

# 时态 XML 索引技术

叶小平 陈铠原 汤庸 汤娜 胡苏

(中山大学计算机科学系 广州 510275)

**摘 要** 首先通过讨论时态 XML 查询数据模型 TXQDM,提出了基于结点有效时间的前缀编码方案.以此为基础,引入 TXQDM 结点间的基于时态连通的等价关系和基于时态包含的拟序关系,建立了时态 XML 索引数据模型 TXIDM,该模型的基本特征是具有二重嵌套的索引框架,适合于 TXQDM 这种不规则的具有较大随意性的树形结构情形.其次,在 TXIDM 框架内,讨论了相应时态查询算法,其中包括基于时态的路径查询和值查询,同时,还讨论了时态索引更新算法,其中包括插入和修改算法.最后,对于文中提出的模型 TXIDM 和时态索引操作算法进行了性能分析且设计了相应模拟实验.实验结果表明,基于 TXIDM 的时态查询与更新算法是可行的和有效的.

**关键词** 时态关系前缀编码;时态 XML 索引数据模型;时态连通与包含关系;时态查询和索引更新  
中图法分类号 TP311

## Technology on Temporal XML Indexing

YE Xiao-Ping CHEN Kai-Yuan TANG Yong TANG Na HU Su

(Department of Computer Science, Sun Yat-Sen University, Guangzhou 510275)

**Abstract** This paper is devoted to the technique and implementation of temporal XML indexing. Firstly, this paper proposes the schema of prefix-code based on the valid time of nodes, and introduces the temporal equivalence and temporal quasi-order relationships which result from the temporal connection and inclusion on the set of the valid time periods of all temporal nodes. Using these relationships, the paper builds the index model of temporal XML data, that is TXIDM, and the TXIDM has a characteristic of the re-nesting structure which may be more suitable to the random tree-type structure of the temporal XML data. Secondly, the paper discusses the index algorithm on temporal query (values query and path query) and temporal update (inserting and modification) and these two may be described uniformly as they are all based on the properties of the relationships especially the quasi-order. Finally, the paper completes the analysis of capability and the designing of the experimental simulation for the model and algorithm, and the experiment results suggest that the operation on temporal update and query are feasible and efficient.

**Keywords** prefix-code based on valid time; temporal XML index data model; temporal connected and included relationships; temporal query and update

收稿日期:2005-05-15;修改稿收到日期:2007-01-16. 本课题得到国家自然科学基金(60373081,60673135)和广东省自然科学基金重点项目(04105503)、广东省自然科学基金(5003348)资助. 叶小平,男,1955年生,博士,副教授,现主要从事数据库技术、时态信息处理技术和知识发现等方面的研究. E-mail: mcsyxp@mail.sysu.edu.cn. 陈铠原,男,1983年生,研究生,主要研究方向为时态数据库、多 Agent 系统和冲突分析. 汤庸(通信作者),男,1964年生,博士,教授,博士生导师,现主要从事数据库、知识库和协同软件等方面的研究. E-mail: issty@mail.sysu.edu.cn. 汤娜,女,1975年生,博士研究生,讲师,主要研究方向为数据库与 XML 技术. 胡苏,女,1983年生,硕士研究生,主要研究方向为时态数据库、时态数据挖掘.

## 1 引言

随着 Internet 技术的发展,XML 作为一种数据发布语言,已经成为计算机行业标准的数据交换格式,基于 XML 数据的查询应用引起了学术界和商业应用界的高度关注.对于 XML 数据的管理,当前存在着两条实现途径,一是基于使能(enabled)XML 数据库技术,一是使用原生(native)XML 数据库技术.前者通过各种方式将 XML 文档转换为关系数据库形式进行相应操作,后者直接对原始 XML 文档进行管理和查询.从原生 XML 数据库技术的观点出发,针对 XML 文档具有的半结构化特征,人们提出了基于 XPath 的 XML 查询数据模型和相应查询语言<sup>①</sup>.由于原生 XML 数据库中还没有像关系数据库中那样成熟有效的查询技术,为了支持相应数据查询,通常需要借助于索引技术,为此,人们提出了各种索引方案,其中基于编码的索引技术的研究和应用比较活跃.文献[1-2]具体讨论了基于区间编码的技术,文献[3]讨论了基于后缀编码的索引技术.这些工作都为时态 XML 索引技术讨论提供了参考和借鉴.时态 XML 文档是具有时态标签的 XML 文档.常用时态标签可分为有效时间和事务时间,它们在 XML 文档中以属性方式出现.由于时态属性不同于一般属性,时态 XML 文档相对通常 XML 文档具有许多基本课题需要讨论,这和常规数据库(关系数据库和对象数据库)与时态数据库的情形类似.例如,关于时态的操作代数不同于一般 XML 数据操作代数<sup>[4]</sup>;时态属性值中的有效时间变量需要特别处理<sup>[5]</sup>;在时态查询模型中,表示内容的文本结点即叶结点可以有兄弟结点<sup>[6]</sup>等,这些都使得非时态 XML 查询及索引技术不能直接套用过来.文献[6]讨论了时态 XML 的查询问题,但没有涉及时态 XML 数据模型的一般讨论.文献[7-9]讨论了基于 XPath 的时态 XML 的查询模型 TXpath,其中文献[7]将 Xpath 映射为有序树,由于认为只有元素结点可以排序,属性及文本结点不能排序,该有序树中所有结点都为 XML 文档元素类结点,而属性、文本结点隐藏在一个信息获取函数中.此时,将有效时间作为结点的一个属性,一个结点可以拥有多个互不相交的有效时间区间,且它们完全包含在父结点的有效时间区间内.文献[8-9]提出的 TXPath 数据模型允许出现图,文献[9]讨论了基于关系数据库的时态 XML 存储问题,文献[10]基于时态 XML 的法律

文档的管理系统原型,建立特定的时态 XML 模型.文献[7-10]实际上给出了 Xpath 模型的一个时态扩展,但没有讨论相应查询语言扩展,也没有涉及时态更新. Vaisman 等<sup>②</sup>建立的 TXPath 数据模型与 W3C 基本一致,但通过对 TXPath 语言的语法语义重新定义扩展了查询语言.其中,在时态查询方面定义了值查询及返回值形式,定义返回结果是一个结点时间区间二元组集合.另外还描述了更新操作几种形式,但没有详细阐述.文献[11]在此基础上提出了一种时态 XML 索引模型.其基本思想是将路径信息与时态信息相结合,是一种以结构索引为主的方法,该方法中使用的数据结构复杂,更新操作难以实现,因此没有讨论索引的更新操作.文献[12]在事务时间方面对 Xpath 模型进行了扩展,通过将事务时间作为 XML 文档本身隐含的时态信息,把文档的事务时间分成“修改”和“访问”两类,并在两个修改时间之间访问时对文档内容进行推断.推断分成两种.一种认为两次修改之间文档内容不变,一种认为文档内容发生改变,由于将事务时间考虑为时间点,与一般的 Xpath 时态扩展存在差异.

据我们掌握资料,目前国内外还没有关于时态 XML 索引技术研究及实现的较为深入的讨论.本文首先借鉴非时态框架下的工作,讨论了一般时态 XML 查询数据模型 TXQDM,引入 TXQDM 结点间时态连通概念,建立了结点间一种等价关系.通过将结点分类和研究等价类中结点间时态包含产生的拟序(quasi-order)结构,建立了时态 XML 索引数据模型 TXIDM. TXIDM 和其弱化形式 STXIDM 可以完成关于基于时态的路径与值查询,同时也能有效实现时态索引的更新操作.本文索引技术关键在于先对 TXQML 结点通过基于时态连通的等价关系进行分类,再对每个等价类的元素通过基于时态包含的拟序关系引入“子”结构,从而在不同层面上建立了索引,避免了遍历.其次,本文设计了拟序关系集基本算法,实现了时态索引的查询和更新(插入与删除).最后,本文对提出的各种算法进行了必要性性能分析,同时设计了相应的模拟实验.算法的性能

① World Wide Web Consortium. XQuery1.0 and XPath 2.0 Data Model. W3c Working Draft, 23 July 2004. <http://www.w3.org/tr/xpath-datamodel>; W3C XQuery 1.0 and XPath 2.0 Functions and Operators. <http://www.w3.org/TR/xquery-operators/>

② Alejandro Vaisman, Alberto O. Mendelzon, Enrique Molinari, Pablo Tome. Temporal XML: Data Model, Query Language and Implementation (2004). <http://www.cs.toronto.edu/~avaisman/papers.html>

分析和实验的结果表明,本文提出时态索引技术有着良好的查询与更新效率,具有实际应用的有效性和可行性.

本文第 2 节讨论了时态 XML 查询数据模型 TXQDM 和基于结点有效时间的前缀编码;第 3 节通过引入时态连通关系和时态拟序关系,建立了时态 XML 索引数据模型 TXIDM 和 STXIDM,两者在原理上相通,分别适用于时态 XML 的路径查询和值查询;第 4 节研究了基于时态索引的 XML 查询方法,其中包括路径查询和值查询;第 5 节研究了时态索引的更新算法,而 XML 索引的更新一直是人们关注的基本问题之一;第 6 节设计了相应的实验来验证本文提出的模型与技术,并对其性能进行了必要评估.

## 2 时态 XML 查询模型与时态前缀编码

XML 文档可以建模为一个基于 XPath 的有序且具有边标记的结点层次图形结构.为了将 XML 文档本身做为“数据”进行查询,人们以这种结点层次结构为基础,建立起相应 XML 查询数据模型<sup>[1]</sup>.时态 XML 查询数据模型则是在常规模型上添加时态信息.

### 2.1 时态 XML 文档与时态 XML 查询数据模型

XML 文档作为一种标记语言,其中元素生命周期即有效时间也能以某种标记形式出现.如果将文档中元素的有效时间以属性形式表现在元素名称标记当中,则可称其为时态 XML 文档.一个时态 XML 文档的片断如下所示:

```
<employees VTs="2004-01-01" VTe="now">
  <employee VTs="2004-01-01" VTe="now">
    <name VTs="2004-01-01" VTe="now">Black</name>
    <title VTs="2004-01-01" VTe="2005-12-31">Engineer</title>
    <title VTs="2006-01-01" VTe="now">Sr. Engineer</title>
    <dept VTs="2004-01-01" VTe="now">IT</dept>
    <salary VTs="2004-01-01" VTe="2005-12-31">7500</salary t>
    <salary VTs="2006-01-01" VTe="now">8500</salary t>
  </employee>
</employees>
```

这里,有效时间期间表示为 $[VTs, VTe]$ ,其中 VTs 和 VTe 分别表示起始和为终止时刻.

按照 W3C 发布的基于 XPath 的 XML 数据模型工作草案<sup>[1]</sup>,XML 数据模型是一种图形结构.在这种模型中,数据值被表示为包含零个或多个项的序列,其中项可以是原子类型值或结点.所有结点之

间存在称为文档顺序的内在顺序.XML 树形结构中主要的结点是根(文档)结点、元素结点、属性结点和文本结点,而结点间相互关系分为两类,由“祖先/子孙”、“双亲/子女”关系组成的包含关系和由“之前/之后”、“左兄弟/右兄弟”关系组成的文档位置关系.

设  $D$  是一个 XML 查询数据模型, $V(D)$  为  $D$  中前述四类结点集合,其中根结点记为  $r$ ,元素、属性和文本结点集合分别记为  $Vele(D)$ 、 $Vatt(D)$  和  $Vtex(D)$ , $V(D) = \{r\} \cup Vele(D) \cup Vatt(D) \cup Vtex(D)$ .

下面给出时态 XML 查询数据模型的形式化定义,这可看作是相关定义的扩充.

**定义 1**(时态 XML 查询模型 TXQDM). 一个时态 XML 查询数据模型 TXQDM 是一个有序、带有时态边标记的树形结构,而该树形结构可形式化为  $TD = \langle V(TD), E(TD), label, rank \rangle$ ,其中

(1)  $V(D)$  是所有结点的集合,  $E(D)$  是所有边的集合.

(2)  $label = \{label_E, label_{Vele}, label_{Vatt}, label_{Vtex}\}$  为映射集合,这里

① 映射  $label_E: E(D) \rightarrow \{((u, v), VT(v))\}$ , 其中,  $u, v$  分别是有向边  $(u, v)$  的输出和输入结点,  $VT(v) = [VTs(v), VTe(v)]$  是结点  $v$  的有效时间区间.  $VTs(v)$  和  $VTe(v)$  分别表示结点  $v$  有效时间期间  $VT(v)$  的起始时刻和终止时刻.

② 映射  $label_{Vele}: Vele(D) \rightarrow \langle string, string, string \rangle$ , 给每个元素结点赋予一个三元字符串组,分别表示该元素结点的对象标识 ID、元素标记名和结点类型. 结点类型的值是“EE”,“ET”,“EM”或“EN”,它们分别表示的内容是子元素、文本、子元素和文本混合或内容为空的元素结点.

③ 映射  $label_{Vatt}: Vatt(D) \rightarrow \langle string, string, string, string \rangle$ , 给每个属性结点赋予一个四元字符串组,分别表示该属性结点的对象标识 ID、属性名、属性值(字符串值)和属性值的类型.

④ 映射  $label_{Vtex}: Vtex(D) \rightarrow \langle string, string, string \rangle$ , 给每个文本结点赋予一个三元字符串组,分别表示该文本结点的对象标识 ID、文本结点的内容(字符串值)和内容的类型.

(3) 映射  $rank: V \rightarrow integer$ , 给每一个结点赋予一个整数,表示该结点的文档序号.结点的文档序号在一个文档内唯一,它反映在先序遍历文档树时该结点的出现次序.对于一个元素结点而言,规定它所拥有属性结点文档序号必须“小于”其内容结点(子

元素结点和文本结点)文档序号. 由于元素结点中所包含各个属性结点之间不区分次序, 因此, 对于某元素结点所包含的各属性结点的文档序号的先后无关紧要.

与通常时态数据模型中情形类似, 对象间时态约束是展开讨论的基础. 设  $TD$  是前述时态 XML 查询模型框架内的一个实例, 则  $TD$  结点需要满足如下时态约束条件:

① 根元素在同一时刻只能包含一个元素. 设  $r$  是  $TD$  的根,  $c_i$  和  $c_j$  是  $r$  的子结点, 则在时态边  $((r, c_i), VT(c_i))$  和  $((r, c_j), VT(c_j))$  中,  $VT(c_i) \cap VT(c_j) = \emptyset (i \neq j, 1 \leq i, j \leq n)$ .

② 元素、属性和文本结点的存在依赖于它们的父结点  $\forall v \in V(TD) - \{r\}$ ,  $t_i$  和  $t'_j (1 \leq i \leq m, 1 \leq j \leq n)$  分别是  $v$  的入边和出边的有效时间标签, 则  $\cup t'_j \subseteq \cup t_i$ .

③ 元素或属性结点不能同时拥有两个文本子结点. 设  $v \in V(TD) - \{r\}$  是一个元素或属性结点,  $c_i$  和  $c_j$  分别为  $v$  的文本子结点, 如果  $((v, VT(c_i)), t_i)$  和  $((v, c'_j), VT(c_j))$  是相应时态边, 则  $VT(c_i) \cap VT(c_j) = \emptyset$ .

本文所述时态 XML 查询模型均假设满足上述条件.

## 2.2 基于时间期间的结点前缀编码

结点编码是进行查询的前提, 这种编码需要既含有结点本身的“值”信息, 又能表示结点间相互关系的结构信息.

**定义 2**(时态结点编码). 当查询模型是树形结构时, 基于前缀的结点编码可以采用递归定义. 对一个结点  $u$  而言,  $VT(u)$  表示以结点  $u$  “入”边表示的有效时间.

① 设  $r$  是根结点, 则  $r$  的编码  $code(r) = (VT(r), dep(r), ID(r))$

② 设  $v$  是  $r$  的子结点, 则  $v$  的编码  $code(v) = code(r). (VT(v), dep(v), ID(v))$

③ 设第  $k$  层结点  $v$  的编码为  $code(v)$ , 则第  $k+1$  层  $v$  的子结点  $u$  的编码  $code(u) = code(v). (VT(u), dep(u), ID(u)). (VT(u), dep(u), ID(u))$  称为  $u$  的自然(编)码,  $code(v)$  称为  $u$  的前缀(编)码.

在上述定义中,  $VT(v)$  是结点  $v$  的有效时间标签,  $dep(v)$  是  $v$  的深度即“自顶向下”所在的层次数,  $ID(v)$  包含  $v$  在“ $dep(v)$ ”层“自左向右”所在的位置编号. 由上述编码概念, 可得如下定理 1.

**定理 1.**  $v$  是  $u$  父结点的充要条件是  $u$  的前缀码等于  $v$  的编码;  $v$  是  $u$  祖先结点的充要条件是  $u$  的前缀码包含  $v$  的编码.

定理 1 说明上述前缀编码中的时态部分可以确定结点间通常的包含关系, 而非时态部分“ $ID(u)$ ”由于是字典顺序, 可以确定结点间的文档位置关系.

在编码过程中, 每个结点只需要时间复杂度为  $O(1)$  的操作(包括记录父结点指针、所在层次、有效时间区间等信息), 由此可知整个前缀码编码过程需要的时间复杂度为  $O(n)$ .

## 3 时态 XML 索引数据模型

时态索引数据模型建立在时间期间集合上的连通关系和包含关系基础之上.

**定义 3**(时态连通和包含关系). 若 TXQDM 中两结点  $u$  和  $v$ ,  $VT(u) \cap VT(v) \neq \emptyset$ , 则称  $u$  和  $v$  之间存在一条时态连接边, 记为  $\langle u, v \rangle$ . 结点  $u$  和  $v$  存在时态连通关系  $\sim: u \sim v$ , 是指存在时间边  $\langle u, u_1 \rangle, \langle u_1, u_2 \rangle \dots, \langle u_{n-1}, u_n \rangle, \langle u_n, v \rangle$ . 可以证明, 时态连通关系“ $\sim$ ”是  $V(D)$  上的等价关系.

若 TXQDM 中两结点  $u$  和  $v$ ,  $VT(u) \subseteq VT(v)$ , 则称  $u$  和  $v$  之间存在时态包含关系  $R$ , 记为  $uRv$ . 可以证明, 时态包含关系“ $R$ ”是  $V(D)$  上满足自反性和传递性的拟序关系(quasi-order). 结点  $b$  的拟序关系集  $R(b) = R_+(b) \cup R_-(b)$ , 其中  $R_+(b) = \{a | VT(b) \subseteq VT(a)\}$ ,  $R_-(b) = \{a | VT(a) \subseteq VT(b)\}$ , 分别称其为结点  $b$  的后向和前向拟序关系集.

在 TXQDM 的同一结点层, 构造时态连通等价类, 得到该层结点集合上的一个划分. 记该结点层上所有等价类集合为  $V([TD])$ , 将其看作由时态查询模型 TXQDM 诱导的新的“结点”集合, 则对每个  $[u] \in V([TD])$ , 定义“结点” $[u]$  的“入边”为时态边, 其有效时间标签为  $VT([u]) = \cup VT(u_i)$ , 其中  $u_i \in [u]$ . 由时态连通概念可得定理 2.

**定理 2.**

① 若  $[u] \neq [v]$ , 则  $VT([u]) \cap VT([v]) = \emptyset$ ;

② 若  $VT(u) \subseteq VT(v)$ , 则  $VT([u]) \subseteq VT([v])$ .

在每个时态连通等价类  $[u]$  中, 引入时态包含关系“ $R$ ”. 对于  $v \in [u]$ , 后向关系类  $R_+(v)$  和前向关系类  $R_-(v)$  分别定义为  $R_+(v) = \{v' | vRv', v' \in [u]\}$  和  $R_-(v) = \{v' | v'Rv, v' \in [u]\}$ .

**定义 4**(时态 XML 索引模型). 给定 TXQDM,

满足如下条件的树形结构称为 TXQDM 对应的时态 XML 索引模型 TXIDM: ① TXIDM 的层数与 TXQDM 层数一致, TXIDM 每层上的结点为 TXQDM 相应层上结点的时态连通等价类组成. ② 设  $[u]$  和  $[v]$  分别属于 TXIDM 中自顶向下的第  $k$  层和第  $k+1$  层, 则  $[v]$  是  $[u]$  的双亲结点当且仅当  $VT([u]) \subseteq VT([v])$ . ③ TXIDM 中每个结点都按照时态包含关系建立了拟序(子)结构.

“①”说明 TXIDM 与 TXQDM 具有“相似”的结构; “②”说明 TXIDM 每层的结点集合是 TXQDM 相应层结点集合的“划分”, 表现 TXIDM 的索引特征; “③”说明 TXIDM 中每个结点都还存在某种结构, 这可以看作对 TXQDM 中结点类集合进行深入分析, 本文后面的分析和实验证明, 这将有效提高查询效率.

如果只在给定 TXQDM 的所有叶结点集合上建立基于时态连通的等价划分, 再在每个等价类上建立基于时态包含的拟序关系覆盖, 得到的模型称为简化的时态 XML 索引模型, 记为 STXIDM.

## 4 时态 XML 查询

时态 XML 查询分为基于时态的路径查询和值查询. TXIDM 适用于路径查询, STXIDM 适用于值查询.

### 4.1 时态路径查询

在 XML 路径查询中, 通常有两种形式的路径表达式. 一种是形如“ $//A \dots$ ”的绝对路径, 它表示从根结点  $r$  开始, 查询所有以结点  $A$  为后裔结点的路径; 另一种是形如“ $\dots A // B \dots$ ”的相对路径, 它表示以当前结点  $A$  开始, 查询所有以  $B$  为后裔结点的路径. 由于绝对路径可转化为相对路径, 下面讨论对相对路径查询.

基于时态的路径查询  $Q_p = \dots A // B \dots$  可分为两种类型.

**类型 1.** 给定查询  $Q_p$  的有效时间  $VT(Q_p)$ , 查询所有有效时间包含  $VT(Q_p)$  的路径“ $\dots A // B \dots$ ”.

**类型 2.** 分别给定  $VT(Q_p(A))$  和  $VT(Q_p(B))$ , 查询满足  $VT(Q_p(A)) \subseteq VT(A')$  和  $VT(Q_p(B)) \subseteq VT(B')$  的路径“ $\dots A' // B' \dots$ ”.

#### 4.1.1 类型 1 的查询算法

此时, 查询  $Q_p = \dots A // B \dots$  的基本要求是在输入标号  $A, B$  和查询目标时间区间  $VT(Q_p)$  前提下, 搜索形如“ $\dots A' // B' \dots$ ”包含  $VT(Q_p)$  的路径.

由于  $VT(B') \subseteq VT(A')$ , 实际只需  $VT(Q_p) \subseteq VT(B')$ , 因此, 需要先找到满足  $VT(Q_p) \subseteq VT(A')$  的起始元素  $A'$ , 然后在其后裔结点中搜索满足  $VT(Q_p) \subseteq VT(B')$  的结点. 由于结点集合  $\{A'\}$  和  $\{B'\}$  确定的路径集合包含所有在  $VT(Q_p)$  中有效的路径, 再根据“ $A$ ”和“ $B$ ”选择出最终的结果路径. 由此可得相应索引算法如下.

① 在 TXIDM 中找出满足  $VT(Q_p) \subseteq VT([u])$  的结点  $[u]$ , 这些  $[u]$  可能属于 TXIDM 的不同层次.

② 对于上述结点集合  $\{[u]\}$ , 在 TXIDM 中找出路径“ $\dots [u] // [v] \dots$ ”, 其中,  $VT(B) \subseteq VT([v]) \wedge VT(B) \not\subseteq VT(v')$ , 这里  $v'$  是  $v$  的子结点(如果存在的话).

③ 对于上述每条路径“ $\dots [u] // [v] \dots$ ”, 在  $[u]$  和  $[v]$  中按照时态包含关系确定的拟序结构分别搜索满足  $VT(Q_p) \subseteq VT(u')$  和  $VT(Q_p) \subseteq VT(v')$  的结点  $u', v'$ . 得到 TXQDM 中的路径即索引项集合

$$IS = \{\dots u' // v' \dots \mid VT(Q_p) \subseteq VT(v') \wedge u' \in [u] \wedge v' \in [v]\}$$

④ 在 IS 进行遍历搜索得到查询结果.

#### 4.1.2 类型 2 的查询算法

类型 2 的查询与类型 1 类似, 也分为 4 个步骤.

① 在 TXIDM 中找出满足  $VT(A) \subseteq VT([u])$  的结点  $[u]$  和所有满足  $VT(A) \subseteq VT([v])$  的结点  $[v]$ .

② 对于上述结点集合  $\{[u], [v]\}$ , 在 TXIDM 中找出路径“ $\dots [u] // [v] \dots$ ”, 其中,  $VT(B) \subseteq VT([v]) \wedge VT(B) \not\subseteq VT(v')$ , 这里  $v'$  是  $v$  的子结点(如果存在的话).

③ 对于上述每条路径“ $\dots [u] // [v] \dots$ ”, 在  $[u]$  和  $[v]$  中分别搜索满足  $VT(A) \subseteq VT([u'])$  和  $VT(B) \subseteq VT([v'])$  结点  $u'$  和  $v'$ .

④ 在上述结点确定的路径索引项集合  $\{\dots u' // v' \dots\}$  中再进行遍历搜索, 得到查询结果.

注意到在  $[u]$  中, 如果  $[u]$  中的当前结点  $a_0$  满足  $VT(A) \subseteq VT(a_0)$ , 则  $R_+(a_0)$  中的所有结点  $a'_0$  都满足  $VT(A) \subseteq VT(a'_0)$ ; 如果  $VT(B) \subseteq VT(a_0)$ , 则  $R_-(a_0)$  中的所有结点也如此. 对于类型 1 和类型 2 中步骤“③”来说, 可使用这些特点以提高搜索效率.

### 4.2 时态值查询

基于时态的值查询  $Q_v$  就是给定有效时间期间  $VT(Q_v)$ , 查询在  $VT(Q_v)$  内成立的 XML 文档内容. 由于文档内容通常为元素的属性值或元素内容,

这些在 TXIDM 中都是作为叶结点存在,因此,值查询就是对叶结点的查询,此时可以基于 STXIDM 建立相应操作.

① 在 STXIDM 中,找出所有满足  $VT(Q_v) \subseteq VT([u])$  的叶结点等价类  $[u]$ .

② 对于上述每个叶结点等价类  $[u]$ ,如果  $a_0$  是当前结点,按照时态期间包含关系拟序从  $a_0$  开始向后搜索.

如果  $VT(Q_v) \subseteq VT(a_0)$ ,此时  $R_+(a_0)$  中的元素都是查询结果.

如果  $VT(Q_v) \subseteq VT(a_0)$  不成立,应有  $VT_s(Q_v) < VT_s(a_0) \vee VT_e(Q_v) < VT_e(a_0)$ .

情况 1.  $VT_s(Q_v) < VT_s(a_0) \wedge VT_e(a_0) < VT_e(Q_v)$ ;

情况 2.  $VT_s(a_0) < VT_s(Q_v) \wedge VT_e(a_0) < VT_e(Q_v)$ ;

情况 3.  $VT_s(Q_v) < VT_s(a_0) \wedge VT_e(Q_v) < VT_e(a_0)$ .

无论哪种情况,首先考察  $\max R_+(a_0)$  是否查询结果,如果是,则索引查询结束;否则,就只在  $R_+(a_0)$  中搜索,因为此时不在  $R_+(a_0)$  中结点不可能是查询结果.

具体搜索方法是:对于  $a_i \in [u] \wedge a_i \in R_+(a_0)$ ,当  $VT(Q_v) \subseteq VT(a_i)$ , $a_i$  为查询结果,则  $R_+(a_i)$  元素也都是查询结果;当  $VT(Q_v) \subseteq VT(a_i)$  不成立,则  $R_+(a_i)$  元素就是可能查询结果.向后搜索,如果查询结果存在,就可得一个结点  $a_{i_0}$ ,使得  $a_0$  和  $a_{i_0}$  之间“ $a_1, a_2, \dots, a_{i_0-1}$ ”都不是查询结果,而  $a_{i_0}$  是.

需要指出的是,同一个查询结果  $b$  可以属于不同的  $R_+(a_i)$  和  $R_+(a_j)$ .

上述搜索算法告诉我们,在具体搜索过程中,如果  $b$  是查询结果,就不必对  $R_+(b)$  中的不同于  $b$  的元素进行搜索,它们都是查询结果;如果  $b$  不是查询结果,就不必对  $R_+(b)$  中的不同于  $b$  的元素进行搜索,它们都不是查询结果.这样可以提高搜索效率.

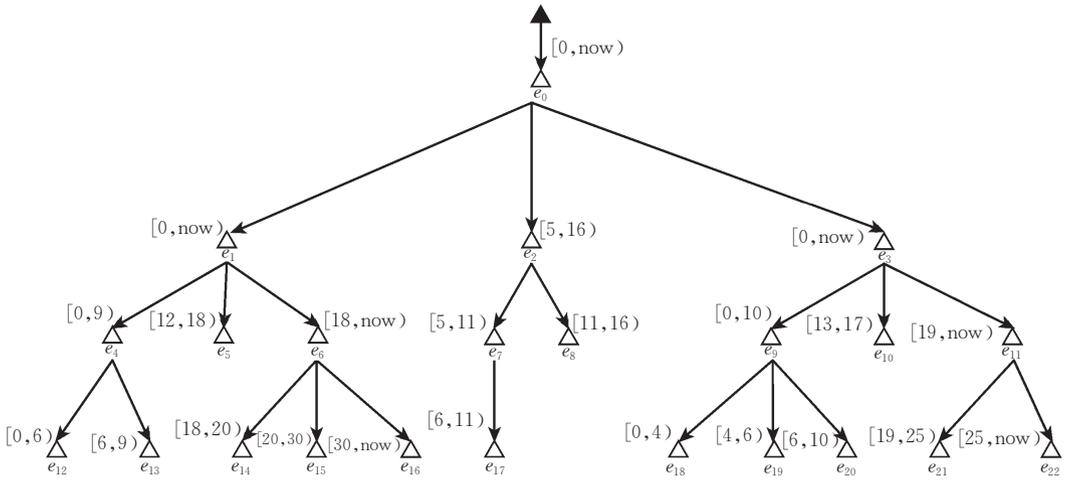


图 1 时态 XML 树实例

例 1. 设有如图 1 所示的时态 XML 树.

先将叶结点按照时态连通关系进行等价类划分,得到等价类如下:

$$E_1 : \{e_{18}, e_{12}, e_{19}\} [0, 6);$$

$$E_2 : \{e_{13}, e_{20}, e_{17}\} [6, 11);$$

$$E_3 : \{e_5, e_8, e_{10}\} [11, 18);$$

$$E_4 : \{e_{14}, e_{21}, e_{15}, e_{22}, e_{16}\} [18, now).$$

将上述等价类中的元素按照拟序关系进行排序:

$$E_1 : e_{18}(0), e_{12}(4), e_{19}(4);$$

$$E_2 : e_{13}(6), e_{20}(6), e_{17}(6);$$

$$E_3 : e_8(11), e_5(12), e_{10}(13);$$

$$E_4 : e_{14}(18), e_{21}(19), e_{15}(20), e_{22}(25), e_{16}(30).$$

查询包含在有效时间  $VT(Q) = [13, 14)$  内的路径.由  $E_3$  得到查询结果如图 2 中虚线所示.

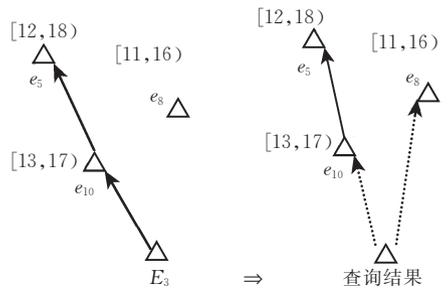


图 2 路径查询实例

### 5 时态 XML 索引更新

数据更新包括数据插入、数据删除和数据修改.数据修改可看作是删除原有数据再将修改后数据

作为新数据插入,不失一般性,本文仅讨论 TXIDM 的插入与删除算法。

### 5.1 插入操作关系集算法

在更新过程中,如果插入一个新的对象,就需将相应前缀码插入到 TXIDM 中. 设插入的新结点(前缀码)为  $b$ ,则需要讨论如何确定  $b$  相应等价类和拟序关系集。

#### 5.1.1 新结点等价类

由于  $b$  需要对应一个一个等价类,此时会有两种情形。

(1)  $b$  加入到某个原有等价类  $D$  中。

① 如果  $\exists D_j (VT(b) \subseteq VT(D_j))$ , 直接把  $b$  加入到  $D_j$  中,不会发生等价类合并。

② 如果  $\exists D_i (VT(D_i) \cap VT(b) \neq \emptyset)$ . 此时,将所有等价类  $D_j (1 \leq j \leq m)$  按其有效时间区间始点排序  $VTS(D_1) \leq VTS(D_2) \leq \dots \leq VTS(D_m)$ . 由  $D_1$  开始遍历搜索到  $D_{i_0}$  结束,这里  $D_{i_0}$  是  $\{D_i\}$  中第一个满足  $(VT(D_i) \cap VT(b) \neq \emptyset) \wedge VT(b) \not\subseteq VT(D_i)$  的等价类. 更改  $D_{i_0}$  的有效时间区间使  $VT(D_{i_0}) = VT(D_{i_0}) \cup VT(b)$ . 由于  $VT(D_{i_0})$  比原来  $D$  有所扩大,如果  $\exists D_j (VT(D_{i_0}) \cap VT(D_j) \neq \emptyset) (1 \leq j \leq t)$ , 则把  $D_j$  并入  $D_{i_0}$  中,且  $VT(D_{i_0}) = VT(D_{i_0}) \cup (\bigcup_{1 \leq j \leq t} VT(D_j))$ . 由此得到新结点  $b$  所在等价类  $D_{i_0}$ .

(2)  $b$  单独构成一个等价类。

如果  $\forall D_j (VT(D_j) \cap VT(b) = \emptyset) (1 \leq j \leq m)$ , 这时  $b$  可以单独构成一个等价类。

#### 5.1.2 新结点拟序关系集

在等价类  $D_{i_0}$  中,需要讨论新结点  $b$  的拟序关系,即建立  $b$  所在的拟序关系集合,也就是求出  $R(b) = R_+(b) \cup R_-(b)$ .

① 计算  $R_+(b)$ . 此时需要在  $D_{i_0}$  中搜索满足  $VT(b) \subseteq VT(a_j)$  的结点  $a_1, \dots, a_s$ , 这实际上是将  $VT(b)$  作为查询时间目标而在 TXIDM 中进行相应搜索. 应用前述索引查询算法就可确定相应结点. 由此得到新结点  $b$  的后向拟序关系集合  $R_+(b) = \{a_1, \dots, a_s\}$ .

② 计算  $R_-(b)$ . 设  $b$  所在等价类  $D_{i_0}$  中有结点  $a_1, \dots, a_t$ , 且  $VTS(a_1) \leq \dots \leq VTS(a_t)$ . 设有  $a_k$  为在  $VTS(a_1) \dots VTS(a_m)$  中不超过  $VTS(b)$  的最大者. 为了找到满足  $VT(a) \subseteq VT(b)$  的所有结点  $a$  可以从  $a_k$  开始向后搜索,直到找到一个  $a_{k+l}$  使  $VTE(b) < VTE(a_{k+l})$  成立,把  $a_k \dots a_{k+l}$  中每个结点的前向拟序关系集并集作为新结点  $b$  的前向拟序关系集。

### 5.2 基于递归的拟序关系集算法

为了提高更新效率,需要采用下述基于递归的

拟序关系算法。

#### 5.2.1 后向算子和前向算子

对于新结点得到的等价类  $D_{i_0}$ ,依照其中结点有效时间区间包含关系建立一个层次结构  $\langle V, E \rangle$ , 这里,  $\forall a, b \in V, \langle a, b \rangle \in E$ , 当且仅当  $VT(b) \subseteq VT(a)$ . 一个层次结构  $\langle V, E \rangle$  的实例如图 3 所示。

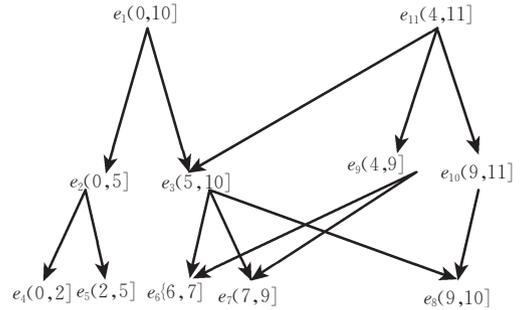


图 3 基于时态包含关系的拟序层次图

需要注意的是,层次结构  $\langle V, E \rangle$  一般而言是图. 对于图结构  $\langle V, E \rangle$ , 定义  $V$  上承继算子  $\mathcal{F}: V \rightarrow 2^V$  当且仅当对于  $\forall e \in V, \langle e, e_i \rangle \in E \vee e_i = e$ , 其中  $e_i \in \mathcal{F}(e)$  这说明对于  $E$  中任意结点  $e$ ,  $\mathcal{F}(e)$  中元素或者是  $e$  在图  $\langle V, E \rangle$  中的父结点, 或者就是  $e$  本身. 在图 3 中,  $\mathcal{F}(e_7) = \{e_7, e_5, e_9\}$ ,  $\mathcal{F}(e_4) = \{e_4, e_2\}$ ,  $\mathcal{F}(e_2) = \{e_2, e_1\}$ ,  $\mathcal{F}(e_1) = \{e_1\}$ .

**定义 5**(后向算子).  $\mathcal{F}: V \rightarrow 2^V$ , 对于  $\forall e \in V, \mathcal{F}(e) = \{e_j\}$ , 其中  $e_j, e_i \in \mathcal{F}(e)$  满足: 或者  $e_i = e$  或者  $\langle e, e_i \rangle \in E$ , 即对于  $E$  中任意结点  $e$ ,  $\mathcal{F}(e) \subseteq V$ , 而  $\mathcal{F}(e)$  中元素或者是  $e$  在图  $\langle V, E \rangle$  中的父结点, 或者就是  $e$  本身. 例如在图 3 中, 通过计算可得  $\mathcal{F}(e_7) = \{e_7, e_5, e_9\}$ ,  $\mathcal{F}(e_4) = \{e_4, e_2\}$ ,  $\mathcal{F}(e_2) = \{e_2, e_1\}$ ,  $\mathcal{F}(e_1) = \{e_1\}$ .

后向算子  $\mathcal{F}$  可以进行“复合”运算, 即  $\mathcal{F}(\mathcal{F}(e)) = \bigcup_{e_i \in \mathcal{F}(e)} \mathcal{F}(e_i)$ . 后向算子  $\mathcal{F}$  的作用是在图结构  $\langle V, E \rangle$  中, 可以递归地求出给定结点的后向拟序关系类. 在图 3 中, 通过计算可得  $R_-(e_7) = \mathcal{F}(\mathcal{F}(e_7)) = \mathcal{F}(e_7) \cup \mathcal{F}(e_5) \cup \mathcal{F}(e_9) = \{e_7; e_5, e_1, e_{11}; e_9\}$ .

类似, 可以定义前向算子。

**定义 6**(前向算子). 前向算子  $\mathcal{H}: V \rightarrow 2^V$  定义如下, 对于  $\forall e \in V, \mathcal{H}(e) = \{e_j\}$ , 其中  $e_j, e_i \in \mathcal{H}(e)$  满足: 或者  $e_i = e$  或者  $\langle e_i, e \rangle \in E$ , 即对于  $E$  中任意结点  $e$ ,  $\mathcal{H}(e) \subseteq V$ , 而  $\mathcal{H}(e)$  中元素或者是  $e$  在图  $\langle V, E \rangle$  中的子结点, 或者就是  $e$  本身。

前向算子也可以进行复合运算. 前向算子的作用是计算前向拟序关系集。

#### 5.2.2 拟序关系集算法

下面讨论后向拟序关系集算法. 在插入操作中,

首先要找到待插入结点  $b$  所在的等价类  $D_{i_0}, D_{i_0}$  中的结点按照有效时间区间始点从小到大排序, 设这个序列  $\mathcal{L}$  为  $a_1, \dots, a_m$ . 按照原来插入过程, 首先要在  $\mathcal{L}$  中找到  $a_i$ , 其中  $i$  是满足  $VT_s(a_k) \leq VT_s(b)$  的  $a_k$  中下标最大者. 若有多个  $a_k$  同时满足条件, 则取其中有效时间区间最小者. 此时, 执行下述计算步骤:

1. 如果  $VT(b) \subseteq VT(a_i)$ , 则执行步 4;
2. 如果  $VT(b) \subseteq VT(a_i)$  不成立, 则执行步 3;
3. 计算  $\mathcal{F}(a_i)$ ;
4. 对于  $a \in \mathcal{F}(a_i)$ , 如果  $VT(b) \subseteq VT(a)$ , 计算出  $\mathcal{F}(\mathcal{F}(\dots \mathcal{F}(a)))$  直到根结点. 此时, 执行步 4;
5. 对于  $a \in \mathcal{F}(a_i)$ , 如果  $VT(b) \not\subseteq VT(a)$ , 执行步 1;
6. 由于  $\mathcal{F}(\mathcal{F}(\dots \mathcal{F}(a))) \subseteq R_+(b)$ , 将  $\mathcal{F}(\mathcal{F}(\dots \mathcal{F}(a)))$  中的元素添加到  $R_+(b)$ .

可以证明, 由此得到的结果就是  $R_+(b)$

**例 2.** 设有层次结构图如图 4 所示, 其中,  $e_5$  是新插入的结点.

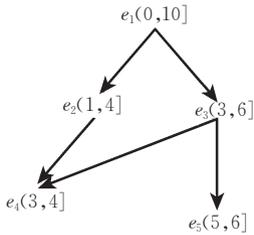


图 4 后向拟序关系集算法实例

现求  $e_5$  的后向拟序关系集.

按照结点有效时间始点大小得到排在  $e_5$  前面的结点为  $e_1, e_2, e_3, e_4$ . 选择  $e_4$  作为算法中的起始结点  $a_i$ . 由于不成立  $VT(e_5) \subseteq VT(e_4)$ , 求出  $\mathcal{F}(e_4) = \{e_2, e_3\}$ .  $VT(e_5) \subseteq VT(e_3)$ , 把  $\mathcal{F}(\mathcal{F}(e_3))$  加入到  $R_+(e_5)$ ; 而  $VT(e_5) \subseteq VT(e_2)$  不成立, 求出  $\mathcal{F}(e_2) = \{e_1\}$ , 而  $VT(e_5) \subseteq VT(e_1)$ , 将  $\mathcal{F}(e_2)$  加入到  $R_+(e_5)$ , 由此即得  $R_+(e_5) = \{e_1, e_2, e_3\}$ .

与后向拟序关系集算法类似, 可以得到前向拟序关系集算法, 这里不再赘述.

**例 3.** 设有层次结构图如图 5 所示,  $e_{11}$  是新插入结点.

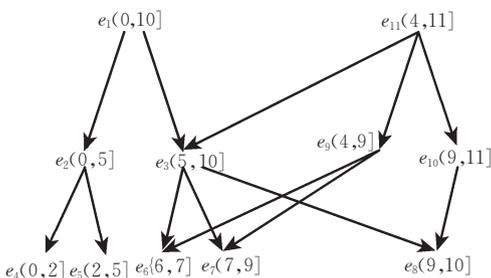


图 5 前向拟序关系集算法实例

现构造  $e_9$  的前向拟序关系集. 在  $\mathcal{L}$  中,  $e_9$  后面的结点按次序排列为  $e_3, e_6, e_7, e_{10}, e_8$ , 取  $e_3$  为算法中起始结点. 由  $VT(e_3) \subseteq VT(e_{11})$  不成立,  $\mathcal{H}(e_3) = \{e_6, e_7, e_8\}$ ,  $VT(e_6), VT(e_7), VT(e_8) \subseteq VT(e_{11})$ ,  $VT(e_3) \subseteq V$  到  $R_-(e_{11})$ . 另外,  $VT(e_{10}) \subseteq VT(e_9)$  不成立, 将  $\mathcal{H}(e_{10}) = \{e_8\}$ , 所以  $R_-(e_9) = \{e_6, e_7\}$ .

构造  $R(b)$  的过程实际上就是求以  $VT(b)$  为查询目标的查询结果, 所以等价于查询的时间复杂度, 而递归式拟序关系集和非递归拟序关系集的区别只在于包含  $VT(b)$  的结点是分布在一个集合中还是分布在多个集合中. 因此无论构造哪种拟序关系集, 耗费的时间是一样的, 设等价类  $D$  有  $n$  个结点则对  $D$  中结点建立拟序关系集要耗费  $O(n^2)$  的时间.

### 5.3 时态 XML 索引记录删除算法

与插入结点类似, 需要考虑删除对等价类和拟序关系集的影响.

#### 5.3.1 删除与等价类

删除结点  $b$  可能会引起等价类分裂, 具体处理策略是:

- ① 如果  $b$  对应祖先结点有效时间区间改变 (即有效时间区间缩小), 调整  $b$  对应祖先结点有效时间区间后执行等价类分裂操作.
- ② 如果  $b$  对应祖先结点有效时间区间不改变, 直接执行等价类分裂操作.

因此, 在删除  $b$  的情况下, 首先, 判断等价类是否需要分裂; 其次, 分裂等价类.

**算法 1.** 判断是否需要分裂算法  $Process1(D)$ .

1.  $D$  中所有结点按时间期间起点从小到大排序, 设这个序列为  $L_0$ ;
2.  $D$  中所有结点按时间区间的终点从小到大排序, 设这个序列为  $L_1$ ;
3. 在序列  $L_0$  中找到  $b$  结点的前邻结点  $a$ , 在序列  $L_1$  中找到  $b$  的后继结点  $c$ ;
4. 如果  $a$  和  $c$  连通, 则  $D$  不需要分裂;
5. 否则  $D$  要进行分裂.

假设算法在步 4 结束, 对于任何  $d \in D - \{b\}$  有: 若  $d$  是  $L$  中在  $b$  之前的结点, 由时态连通关系可知, 所有  $L$  中在  $b$  之前的结点在同一个等价类中. 若  $d$  结点就是  $c$  结点, 则  $c$  结点与  $L$  中在  $b$  之前的结点在同一个等价类中. 若  $d$  是  $L$  中在  $b$  之后的结点, 显然存在  $w_1, \dots, w_m$ , 在  $L$  中排在  $d$  之前, 且  $VT(a) \cap VT(w_1) \neq \emptyset, VT(w_1) \cap VT(w_2) \neq \emptyset, \dots, VT(w_{m-1}) \cap VT(w_m) \neq \emptyset, VT(w_m) \cap VT(d) \neq \emptyset$ . 若  $b \notin \{w_1, \dots, w_m\}$ , 则  $d$  与  $L$  中在  $b$  之前的结点在同一个等价类中; 若  $b \in \{w_1, \dots, w_m\}$ , 则设  $b =$

$w_i$ , 现在用  $v, c$  替代  $b$  有:  $VT(a) \cap VT(w_1) \neq \emptyset$ ,  $VT(w_1) \cap VT(w_2) \neq \emptyset, \dots, VT(w_{i-1}) \cap VT(v) \neq \emptyset, VT(v) \cap VT(c) \neq \emptyset, VT(c) \cap VT(w_{i+1}) \neq \emptyset, \dots, VT, VT(w_m) \cap VT(d) \neq \emptyset$ . 所以  $d$  和  $L$  中在  $b$  之前的结点在同一个等价类中. 综上所述, 等价类  $D$  不需要分裂.

**算法 2.** 分裂等价类算法  $Process2(D)$ .

1. 运行  $process1(D)$ , 如果  $D$  不需要等价类分裂, 算法结束;
2. 否则执行如下步骤
3. 把  $L$  中在  $b$  之前的节点  $d$  及  $R_-^*(d)$  都合并成等价类  $D_1$ ;
4. 把  $R_-^*(c)$  和  $c$  都合并成等价类  $D_2$ ;
5. 若  $e \in R^-(b)$  且  $VT(e) \cap VT(D_1) \neq \emptyset$ , 则把  $e$  合并到  $D_1$ , 且  $VT(D_1) = VT(D_1) \cup VT(e)$ ;
6. 若  $e \in R^-(b)$  且  $VT(e) \cap VT(D_2) \neq \emptyset$ , 则把  $e$  合并到  $D_2$ , 且  $VT(D_2) = VT(D_2) \cup VT(e)$ ;
7. 使用时态连通关系对  $R^-(b)$  中剩余的节点进行等价类划分.

### 5.3.2 删除与拟序关系集

对于被删除结点, 其对应的拟序关系集也会被删除,  $b$  子树中所有结点及其拟序关系集也会被删除. 此时, 需要考虑  $b$  对应祖先结点有效时间区间是否发生改变. 在改变情况下, 可以先更改相应时间区间, 再改变拟序关系集.

①对于  $R_-(b)$  结点  $e$ , 在  $R_+(e)$  中删除  $R_+(b)$  中是  $b$  的子孙结点;

②对于  $R_+(b)$  结点  $e$ , 如果  $e$  不是  $b$  子孙结点, 则在  $R_-(e)$  中删除  $b$ ;

③对  $b$  子孙结点重复①②的操作;

④对于每个时间区间发生变更的结点  $d$ , 检查  $R_+(d)$  中每个结点  $d'$ , 若  $VT(d') \not\subseteq VT(d)$ , 则把  $d'$  从  $R_+(d)$  中删除  $d'$ , 而从  $R_-(d')$  中删除  $d$ .

## 6 实验设计与结果评估

为了检验本文提出索引技术的合理性、有效性和实用性, 我们设计了相应的实验.

### 6.1 实验环境、参数和实验数据

为了适应一般的情况, 本文选取相对较弱的硬件和软件环境: Windows 98; CPU 主频 500MHz; 主存 192MB.

时间期间  $[VT_s, VT_e]$  的跨度定义为  $VT_e - VT_s$ . 由于本文提出的索引技术是基于时态期间等价类, 时间区间跨度对于等价类建立具有一定影响: 一般而言时间区间跨度较大则等价类数目较少, 且会引起结点在等价类之间分布不均, 索引模型效率较低. 对于记录中的有效时间区间, 使用随机方式生成. 由于时间具一维线性性质,  $VT_s, VT_e$  均与整数对应, 生成的有效时间区间可用整数对表示, 时间期间跨度对应一个正整数. 在实验中, 取结点时间期间跨度平均值为 200(单位), 内部结点时间区间跨度平均值为 500(单位). 实验中所使用的数据如表 1 所示.

表 1 实验基本数据

结点数	分支数	次数比值		时间耗费比	结点数	分支数	次数比值		时间耗费比
		TXQDM	TXIDM				TXQDM	TXIDM	
18	5	11.27	10.36	0.91930	3944	25	730.22	50.42	0.06905
43	10	19.57	14.40	0.73582	4725	30	782.49	52.03	0.06649
118	25	47.42	22.52	0.47490	5650	45	916.65	44.69	0.04905
209	10	62.96	27.73	0.44040	6934	20	828.81	47.53	0.05735
343	15	85.38	33.91	0.39717	8094	25	1052.44	51.55	0.04898
447	20	107.91	39.36	0.36475	8912	28	1149.29	49.95	0.04346
1125	8	161.90	37.64	0.23249	10001	30	1593.25	60.28	0.03783
1920	10	299.59	35.81	0.11953	10758	35	1727.78	53.45	0.03094
2891	15	505.07	60.56	0.11990	11435	40	1966.06	53.85	0.02734

说明: (1) 结点数(node number): 随机生成的 XML 数据.

(2) 分支数: 时态 XML 树中各内部结点具有的最多子结点数.

(3) 完成给定查询所需次数的比值: 测试过程中的比较总次数(无论成功与失败)与测试用例数比值.

(4) 时间耗费比: 使用等价类中拟序结构的比较次数与查询模型查询比较次数比值.

## 6.2 实验结果分析与评估

### 6.2.1 基于时态的值查询

(1) 索引模型与遍历 XML 文档的比较

此结果如图 6 所示, 由此可知, 随着时态 XML 树规模增加, 索引模型的查询优势会更加明显.

(2) 查询时间耗费比例趋势

查询时间耗费比例变化趋势如图 7 所示.

这里, 以遍历 XML 文档的时间耗费为基准, 其耗费比为“1”.

### 6.2.2 路径查询的综合测试

针对“ $\dots A // B \dots$ ”“ $// A \dots$ ”两种情况, 不再细分为内结点和叶结点的差别. 在用户给定“ $\dots A // B \dots$ ”

和 VTQ 时, 查询分两步进行. 首先在叶结点的索引模型中查找, 然后在内结点的索引模型中查找, 时间

耗费是在这两者中的时间耗费之和.

路径查询使用的数据基本情况如表 2 所示.

表 2 内部结点查询基本数据

结点数	分支数	次数比值		时间耗费比	结点数	分支数	次数比值		时间耗费比
		TXQDM	TXIDM				TXQDM	TXIDM	
10122	50	1106.09	255.93	0.2314	3175	12	377.82	159.56	0.4223
10032	45	1079.82	257.72	0.2386	2245	10	278.79	147.69	0.5307
9263	40	1065.31	254.42	0.2388	1847	9	210.92	145.87	0.6916
8973	35	986.98	237.93	0.2411	1570	8	175.80	127.71	0.7265
7946	30	888.87	242.80	0.2732	1188	7	108.02	120.16	1.1124
7310	25	863.17	230.75	0.2673	532	5	66.93	91.53	1.3675
6894	23	791.13	221.69	0.2802	231	4	39.73	60.39	1.5200
4659	20	680.61	204.74	0.3009	91	3	20.20	44.19	2.1876
3731	15	507.20	177.68	0.3503	33	2	10.97	28.48	2.5962

(1) 路径查询的比较次数趋势图

路径查询比较次数趋势如图 8 所示.

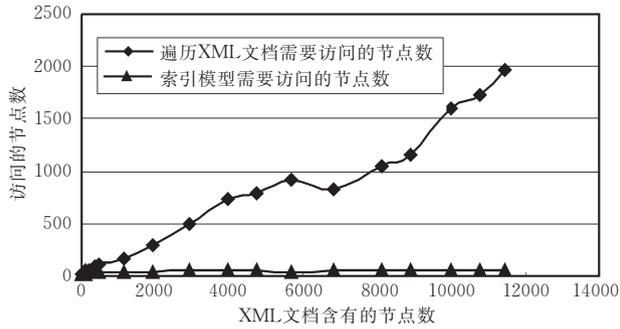


图 6 TXIDM 与 TXQDM 比较

(2) 路径查询时间耗费比例趋势

路径查询时间耗费比例趋势如图 9 所示.

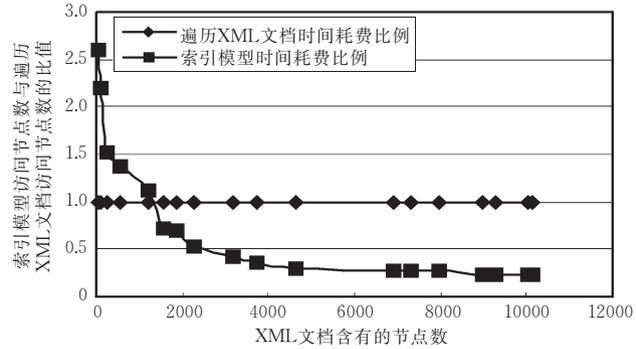


图 9 时间耗费比例趋势

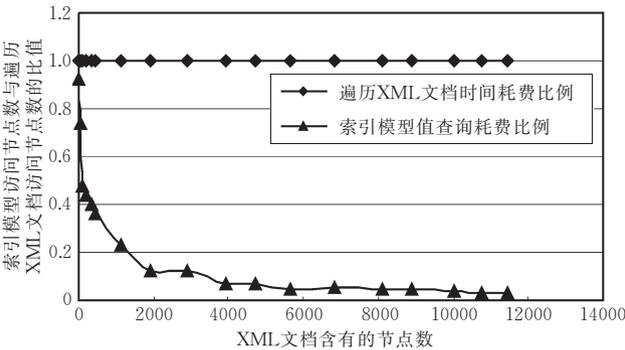


图 7 查询时间耗费比例趋势

(3) 时间期间与查询时间耗费的关系

时间期间与查询时间耗费关系如图 10 所示.

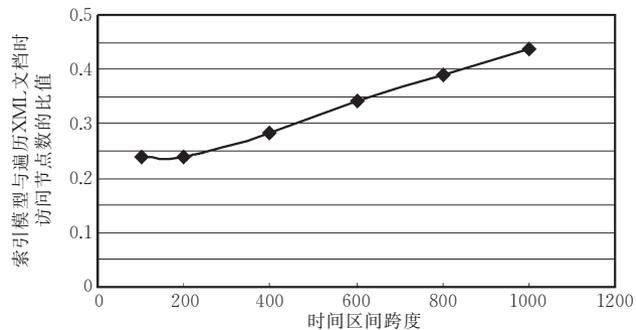


图 10 时间区间与查询时间耗费的关系

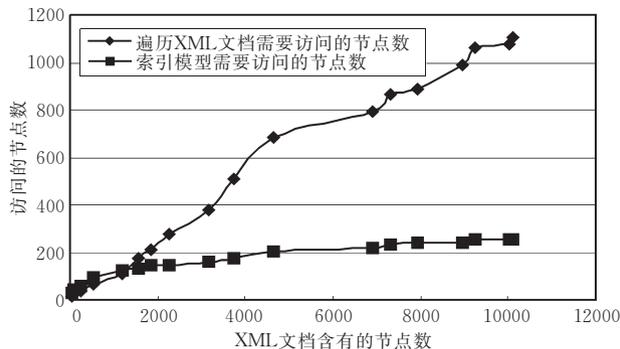


图 8 索引模型与查询模型比较

图 10 中以查询模型的时间耗费为单位, 横轴给定的是时间区间参数, 参数相差 0.1 则区间跨度平均相差 100. 图 10 横轴从左到右说明时间区间依次增大, 而在时间区间较小时模型查询效率具有优势.

6.2.3 两种关系集算法空间复杂度比较  
关系集算法与递归式算法空间复杂度比较仅列出在 4000 记录以上的两种拟序关系算法空间复杂度比较如图 11 所示.

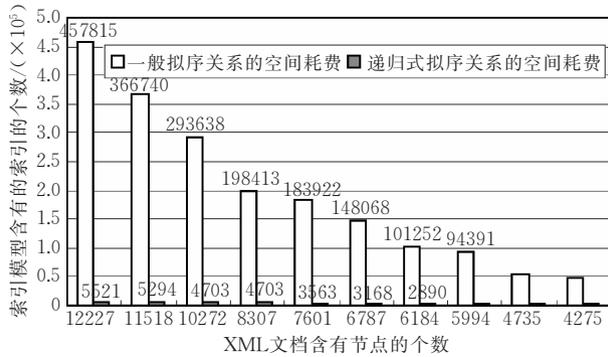


图 11 关系集算法与递归算法空间复杂度比较

## 7 结 语

本文研究了时态 XML 文档索引技术及其实现. 首先讨论了时态 XML 查询数据模型 TXQML, 提出了基于结点有效时间的前缀编码方案; 其次通过引入了结点间的时态连通关系, 将 TXQML 结点进行等价分类, 再在等价类中排序引入时态拟序关系, 对类中元素进行排序, 从而建立了具有“嵌套”特征的时态 XML 索引模型 TXIDM 和 STXIDM, 其中 TXIDM 用于基于时态的路径查询, STXIDM 应用于基于时态的值查询. XML 数据的更新一直是 XML 数据操作的难点, 对于时态 XML 索引也是如此. 本文通过统一的基于拟序关系集的一系列算法, 依照时态查询操作的基本思路, 实现了时态 XML 索引插入与删除的更新算法. 最后, 设计了相关的实验. 理论分析和相关实验结果说明, 本文提出的时态 XML 索引技术是有效的和可行的. 在本文提出的框架下, 可以继续探讨其他相关的基本问题, 例如双时态 XML 索引技术、在索引建立与操作过程中有效时间变量处理等, 限于篇幅, 这些课题将另文讨论.

## 参 考 文 献

[1] Wan Chang-Xuan, Liu Yun-Sheng, Xu Sheng-Hua, Liu Xi-Ping, Lin Da-Hai. Indexing XML data based on region coding for efficient processing of structural joins. *Chinese Journal of Computers*, 2005, 28(1): 113-127(in Chinese)

(万常选, 刘云生, 徐升华, 刘喜平, 林大海. 基于区间编码的 XML 索引结构的有效结构连接. *计算机学报*, 2005, 28(1): 113-127)

- [2] Bao Xiao-Yuan, Song Zai-Sheng, Tang Shi-Wei, Yang Dong-Qing, Wang Teng-Jiao. Suffix index—An XML index structure based on suffix tree. *Journal of Computer Research and Development*, 2004, 41(10): 1793-1801(in Chinese)  
(包小源, 宋再生, 唐世渭, 杨冬青, 王腾蛟. Suffix Index——一种基于后缀树的 XML 索引结构. *计算机研究与发展*, 2004, 41(10): 1793-1801)
- [3] Xu Hai-Yuan, Wu Quan-Yuan, Wang Huai-Min, Jia Yan. Containment based XML indexing. *Acta Electronica Sinica*, 2003, 31(8): 1155-1159(in Chinese)  
(徐海渊, 吴泉源, 王怀民, 贾焰. 基于相容关系的 XML 索引机制. *电子学报*, 2003, 31(8): 1155-1159)
- [4] Tansel A U. Temporal relational data model. *IEEE Transactions on Knowledge and Data Engineering*, 1997, 9(3): 464-479
- [5] Ye Xiao-Ping, Tang Yong. Semantics on "Now" and calculus on temporal relations. *Journal of Software*, 2005, 16(5): 838-846(in Chinese)  
(叶小平, 汤庸. 时态变量“Now”语义及相应时态关系运算. *软件学报*, 2005, 16(5): 838-845)
- [6] Kha D D, Yoshikawa M, Uemura S. An XML indexing structure with relative region coordinate//*Proceedings of the 17th International Conference on Data Engineering*. Heidelberg, Germany, 2001: 313-320
- [7] Zhang Shuo-Hao, Dyreson C E. Adding valid time to XPath//Bhalla S ed. *Proceedings of the DNIS 2002*, LNCS 2544. Heidelberg, Berlin: Springer-Verlag, 2002: 29-42
- [8] Amagasa T, Yoshikawa N, Uemura S. Realizing temporal XML repositories using temporal relational database//*Proceedings of the 3rd International Symposium on Cooperative-Database System for Advanced Applications*. Beijing, China, 2001: 60-64
- [9] Amagasa T, Yoshikawa M, Uemura S. A data model for temporal XML documents//*Proceedings of the DEXA 2000*. England, 2000
- [10] Grandi F, Mandreoli F, Tiberio P, Bergonzini M. A temporal data model and management system for normative texts in XML format//*Proceedings of the WIDM'03*. New Orleans, Louisiana, USA, 2003
- [11] Mendelzon A O, Rizzolo F, Vaisman A. Indexing temporal XML documents//*Proceedings of the 30th VLDB Conference*. Toronto, Canada, 2004
- [12] Dyreson C E. Observing transaction-time Semantics with XPath//*Proceedings of the 2nd International Conference on Web Information Systems Engineering*. Kgotto, Japan, 2001: 193-202



**YE Xiao-Ping**, born in 1955, Ph.D., association professor. His research interests are database, processing of temporal information, discovery of knowledge.

**CHEN Kai-Yuan**, born in 1983, graduate student. His research interests are temporal database, MAS, analysis of conflict.

**TANG Yong**(corresponding author), born in 1964, Ph.D., professor, Ph.D. supervisor. His research interests are database, knowledge base and cooperative software.

**TANG Na**, born in 1975, Ph.D. candidate, lecturer. Her research interests are databases and the technology of

XML.

**HU Su**, born in 1983, M. S. candidate. Her research

interests are temporal database, temporal data mining.

## Background

This work was supported by the following research projects: The National Nature Science Foundation of China: "Research on Bi-Temporal XML Model and Applications" (No. 60673135), "Research on Temporal Knowledge/Data Model and Software Components" (No. 60373081), and the Nature Science Foundation of Guangdong Province in China: "Study of Unified Model and Applied Software Component on Temporal Knowledge and Data" (No. 04105503), "Study on Cooperated and Temporal Mechanism of Multi-Agent System and Exploitation of Software component" (No. 5003348). These projects are mainly to be devoted to develop the new theories and techniques for the processing on the temporal data and knowledge information, such as an implementation on temporal query language, the temporal middle wares, temporal variables, the methods on bi-temporal index, the storing and query technology of temporal XML data, the approaches on temporal cooperation and so on.

The authors have implemented the middleware TempDB V2.0 of temporal databases based on TSQL2 and the software component TempKB V1.0 ground on generation rules. The latter adopts the positive reasoning mechanism with tem-

poral data which be driven by temporal demanding and supports the temporal driven constrain. These works are the fundamental of the paper. In addition, the authors have also completed the study on the semantics of temporal variables and proposed the processing techniques of temporal data based on these variables, and the manipulation approaches on bi-temporal data in the variable database. And then, the authors further extend these to the issues of temporal XML data. Of cause there are sharp differences between the temporal relation and the temporal XML, but both two may be enough to using the basic characteristic of the time periods of the corresponding data. The temporal relationships just are the key issue of the works described in the paper, for example, the paper builds the temporal index model by means of the temporal connection and inclusion relationships, and proposes the arithmetic of the query and update using the property of quasi-order. The further work of the authors is extending the TXIDM in order to implement the integrated semantics of the valid time variable, and study the binding of the variable within the operation of the temporal XML index.