模糊查询;Bosc 等对标准 SQL 进行了模糊扩展[4-5],提出了 SQLf 语言,该语言也是对以前经典数据库模糊查询方法(如 Tahani[3]、Bosc[6]、Nakajima[7])的特点和功能的集成. Chen[8] 和 Ma[9] 提出了模糊查询条件转换规则,能够处理查询条件中包含的模糊关系或模糊词. 文献[10-11]提出了基于模糊集的模糊描述逻辑的推理方法. 近年来,研究者开始关注 XML 数据的柔性查询工作[12-17]. 文献[12-13]利用本体处理 XML 文档的柔性查询,使用查询重写机制获取近似查询结果,查询重写过程依赖于领域专家参与建立的本体知识库和映射规则. 文献[14]提出了 XML 近似查询结构匹配 TreeSketch 算法,通过在精确的 TreeSkech 上构建概要来处理 XML 小枝模式查询,进而快速返回结构近似匹配的查询结果. 文献[15]在一个 XML 基本模式中,首先鉴定用户提出的 XML 查询模式与其他模式的结构相似性,然后在查询处理阶段通过自动重写当前查询,使查询模式与其他相关 XML 模式兼容,从而找到更多模式匹配信息. 文献[16]提出了 AQAX 在线 XML 查询系统,该系统基于 XML 数据概要实现快速的大规模数据集探测,AQAX 使用了双层体系结构:第 1 层在 XML 聚类(XCluster)框架上生成近似结果;第 2 层在 XML 聚类元素上构建分类树,为用户提供查询可视化界面. 文献[17]使用基本变异操作将用户初始 XML 查询树改写成系列变异查询树,在此基础上根据基本变异操作代价和嵌入代价得到松弛查询树,从而实现近似查询. 然而需要指出的是,现有的 XML 数据柔性/近似查询方法都是对精确查询条件进行放松处理,进而获取相关查询结果,但这类方法不支持用户直接表达模糊查询要求,并且也不能对模糊查询要求进行转换处理. 因此,本文在模糊集理论基础上,提出了一种直接支持用户模糊查询要求的查询处理方法,该方法能够处理模糊关系和模糊词以及对精确数值区间进行模糊扩展,并且查询处理过程考虑了用户偏好. 该方法能够在不改变传统 XML 查询语言和 XML 数据库引擎的前提下实现模糊查询,从而提高用户与系统之间的交互能力.

## 1 XML 模糊查询谓词的构成形式

传统的 XPath 查询形式由路径(path)和基于内容的查询谓词(value-based predicate)2 部分构成. 基于内容的查询谓词形式为 $A\theta Y$,其中 $A$ 代表 XML 数据树的叶节点,$\theta$ 代表操作运算中的规则关系(例如 =、>、<、≥、≤、≠、(not) between),$Y$ 代表操作数(用户查询要求中指定的精确查询值). 在实际应用中,由于用户提出的模糊查询要求通常体现在基于内容的查询谓词中,也就是说在查询谓词中使用模糊关系(如“close to”)或模糊词(如“young”). 因此,本文主要研究 XML 模糊查询谓词的构成形式及其向精确查询谓词的转换方法. 模糊查询谓词是由规则关系与模糊词的结合或模糊关系(如(not) at most、(not) at least、(not) close to/around)与精确值的结合而构成的,这类基本模糊谓词条件可由具有隶属函数的模糊集来表示.

### 1.1 模糊词作为操作数

模糊词作为操作数与规则关系作为操作符构成的模糊查询谓词,形式为 $A\theta\tilde{Y}$,其中 $A$ 代表 XML 数据树的叶节点,$\theta$ 代表规则关系,$\tilde{Y}$ 是模糊词. 其中模糊词包括简单模糊词、复合模糊词和混合模糊词,它们的含义都可由模糊集合表示[8].

### 1.2 模糊关系作为操作符

精确值作为操作数与模糊关系作为操作符构成的模糊查询谓词,形式为 $A\tilde{\theta}Y$,其中 $A$ 代表 XML 数据树的叶节点,$\tilde{\theta}$ 是模糊关系,$Y$ 代表精确值,$\tilde{\theta}Y$ 是一个模糊集. 在前期工作[9, 18] 中,笔者已讨论了由上述 3 种模糊关系与精确值构成的模糊集隶属函数的构建方法.

### 1.3 数值区间作为操作数

数值区间作为操作数与规则关系作为操作符构成的查询条件,形式为 $A\theta Y$,其中 $A$ 代表 XML 数据树的叶节点,$\theta$ 代表规则关系“between”,$Y$ 代表数值区间,用 $[Y_1, Y_2]$ 表示. $A\theta Y$ 的形式为“$A$ between $Y_1$ and $Y_2$”. 这里,按照隶属函数和用户给出的阈值对数值区间进行模糊扩展.

下面,结合具体实例描述以上 3 种情况下模糊查询谓词向精确查询谓词的转换方法.

## 2　领域知识库

领域知识库由若干 XML 文档构成,存储领域知识信息,这些信息用来对模糊查询谓词进行转换,这些信息包括模糊查询谓词涉及的叶节点,以及根据叶节点的语义对模糊查询谓词进行扩展的范围和程度. 领域知识库由领域专家维护或自学习获取知识.

### 2.1　文档的 DTD

本文引言部分的模糊查询谓词转换需要如下 4 个 XML 文档.

文档 NodeRelax. xml 提供 XML 数据树的叶节点信息. llimit 和 rlimit 分别代表查询谓词所涉及的叶节点上进行扩展的数值区间左极限和右极限; nimp 中的语义词代表该叶节点在 XML 文档中的重要程度,这也是查询条件中相应的查询谓词的重要程度.

#### 2.1.1　NodeRelax. xml 的 DTD

```
<! DOCTYPE NodeRelax[
    <! ELEMENT NodeRelax (nrelax * ) >
    <! ELEMENT nrelax (leaf_node, llimit, rlimit, nimp) >
    <! ELEMENT leaf_node (#PCDATA) >
    <! ELEMENT llimit (#PCDATA) >
    <! ELEMENT rlimit (#PCDATA) >
    <! ELEMENT nimp (#PCDATA) > ] >
```

文档 NodeImportance. xml 存储表达叶节点重要程度的语义词(如 high、medium、low 等),mdegree 代表 nimp 中不同语义词所对应的隶属度.

#### 2.1.2　NodeImprotance. xml 的 DTD

```
<! DOCTYPE NodeImportance[
  <! ELEMENT NodeImportance (nimportance * ) >
  <! ELEMENT nimportance (nimp, leaf_node, mdegree) >
  <! ELEMENT nimp (#PCDATA) >
  <! ELEMENT leaf_node (#PCDATA) >
  <! ELEMENT mdegree (#PCDATA) > ] >
```

文档 Relaxation. xml 指明了查询谓词在相应叶节点上的扩展范围、扩展程度和叶节点上的值对查询谓词的满足方式. Directionrel 代表查询扩展范围,例如查询谓词"Price around(大约)100 000",如果 Directionrel 的值为"left、right",则"Price"的扩展范围同时包括小于和大于 100 000 的值. ldegrel 和 rdegrel 分别代表查询谓词的左右扩展程度. lsatisfy 和 rsatisfy 分别表示扩展范围内叶节点的值对查询谓词的满足方式. 例如,当 lsatisfy 取值为"non desc"时,表示叶节点"Price"上小于 100 000 的值对查询谓词的满足程度等于 1;当 lsatisfy 取值为"desc"时,表示叶节点"Price"上小于 100 000 的值对查询谓词的满足程度小于 1.

#### 2.1.3　Relaxation. xml 的 DTD

```
<! DOCTYPE Relaxation[
  <! ELEMENT Relaxation (relax * ) >
  <! ELEMENT relax (leaf_node, operator, directionrel, ldegrel, rdegrel, lsatisfy, rsatisfy) >
  <! ELEMENT leaf_node (#PCDATA) >
  <! ELEMENT operator (#PCDATA) >
  <! ELEMENT directionrel (#PCDATA) >
  <! ELEMENT ldegrel (#PCDATA) >
  <! ELEMENT rdegrel (#PCDATA) >
  <! ELEMENT lsatisfy (#PCDATA) >
  <! ELEMENT rsatisfy (#PCDATA) > ] >
```

文档 FuzzyTerm. xml 存储描述叶节点取值范围的模糊词,每个模糊词都对应一个预先定义的模糊集,模糊集的隶属函数参数分别取自元素 para1、para2、para3、para4 中的值.

#### 2.1.4　FuzzyTerm. xml 的 DTD

```
<! DOCTYPE FuzzyTerm[
  <! ELEMENT FuzzyTerm (fterm * ) >
  <! ELEMENT fterm (fuzzy_term, leaf_node, para1, para2, para3, para4) >
  <! ELEMENT fuzzy_term (#PCDATA) >
  <! ELEMENT leaf_node (#PCDATA) >
  <! ELEMENT para1 (#PCDATA) >
  <! ELEMENT para2 (#PCDATA) >
  <! ELEMENT para3 (#PCDATA) >
  <! ELEMENT para4 (#PCDATA) > ] >
```

### 2.2　文档实例

#### 2.2.1　NodeRelax. xml 文档实例

```
<NodeRelax >
  <nrelax >
    <leaf_node >Price </leaf_node >
    <llimit >150 000 </llimit >
    <rlimit >1 000 000 </rlimit >
    <nimp >high </nimp >
  </nrelax >
  <nrelax >
    <leaf_node >SqFt </leaf_node >
    <llimit > - </llimit >
    <rlimit >800 </rlimit >
```

```
    <nimp > medium </nimp >
  </nrelax >
  <nrelax >
    <leaf_node > Buildyear </leaf_node >
    <llimit > 1981 </llimit >
    <rlimit > - </rlimit >
    <nimp > medium </nimp >
  </nrelax >
  …
</NodeRelax >
```

2.2.2 NodeImportance.xml 文档实例

```
<NodeImportance >
  <nimportance >
    <nimp > high </nimp >
    <leaf_node > Price </ leaf_node >
    <mdegree > 0.8 </mdegree >
  </nimportance >
  <nimportance >
    <nimp > medium </nimp >
    <leaf_node > SqFt </ leaf_node >
    <mdegree > 0.5 </mdegree >
  </nimportance >
  <nimportance >
    <nimp > medium </nimp >
    <leaf_node > Buildyear </ leaf_node >
    <mdegree > 0.5 </mdegree >
  </nimportance >
  …
</NodeImportance >
```

2.2.3 Relaxation.xml 文档实例

```
<Relaxation >
  <relax >
    <leaf_node > Price </leaf_node >
    <operator > at most </operator >
    <directionrel > right </directionrel >
    <ldegrel > - </ ldegrel >
    <rdegrel > 0.2 </rdegrell >
    <lsatisfy > nondecr </lsatisfy >
    <rsatisfy > decr </rsatisfy >
  </relax >
  <relax >
    <leaf_node > SqFt </leaf_node >
    <operator > between </operator >
    <directionrel > left, right </directionrel >
    <ldegrel > - </ldegrel >
    <rdegrel > - </rdegrell >
    <lsatisfy > decr </lsatisfy >
    <rsatisfy > nondecr </rsatisfy >
  </relax >
  <relax >
    <leaf_node > Buildyear </leaf_node >
    <operator > = </operator >
    <directionrel > left, right </directionrel >
    <ldegrel > - </ ldegrel >
    <rdegrel > - </rdegrell >
    <lsatisfy > decr </lsatisfy >
    <rsatisfy > decr </rsatisfy >
  </relax >
  …
</Relaxation >
```

2.2.4 FuzzyTerm.xml 文档实例

```
<FuzzyTerm >
  <fterm >
    <fuzzy_term > recent </ fuzzy_term >
    <leaf_node > Buildyear </ leaf_node >
    <para1 > 0 </para1 >
    <para2 > 0 </para2 >
    <para3 > 5 </para3 >
    <para4 > 10 </para4 >
  </fterm >
  <fterm >
    <fuzzy_term > moderate </ fuzzy_term >
    <leaf_node > SqFt </ leaf_node >
    <para1 > 80 </para1 >
    <para2 > 130 </para2 >
    <para3 > 200 </para3 >
    <para4 > 250 </para4 >
  </fterm >
  …
</FuzzyTerm >
```

## 3 XML 模糊查询转换方法

本文提出的 XML 模糊查询转换实现方法首先将模糊查询条件中的每个模糊查询谓词用相应的模糊集表示,然后借助领域知识库中的领域知识构建隶属函数,同时根据模糊查询谓词的重要程度(在提出查询要求时,用户可根据个人偏好指定每个查询谓词的重要程度),利用权重函数对其进行放松处理,最后根据阈值和 α-截集运算将其转换成精确查询谓词.本节将使用引言中的例子:

$\widetilde{Q}$: -/HouseDB/House[Price at most 300 000][Buildyear = 'Recent'][SqFt between 130 and 200],结领域知识库中的领域知识,描述不同情况下的模糊查询谓词转换实现方法.

### 3.1　含模糊关系的模糊查询谓词转换

对于模糊查询条件中的模糊查询谓词"Price at most 300 000",使用模糊集 $P$ 表示该模糊查询谓词.为了确定模糊集 $P$ 的隶属函数,首先从领域知识库文档 Relaxation.xml 中找出对应模糊关系"at most"和叶节点"Price"的左右扩展方向(Relaxation.xml 中的 directionrel 元素),叶节点值对模糊查询谓词的满足方式 Relaxation.xml 中的 lsatisfy 和 rsatisfy 元素.这里,对应叶节点"Price"和模糊关系"at most"的扩展方向 directionrel 元素取值为"right",叶节点值对模糊查询谓词的满足方式 lsatisfy 和 rsatisfy 元素取值分别为"nondecr"、"decr".然后,再根据模糊集"close to $Y$"的隶属函数[9],可将模糊集 $P$ 的隶属函数定义为:

$$P(p) = \begin{cases} 0, & p \geqslant d; \\ 1, & p \leqslant 300\ 000; \\ \dfrac{d-p}{d-300\ 000}, & 30\ 000 < p < d. \end{cases} \quad (1)$$

对于参数 $d$ 的计算,需要用到 Relaxation.xml 中提供的扩展程度 rdegrel 元素的值.在 Relxation.xml 中叶节点 Price 上对应模糊关系"at most"的向右扩展程度 rdegrel = 0.2,该扩展程度与模糊查询谓词的重要程度 $\mu_{\text{importance}}(\text{high}) = 0.8$ 相关,于是有

$$\text{rdegrel} = (d - 300\ 000)/300\ 000 = 0.2 \Rightarrow$$

$$d = 360\ 000,$$

因此,隶属函数式(1)可转换成:

$$P(p) = \begin{cases} 0, & p \geqslant 360\ 000; \\ 1, & p \leqslant 300\ 000; \\ \dfrac{360\ 000-p}{60\ 000}, & 30000 < p < 360000. \end{cases}$$

假设用户(或系统)设定的阈值为 0.8,则模糊集 $P$ 的 0.8 截集运算结果为:

$$P(p) = \frac{360\ 000-p}{60\ 000} = 0.8 \Rightarrow p \leqslant 312\ 000.$$

根据文献[8-9]提出的模糊查询转换规则,模糊查询谓词"Price at most 300 000"可转换成精确查询谓词"Price 312 000".

### 3.2　含模糊词的模糊查询谓词的转换

对于模糊查询条件中的模糊查询谓词"Buildyear = more or less recent (建筑时间比较近)",使用模糊集 $B$ 表示该模糊查询谓词.首先从知识库文档 FuzzyTerm.xml 中找到描述叶节点"Buildyear"的模糊词"Recent"及其对应的参数值(注意,计算过程中将建筑时间转换成距今的年数).然后,构建模糊词"Recent"的隶属函数,即

$$\mu_{\text{Recent}}(b) = \begin{cases} 1, & b \leqslant 5; \\ \dfrac{10-b}{5}, & 5 < b < 10; \\ 0, & b \geqslant 10. \end{cases}$$

根据扩张原理和模糊词"Recent"的隶属函数,则可得模糊集 $B$ 的隶属函数为

$$B(b) = \begin{cases} 1, & b \leqslant 5; \\ \left(\dfrac{10-b}{5}\right)^{\frac{1}{2}}, & 5 < b < 10; \\ 0, & b \geqslant 10. \end{cases} \quad (2)$$

然而,该隶属函数没有考虑它所代表的模糊查询谓词的重要程度(假设用户指定该模糊查询谓词的重要程度为"medium").为了使隶属函数式(2)体现模糊查询谓词的重要程度,本文利用了文献[19-20]提出的权重函数对隶属函数式(2)进行转换,即

$$g(w, B(b)) = 1 - w(1 - B(b)). \quad (3)$$

式中:$w$ 代表模糊查询谓词的重要程度.根据知识库文档 NodeImportance.xml,可得式(3)中的 $w = \mu_{\text{importance}}(\text{medium}) = 0.5$.于是,得到转换后的隶属函数为

$$B(b) = \begin{cases} 1, & b \leqslant 5; \\ \dfrac{1}{2} + \left(\dfrac{10-b}{5}\right)^{\frac{1}{2}}, & 5 < b < 10; \\ 0.5, & b \geqslant 10. \end{cases} \quad (4)$$

权重函数的功能是根据用户指定的模糊查询谓词在模糊查询条件中的权重来决定其扩展程度.对于同一个模糊查询谓词,其重要程度越高,则扩展程度越小,在相同阈值下的查询范围相应地越小,这样才能确保转换后得到的精确查询谓词与用户查询意图和偏好最为相关.

根据隶属函数(4),假设用户(或系统)设定的阈值为 0.8,则模糊集 $B$ 的 0.8 截集运算结果为

$$B(b) = \frac{1}{2} + \left(\frac{10-b}{5}\right)^{\frac{1}{2}} = 0.8 \Rightarrow b = 8.2.$$

这样,模糊查询谓词"Buildyear = more or less recent"可转换成精确查询谓词"Buildyear≤8.2".需要说明一点,若给定的阈值 $\alpha$ 低于 $(1-w)$,则 $\alpha$-截集运算结果规定为该模糊查询谓词所对应隶属函数的支集.

### 3.3　含数值区间的模糊查询谓词的转换

考虑模糊查询谓词"SqFt between 130 and 200",使用模糊集 $S$ 表示该模糊查询谓词.为了确

定模糊集 $S$ 的隶属函数,需要在知识库文档 FuzzyTerm.xml 中找出一个模糊词,该模糊词满足如下 2 个条件:1)代表该模糊词的模糊集与模糊查询谓词的隶属函数形状相同;2)代表该模糊词的模糊集的核与模糊查询谓词所指定的数值区间具有相同范围. 通过查找,发现知识库中描述叶节点"SqFt"的模糊词"moderate"满足上述条件(即模糊词"moderate"的隶属函数为梯形隶属函数,并且其核为[130, 200]),因此模糊集 $S$ 的隶属函数可由定义模糊词"moderate"的隶属函数表示,即

$$S(s) = \begin{cases} 0, & s \leqslant 80 \text{ or } s \geqslant 250; \\ \dfrac{s-80}{50}, & 80 < s < 130; \\ 1, & 130 \leqslant s \leqslant 200; \\ \dfrac{250-s}{50}, & 200 < s < 250. \end{cases} \tag{5}$$

如果文档 FuzzyTerm.xml 中不存在满足上述条件的模糊词,需要采用如下方法解决:假设 $[Y_1, Y_2]$ 为模糊查询谓词指定的数值区间,则代表该模糊查询谓词的隶属函数取值分别为

$$\left(\frac{Y_1}{1+\lambda}, Y_1, Y_2, Y_2(1+\lambda)\right).$$

式中:$\lambda(\lambda \in [0, 1])$是阈值,$\lambda$ 值越小,则该模糊查询谓词的模糊集支集与核的距离越小.

假设用户没有指定模糊查询谓词"SqFt between 130 and 200"的重要程度,此时需访问领域知识库文档 NodeRelax.xml 来获取该模糊查询谓词对应叶节点的权重(在 NodeRelax.xml 文档中对应叶节点 SqFt 的权重为"nimp = medium"),利用权重函数 $g(0.5, S(s)) = 1 - 0.5(1 - S(s))$ 对隶属函数(5)进行转换,得到转换后的隶属函数为

$$S(s) = \begin{cases} 0.5, & s \leqslant 80 \text{ or } s \geqslant 250; \\ \dfrac{s-30}{100}, & 80 < s < 130; \\ 1, & 130 \leqslant s \leqslant 200; \\ \dfrac{300-s}{100}, & 200 < s < 250. \end{cases}$$

假设用户(或系统)设定的阈值为 0.8,则模糊集 $S$ 的 0.8 截集运算结果为

$$\begin{cases} S(s) = \dfrac{s-30}{100} = 0.8 \Rightarrow s = 110; \\ S(s) = \dfrac{300-s}{100} = 0.8 \Rightarrow s = 220. \end{cases}$$

这样,模糊查询谓词"SqFt between 130 and 200"可转换成精确谓词"SqFt between 110 and 220".

至此,引言中的模糊查询"$\widetilde{Q}$: -/HouseDB/House[Price at most 300 000][Buildyear = 'Recent'][SqFt between 130 and 200]"可转换成精确查询:"$Q$: -/HouseDB/House [Price ≤ 312 000][Buildyear ≤ 8.2][SqFt between 110 and 220]".

### 3.4 XML 模糊查询谓词转换算法

根据上述转换方法,下面给出通用的 XML 模糊查询谓词转换算法:

**算法 1** XML 模糊查询谓词转换算法

**输入**:模糊查询 $\widetilde{Q}$ 中的模糊查询谓词集合 $\{C_1', C_2', \cdots, C_k'\}$,知识库 KB,阈值 $\lambda$

**输出**:转换后的精确查询谓词集合 $\{C_1, C_2, \cdots, C_k\}$.

1) $Q \leftarrow \varnothing$;

2) for 每一个模糊查询谓词 $\widetilde{C}_j \in \widetilde{Q}$ do

3) MFS MembershipFunctionShape (relaxation direction, satisfaction type);

4) $w_j \leftarrow$ 用户指定(或系统默认)的模糊查询谓词

$\widetilde{C}_i$ 的重要程度;

5) if (扩展程度(relaxation degree) null) then

6) $C_j$ ComputeMembershipFunction1 (relaxdegree, $w_j, \alpha$);

7) else

8) if ($\exists$FuzzyTerm $T = \widetilde{C}_j$ 中的模糊词) (shape($T$) = MFS) then

9) $C_j \leftarrow$ ComputeMembershipFunction2($T, w_j, \alpha$);

10) else

11) if ($\exists$FuzzyTerm $T$) $\wedge$ (core($T$) = $\widetilde{C}_j$ 的数值区间) $\wedge$ (shape($T$) = MFS) then

12) $C_j \leftarrow$ ComputeMembershipFunction2($T, w_j, \alpha$);

13) else

14) $C_j \leftarrow$ ComputeMembershipFunction3 ($f$_interval, $w_j, \alpha$);

15) return $\{C_1, C_2, \cdots, C_k\}$.

**算法 1** 首先识别出模糊查询 $\widetilde{Q}$ 中的每个模糊查询谓词 $\widetilde{C}_i$,并根据 $\widetilde{C}_i$ 对应的模糊关系和叶节点在领域知识库中的扩展方向及叶节点取值对模糊查询谓词的满足方式,确定该模糊查询谓词的隶属函数形状(第 2) ~3)步;然后,根据 $\widetilde{C}_i$ 的权重、对应叶节点在领域知识库中的扩展程度以及相应参数值来构建隶属函数;根据用户或系统提供的阈值,对隶属函数进行 $\alpha$-截集运算,从而将 $\widetilde{C}_j$ 转换成精确查询谓词

$C_j$(第 4) ~14)步);最后,将转换后得到的精确查询谓词$\{C_1, C_2, \cdots, C_k\}$合并返回(第 15 步). 利用转换后的精确查询谓词替换最初的 XPath 查询中相应的模糊查询谓词,最终得到精确的 XPath 查询.

## 4　查询匹配与结果排序方法

### 4.1　查询匹配方法

对于 Xpath 查询匹配方法,本文利用了文献[21]提出的基于 TwigStack 的查询结构匹配方法.

函数:TwigStack ($Q$, $T_1$, $T_2, \cdots, T_n$).

输入:小查询 $Q$,$n$ 个查询结点流 $T_1$, $T_2, \cdots, T_n$ (其中 $n$ 代表查询 $Q$ 的查询结点数量).

输出:按从叶到根有序的小枝查询 $Q$ 的匹配结果.

1) while (not end ($Q$)) //如果查询 $Q$ 没有结束,执行循环;

2) qact←getNext($Q$); //取下一个 XPath 查询条件;

3) if (isRoot (qact) or (not empty (Sparent ($Q_{act}$)))) //如果 qact 是根或//qact 父亲不为空;

4) cleanStack ($S_{qact}$, nextBegin ($T_{qact}$)); //将 qact 待匹配结点流中 begin 值最小的流放入//Sqact;

5) moveStreamToStack ($T_{qact}$, $S_{qact}$, pointer to top (Sparent(qact)));

6) if (! isRoot (qact)) //当 qact 不是根时;

7) cleanStack (Sparent (qact), nextBegin ($T_{qact}$)); //下一个待查询结点入栈;

8) if (isLeaf (qact)) //当 qact 是叶节点时;

9) showSolutions (qact, 1);

10) pop ($S_{qact}$);

11) else advance ($T_{qact}$);

12) mergeAllPathSolutions ().

子函数:cleanStack ($S$, actBegin)

1) while (not empty ($S$) and topEnd ($S$) < actBegin)

2) pop ($S$).

子函数:getNext ($Q$)

1) if (isLeaf ($Q$)) return $Q$;

2) for (every $q_i$ in children ($Q$))

3) $n_i$ = getNext ($q_i$);

4) if($n_i$! = $q_i$) return $n_i$;

5) nmin = $\min arg_{ni}\{$nextBegin ($T_{ni}$)$\}$;

6) nmax = $\max arg_{ni}\{$nextBegin ($T_{ni}$)$\}$;

7) while (nextEnd ($T_q$) < nextBegin ($T_{nmax}$))

8) advance ($T_q$);

9) if (nextBegin ($T_q$) < nextBegin($T_{nmin}$)) return $Q$;

10) else return nmin.

子函数:showSolutions (SN,SP).

/* 假设路径查询 $q$ 中从根到叶的 $n$ 个查询结点的栈编号为 1, 2, …, $n$,假设通过数据 index[1, 2, …, $n$]存储正在生成的匹配结果在各个栈中的位置,即 index[$i$]表示正在生成的匹配结果 $n$ 元组中第 $i$ 个数据结点在第 $i$ 个栈中的位置. 这里,1 代表栈底位置,SN, SP 分别表示当前正在处理栈的编号以及栈中当前正在处理数据结点的位置. */

1) index[SN] = SP;

2) if (SN = = 1) //当前正在处理根栈;

3) output (index[$n$], …, index[1]); //输出来自栈链的已经生成的匹配结果 $n$ 元组

4) else //递归调用

5) for ($i$ =1 to index[SN]. pointer) index[SN]. pointer //表示 index[SN]所指向结点在双亲栈中的位置

6) showSolutions(SN − 1, $i$).

### 4.2　结果排序方法

模糊查询结果排序的基本思想是按结果元素对模糊查询的满足程度进行排序. 结果元素 $t$ 对模糊查询 $\widetilde{Q}$ 的满足程度定义为

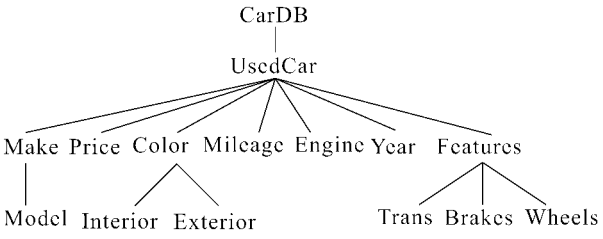
$$\text{Satisfaction}(t,\widetilde{Q}) = \sum_{i=1}^{k} w_i \times \mu_{\widetilde{C}_i}(v_i). \quad (6)$$

式中:$k$ 为 $\widetilde{Q}$ 中包含的模糊查询谓词个数; $\mu_{\widetilde{C}_i}(v_i)$ 代表元素 $t$ 中的叶节点值 $v_i$ 对模糊查询谓词 $\widetilde{C}_i$ 的隶属度; $w_i$ 代表模糊查询谓词 $\widetilde{C}_i$ 在模糊查询条件中的重要程度. 利用式(6)计算出的满足程度越高,表示该结果元素的排序分值越大,因此其排序位置就越靠前.

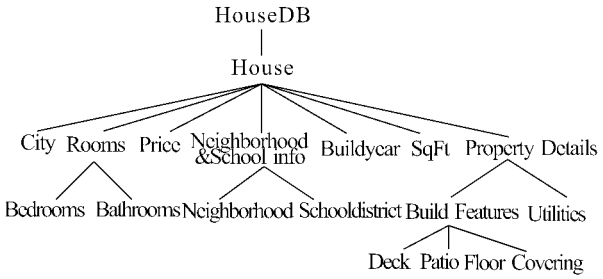
## 5　实验测试与结果分析

### 5.1　实验环境和测试数据

所有实验都是在配置为 Pentium (R) Dual-Core CPU E6500 @2.93GHz,2GB 内存,160GB、7200rpm 硬盘和 Microsoft Windows XP 操作系统下进行,使用 SAX2 解析 XML 文档,用 Java 和 eclipse 集成环境开发实验系统. 测试文档由 IBM XML data generator[22] 产生,生成 2 个 XML 数据集,Car DB 和 House DB 的文档结构如图 1 所示).




(a) CarDB




(b) HouseDB

**图1 CarDB 和 HouseDB 的文档结构图**

**Fig. 1 The schema of CarDB and House DB**

1) CarDB. xml 数据集:从 http://autos. yahoo. com/网站随机抽取了 200 000 条二手车信息,利用 IBM XML data generator 生成 CarDB. xml 数据集,该数据集包括 200 000 个 UsedCar 元素.

2) HouseDB 数据集:从 http://homes. yahoo. com/网站随机抽取了 250 000 条房产信息记录,利用 IBM XML data generator 生成 HouseDB. xml 数据集,该数据集包括 250 000 个 House 元素.

### 5.2 测试模糊查询效果

该实验以用户调查方式测试 XML 模糊查询方法的查全率(recall)和准确率(precision). 查全率是指查询结果中的相关元素数占文档中相关元素总数的百分比. 准确率是指查询结果中的相关元素数占查询结果中总元素数的百分比. 邀请了 10 位测试者,每位测试者根据自己的需求和偏好,对 CarDB 和 HouseDB 分别提出各 1 条模糊查询进行效果测试. 用户提出的测试模糊查询如表 1 所示.

**表1 HouseDB 和 CarDB 上的模糊测试查询**

**Table 1 Fuzzy test queries over HouseDB and CarDB**

| ID | Fuzzy XML queries over HouseDB | Fuzzy XML queries over CarDB |
|---|---|---|
| Q1 | //House[Price at most 300 000] [SqFt between 80 and 130] [City = 'Bellevue'] | //UsedCar[Price between 10 000 and 20 000] Make/[Model = 'Camry'][Mileage at most 50 000] |
| Q2 | //House[Price between 700 000 and 800 000] [View = 'Waterview'][SqFt at least 300] | //UsedCar[Make = 'Ford'][Price at most 10 000] [Year = 'more or less recent'] |
| Q3 | //House[City = 'Redmond'][Price = 'medium'] [SqFt between 150 and 200] | //UsedCar[Price = 'moderate'][Year = 'Recent'] [Mileage at most 30 000] |
| Q4 | //House[Price = 'moderate'][SqFt = 'more or less large'] [BuildYear = 'more or less recent'] | //UsedCar[Make = 'BMW'][Price close to 30 000] Color/[Exterior = 'white'] |
| Q5 | //House[City = 'Seattle'][SqFt = 'large'][Price at most 700 000] | //UsedCar[Make = 'Nissian'][Price between 5 000 and 10 000][Mileage = 'more or less little'] |
| Q6 | //House[City = 'Seattle'][Price = 'more or less low'] Rooms/[Bedrooms = 3] | //UsedCar[Make = 'Dodge'][Price at most 15 000] [Year = 'moderate'] |
| Q7 | //House[SqFt = moderate][Bedrooms > =5] Neighborhood&School info/[Neighborhood = 'shoreline'] | //UsedCar[Make = 'Honda'][Price = 'very low'] |
| Q8 | //House[Price = 'moderate'][SqFt between 200 and 250] Neighborhood&School info/[Schooldistrict = 'Central Seattle'] | //UsedCar[Make = 'Ferrari'][Price between 25 000 and 30 000][Year = 'more or less Recent'] |
| Q9 | //House[City = 'Bellevue'][Bedrooms between 4 and 6] [SqFt = 'moderate'][Buildyear = 'Recent'] | //UsedCar [Make = 'Ford'] [Price = 'moderate'] [Mileage = 'little'][Year = 'Recent'] |
| Q10 | //House[City = Seattle][Price between 300 000 and 500 000][SqFt = 'more or less large'] | //UsedCar[Make = 'Tucson'] [Price = 'more or less low'][Engine =3.2] |

对于 CarDB 和 HouseDB 上的每个测试查询 $Q_i$,从原始数据集中为其生成一个小型数据集 $H_i$,$H_i$ 中包含了 60 个与该查询相关的和不相关的适量元素. 然后,将所有测试查询和相应数据集提供给每个测试者,让这些测试者从 $H_i$ 中标出与查询 $Q_i$ 相关的所有元素. 最后将系统返回的结果与用户标注的结果进行对比来测试本文方法的查全率和准确率.

本文的测试方法是从 0 ~ 100% 之间选取 10 个不同的查全率水平,然后计算对应的准确率(通过调整阈值,可以获得不同的查全率). 也就是说,对于由本文方法返回的模糊查询结果,用户逐个检查结果元素是否为用户标注的相关元素. 如果是相关元素,则对应该元素的准确率为 100%,否则为 0,最终该测试查询的准确率为所有结果元素对应准确率的平均值. 图 2 给出了从 0 ~ 100% 之间选取的 10 个不同查全率水平下获得的对应准确率. 图中显示的评估值是分别在 CarDB 和 HouseDB 上的 10 个测试查询的准确率的平均值.

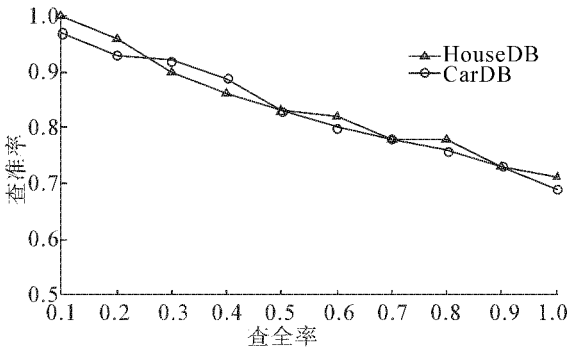



**图 2　HouseDB 和 CarDB 上的查全率和准确率**

**Fig. 2　Precision and recall over HouseDB and CarDB**

从图 2 中可以看出,高准确率的获得是以查全率的牺牲为代价的,反之亦然. 但该实验主要反映的是,当本文方法的查全率为 100% 时,HouseDB 和 CarDB 上的平均准确率分别为 71% 和 69%;当查全率为 90% 时,HouseDB 和 CarDB 上的平均准确率均为 73%;当查全率为 80% 时,HouseDB 和 CarDB 上的平均准确率分别为 78% 和 76%. 由于用户在提交查询时可以根据个人偏好为查询谓词指定重要程度,因此返回的查询结果也就更能贴近用户需求和偏好. 也就是说,本文提出的模糊查询方法能够同时具有较高的查全率和准确率.

### 5.3　测试结果排序质量

该实验目的是测试结果排序方法的排序质量. 为了观察不同测试查询下排序方法的质量,让测试者从他们所标注的相关元素中选出最相关的前 10 个元素. 然后,分别在 CarDB 和 HouseDB 上执行排序方法为每条测试查询返回前 10 个元素. 本文采用式(7)所示的排序准确率评价排序质量:

$$R = \sum_i \frac{r_i}{2^{\left(\frac{i-1}{9}\right)}}. \tag{7}$$

式中:$r_i$ 代表排序函数返回的元素列表中第 $i$ 个元素的得分(若该元素被标记为最相关的,则 $r_i$ 取值为 1,否则 $r_i$ 取值为 0). 可见,在 $R$ 度量方法下,如果本文排序方法返回的相关元素在列表中所处的位置越靠前,则它对 $R$ 值的影响就越大. 对于每个测试查询,取 10 个测试者根据式(7)计算得到的平均排序准确率作为该查询结果的排序准确率. 图 3 给出了在 CarDB 和 HouseDB 上对应于每个测试查询的查询结果排序准确率. 实验结果表明,本文提出的排序方法在 2 个数据集上的平均排序准确率分别达到 0.77 和 0.80.

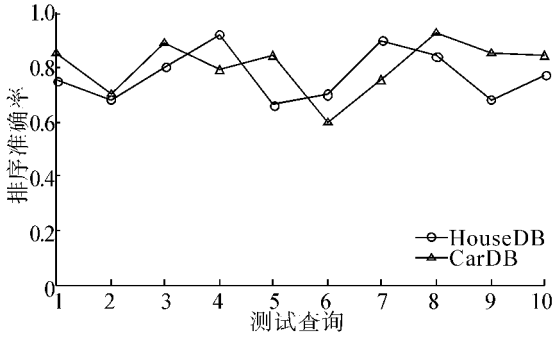



**图 3　HouseDB 和 CarDB 上的排序准确率**

**Fig. 3　Ranking quality over HouseDB and CarDB**

### 5.4　响应时间测试

本文方法的执行过程主要包括模糊查询转换和结果排序处理 2 个阶段. 第 1 阶段完成模糊查询谓词到精确查询谓词的转换,其时间复杂度为 $O(mk)$,其中 $m$ 代表模糊查询条件中的模糊查询谓词个数,$k$ 代表每个模糊查询谓词所涉及知识库文档中包含的不同元素个数的平均数. 第 2 阶段包括结果元素的排序分值计算和排序处理 2 个部分,结果元素的排序分值计算时间复杂度为 $O(n)$,$n$ 代表结果元素个数;排序处理的时间复杂度为 $O(n\log n)$,$n$ 代表结果元素个数. 因此,第 2 阶段的总体时间复杂度为 $O(n\log n)$. 图 4 给出了本文方法在 2 个数据集上不同结果元素个数下的响应时间. 从图中可以看出,本文的响应时间随着查询结果元素个数的增加几乎呈线性增长趋势.

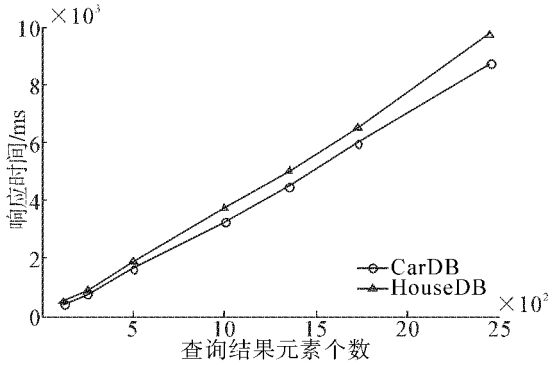



**图4 不同查询结果元素个数下的响应时间**

**Fig. 4 Response time of different number of elements in query results**

## 6 结束语

本文以模糊集理论为基础,提出了一种利用隶属函数、领域知识和模糊集的α-截集运算相结合实现XML模糊查询和结果排序的方法.该方法不需要改变传统XDBMS的结构和XML查询语言就能够实现XML数据的模糊查询转换和结果排序处理,从而在很大程度上提高了用户与系统之间的交互能力.实验结果表明,本文方法能够同时具有较高的查询准确率和查全率,具有较好的查询结果排序质量,能够较好地满足用户需求和偏好.未来工作将从查询结果排序和知识库的自学习方面作进一步的完善.

## 参考文献:

[1] BRAY T. Extensible markup language (XML) 1.0 [CP/OL]. [2011-03-18]. http://www.w3.org/TR/REC-xml/.

[2] TAHANI V. A comceptual framework for fuzzy querying processing: a step toward very intelligent databases systems [J]. Information Processing Management, 1997, 13: 289-303.

[3] ZADEH L A. Fuzzy sets [J]. Information and Control, 1965, 8(3): 338-353.

[4] BOSC P, PIVERT O. SQLf: a relational database language for fuzzy querying [J]. IEEE Transactions on Fuzzy Systems, 1995, 3(1): 1-17.

[5] BOSC P, PIVERT O. Extending SQL retrieval features for the handing of flexible queries[C]//Proceedings of International Conference on Fuzzy Information Engineering. New-York, USA, 1997: 233-251.

[6] BOSC P, GALIBOURG M, HAMON G. Fuzzy querying with SQL: extensions and implementation aspects [J]. Fuzzy Sets Systems, 1988, 28: 333-349.

[7] NAKAJIMA H, SOGOH T, ARAO M. Fuzzy database language and library: fuzzy extension to SQL[C]//Proceedings of the 1993 International Conference on Fatigue Science. Washington DC, USA, 1993: 477-482.

[8] CHEN S M, JONG W T. Fuzzy query translation for relational database systems [J]. IEEE Transcations Systems, Man, and Cybernetics, Part B: Cybernetics, 1997, 27 (4): 714-721.

[9] MA Z M, YAN Li. Generalization of strategies for fuzzy query translation in classical relational databases [J]. Information and Software Technology, 2007, 49(2): 172-180.

[10] 李言辉, 徐宝文, 陆建江. 一般术语公理下的模糊描述逻辑FALCN推理[J]. 软件学报, 2008, 19(3): 594-604.
LI Yanhui, XU Baowen, LU Jianjiang. Reasoning with general terminological axioms in fuzzy description logic FALCN[J]. Journal of Software, 2008, 19(3): 594-604.

[11] 康达周, 徐宝文, 陆建江, 等. 支持模糊隶属度比较的扩展模糊描述逻辑[J]. 软件学报, 2008, 19(10): 2498-2507.
KAN Dazhou, XU Baowen, LU Jianjiang, et al. Extended fuzzy description logics with comparisons between fuzzy membership degrees [J]. Journal of Software, 2008, 19 (10): 2498-2507.

[12] KANZA Y, SAGIV Y. Flexible queries over semi-structured data [C]//Proceedings of the 20th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. Scottsdale, USA, 2001: 40-51.

[13] 王真星, 顾宁, 施伯乐. 基于本体的半结构化数据的柔性查询[J]. 计算机研究与发展, 2003, 40 (11): 1571-1578.
WANG Zhenxing, GU Ning, SHI Bole. An ontology-based flexible query method for semistructured data[J]. Journal of Computer Research and Development, 2003, 40 (11): 1571-1578.

[14] POLYZOTIS N, GAROFALAKIS M, IOANNIDIS Y. Approximate XML query answers [C]//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. Paris, France, 2004: 263-274.

[15] MANDREOLI F, MARTOGLIA R, TIBERIO P. Approximate query answering for a heterogeneous XML document base[C]//Proceedings of the International Conference on Web Information Systems Engineering. Brisbane, Australia, 2004: 337-351.

[16] SPIEGEL J, PONTIKAKIS E D, BUDALAKOTI S, et al. AQAX: a system for approximate XML query answers [C]//Proceedings of the 32nd International Conference on Very Large Data Bases. Seoul, Korea, 2006: 1159-1162.

[17] 衡星辰, 覃征, 邵利平, 等. 基于两阶段查询重写的XML近似查询算法[J]. 电子学报, 2007, 35 (7): 1271-1278.
HENG Xingchen, QIN Zheng, SHAO Liping, et al. Two-phase query rewriting based approximate XML query algorithm [J]. Chinese Journal of Electronics, 2007, 35 (7): 1271-1278.

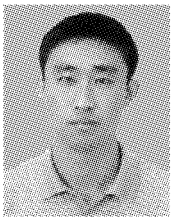
[18] MA Z M, MENG X F. A knowledge-based approach for answering fuzzy queries over relational databases [C]// Proceedings of the 12th International Conference on Knowledge-based and Intelligent Information and Engineering Systems. Zagreb, Croatia, 2008: 623-630.

[19] LARSEN H L. An approach to flexible information access systems using soft computing [C]//Proceedings of the 32nd Annual International Conference on System Sciences. Maui, USA, 1999: 6042.

[20] YAGER R R. Information fusion and weighted median aggregation[C]//Proceedings of the 5th International Work-

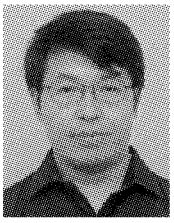
shop on Current Issues in Fuzzy Technologies. Trento, Italy, 1995: 209-219.
[21] BRUNO N, KOUDAS N, SRIVASTAVA D. Holistic twig joins: optimal XML pattern matching[C]//Proceedings of the 2002 ACM SIGMOD Conference on Management of Data. Madison, USA, 2002: 310-321.
[22] IBM Corporation. XML data generator [EB/OL]. [2011-10-15]. http://www.alphaworks.ibm.com/tech/xmlgenerator.

**作者简介:**



孟祥福,男,1981 年生,副教授,博士,主要研究方向为 Web 数据库与 XML 数据柔性查询.

张霄雁,女,1983 年生,助教,主要研究方向为 XML 数据查询结果 Top-k 排序.



马宗民,男,1965 年生,教授,博士生导师,IEEE 高级会员,主要研究方向为智能数据与知识工程,担任多个国际期刊的副主编,承担科研项目 20 余项,发表学术论文 200 余篇,其中被 SCI 检索 40 余篇,被 EI 检索 150 余篇,出版专著 10 余部.