

基于 XML 的时态数据管理

蒋夏军, 皮德常

(南京航空航天大学 信息科学与技术学院, 江苏 南京 210016)

[摘要] 时态数据的多种查询操作要求数据以时间为基准成组存放, XML 文档中数据的半结构化特性很好地适应了这一要求. 在不考虑索引技术等查询优化方法的情况下, 时态数据文件的长度是影响查询效率的关键因素之一. 首先探讨了两种常用的表示时间属性的方法: 属性时戳模型和元素时戳模型; 提出了一种新的方法: 前缀时戳模型. 在 3 种时态 XML 数据模型的基础上, 利用 XML 文档中祖先元素与子孙元素在时间区间上的相关性, 消除了文档中的时间冗余信息, 实验结果表明这一方法能够取得较好的压缩比.

[关键词] XML, 时态数据, 数据模型

[中图分类号] TP 392 [文献标识码] A [文章编号] 1672-1292(2008)04-0082-05

Temporal Data Management Based on XML

Jiang Xiajun, Pi Dechang

(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract The efficiency of some query operations can benefit from temporally grouped data models greatly. And the semi-structural characteristic of XML document is well suited for such models of temporal database. Without considering the query optimization of index technology, the length of temporal data document is one of the key factors for the query operations. Two methods of time representation for temporal XML database—attribute time-stamped model and element time-stamped model are researched firstly in this paper. Then a new method—text prefix time-stamped model is presented. Based on the three temporal XML data models, the elimination of time information redundancy is also studied by making use of time interval relationship among elements of the temporal XML documents. Experiments show that after eliminating of time information redundancy, the length of documents is rapidly dropped.

Key words XML, temporal data, data model

数据的互操作性是信息科学领域的一个重要问题, 国内外研究人员为此展开了多方面的研究工作, XML 的诞生是上述工作取得的一项标志性成就. XML 的显著特点是平台无关性和半结构化, 对于人类和计算机程序, XML 文档内容都是容易阅读和编写的^[1].

时间属性和空间属性往往是多维的, 在时态数据库 (Temporal Database) 中, 研究人员将时间属性作为一种特殊属性对待, 通常使用两种模型来描述^[2]: 线性 (Linear) 模型和分支 (Branching) 模型. 时间属性与一般属性的不同组合使得数据库中与实体相对应的单元的表示非常复杂. 时态数据库中常见的时间维包括两类: 有效时间和事务时间. Clifford 等将时态数据模型分为两类^[3]: 以时间为基准成组存放数据的模型和非成组存放的模型. 前者具有更强的表达能力, 表现实体状态随时间的变化过程更加自然, 是一种比较适合时态数据管理系统的模型. 然而, 传统的关系数据模型更加适用于后者, 如 TQues^[4]中使用的元组时戳模型, 它的最大缺陷是任何属性的变化都将产生一个新的带时戳的元组, 而且同一实体不同时刻对应的元组在存储位置上并不邻近.

XML/XQuery 标准的推出为时态数据管理提供了新的机会, 基于 XML 的时态数据模型具有天然的成组存放同一实体不同时刻的状态数据的特点. 此外, XML 技术管理时态数据的优势还表现在: XQuery 提供

收稿日期: 2008-06-18

基金项目: 南京航空航天大学引进人才科研基金 (S0677-042) 资助项目.

通讯联系人: 蒋夏军, 博士, 讲师, 研究方向: 领域时空数据库、计算机仿真. E-mail: xiajunji@nuaa.edu.cn

了一种可扩展的、Turing 完备的语言, 在这种语言的基础上能够定义各种时态查询函数^[5 6].

XML 中表示时间的最直接方法是将时间信息作为元素的属性来存放, 如文献 [7] 将 职工 实体表示成图 1 的形式; 另一种常用的方法是定义一个孩子元素表示时间信息, 如文献 [8] 定义 < valid> 标识表示实体的有效时间. 本文在上述两种方法的基础上, 对时间表示的冗余信息进行消除, 同时提出一种将时间属性表示成元素的文本值前缀的方法.

在比较时态数据模型以前, 首先定义两类与时间信息联系密切的典型的查询, 为了方便描述, 以图 1 给出的 employees.xml 为例:

Q_1 为查询给定时刻所有职工的信息, 即 快照查询; Q_2 为查询给定职工号的某个职工的所有历史和当前信息, 即 实体轨迹查询.

1 属性时戳模型

图 1 所示的职工文档在元素属性中增加了时间项: tend 和 tstart, 分别表示事务时间区间的终止值和起始值. 图中省略了次要元素, 工资的变化过程也没有完整列出.

```
< employees tend =" 9999-01-01 " tstart =" 1985-01-01 ">
...
< employee tend =" 9999-01-01 " tstart =" 1987-04-03 ">
< empno tend =" 9999-01-01 " tstart =" 1987-04-03 "> 10018 </ empno >
...
< deptno tend =" 1992-07-29 " tstart =" 1987-04-03 "> d005 </ deptno >
< deptno tend =" 9999-01-01 " tstart =" 1992-07-29 "> d004 </ deptno >
...
< title tend =" 1995-04-03 " tstart =" 1987-04-03 "> Engineer </title>
< title tend =" 9999-01-01 " tstart =" 1995-04-03 "> Senior Engineer </title>
< salary tend =" 1988-04-02 " tstart =" 1987-04-03 "> 55881 </salary>
< salary tend =" 1989-04-02 " tstart =" 1988-04-02 "> 59206 </salary>
...
< salary tend =" 9999-01-01 " tstart =" 2002-03-30 "> 84672 </salary>
</ employee >
...
</ employees >
```

图 1 包含时间信息的职工情况 XML 文档

Fig.1 The employee XML document including time attribute

1.1 模型定义

属性时戳模型的一般形式为:

```
< element 0 tend=" element 0-end-time " tstart=" element 0-start-time ">
< element 1_1 tend=" element 1_1-end-time " tstart=" element 1_1-start-time ">
  Element 1_1 value </ element 1_1 >
< element 1_2 tend=" element 1_2-end-time " tstart=" element 1_2-start-time ">
  Element 1_2 value </ element 1_2 >
...
< element 1_j tend=" element 1_j-end-time " tstart=" element 1_j-start-time ">
  Element 1_j value </ element 1_j >
...
</ element 0 >
```

且满足如下条件(其中, i, j, k 为正整数):

(1) element i_j value (i 表示层次, j 表示兄弟序号)可能的两种取值:

- A. 文本, 则 element i_j 在树结构中为叶子结点;
- B. 元素, 其孩子结点(一个或多个)形式如下:

```
< element i+1_k tend=" element i+1_k-end-time " tstart=" element i+1_k-start-time ">
  Element i+1_k value </ element i+1_k >
```

(2) [element i_j -start-time, element i_j -end-time) \supset

$$\bigcup_{\forall k, \text{child}(\text{element } i_j, \text{element } i+1_k)} [\text{element } i+1_k\text{-start-time}, \text{element } i+1_k\text{-end-time})$$

其中, $\text{child}(\text{element } i_j, \text{element } i+1_k)$ 表示 $\text{element } i+1_k$ 为 $\text{element } i_j$ 的孩子元素.

1.2 时间冗余信息的消除

与关系模型不同的是,XML 文档中的元素之间具有层次关系.在属性时戳模型中,父亲和孩子结点的时间区间具有一定的相关性,利用这种相关性,能够消除子孙结点的多余时间属性值.

图 2 为消除冗余时间信息后的 employees.xml 文档的部分数据.图中许多结点的时间信息不再完备,查询时可能无法直接获得元素结点的事务时间,但通过祖先结点的时间属性值可以间接取得.消除时间冗余信息后,最直接的结果是数据文件的长度缩小.对于 Q_2 类查询,查询目标涉及实体的完整信息,由于 XML 文档以时间为基准成组存放, I/O 开销将显著降低;对于 Q_1 类查询,查询目标涉及所有实体,存储页面的 I/O 开销也将有所降低.

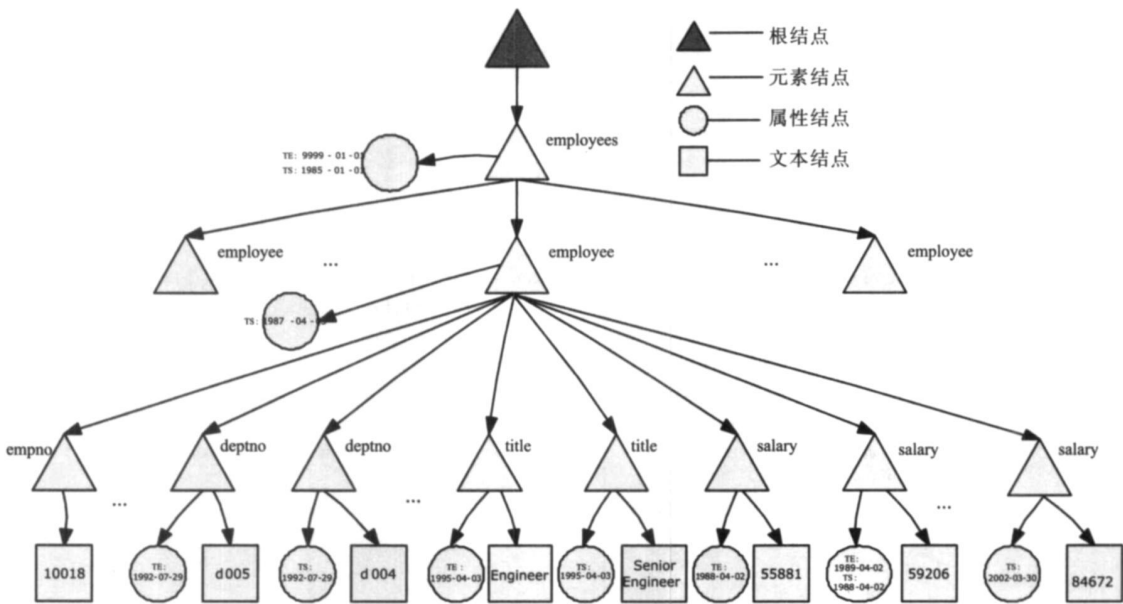


图 2 消除时间冗余信息后的属性时戳模型例子
Fig.2 Attribute time-stamped model eliminating time redundancy

2 元素时戳模型

另一种方法是时间维定义成元素结点的孩子元素,在例子 employees.xml 中,如果只考虑事务时间,则在每个元素的孩子元素中增加一个兄弟元素 <trans>.

2.1 模型定义

元素时戳模型的一般形式为:

```
<element 0>
  <Trans> element 0-end-time _element 0- start-time </Trans>
  <element 1_1>
    <Trans> element 1_1-end-time _element 1_1- start-time </Trans>
    Element 1_1 value
  </element 1_1>
  <element 1_2>
    <Trans> element 1_2-end-time _element 1_2- start-time </Trans>
    Element 1_2 value
  </element 1_2>
  ...
  <element 1_j>
    <Trans> element 1_j-end-time _element 1_j- start-time </Trans>
    Element 1_j value
  </element 1_j>
```

...

</element0>

并且满足如下条件(其中, i, j, k 是正整数.):

(1) element i_j value(i 表示层次, j 表示兄弟序号)可能的两种取值:

- A. 文本, 则 element i_j 在树结构中为叶子结点的父结点;
- B. 元素, 其孩子结点(一个或多个)形式如下:

```
<element i+1_k>
  <Trans> element i+1_k-end-time_ element i+1_k-start-time </Trans>
  Element i+1_k value
</element i+1_k>
```

且 Trans 结点为相应元素的第一个孩子结点.

(2) 与 1.1 中条件(2)类似.

2.2 时间冗余信息的消除

父亲结点的时间信息定义为其第一个孩子元素结点, 所以每个元素的时间信息除了由其第一个孩子结点确定外, 还与其表示父亲结点的时间信息的兄弟结点相关.

在元素时戳模型中, 时间信息是作为元素存放的, 消除冗余后可能引起某些 <trans> 元素的删除. 如果子孙结点与祖先结点的时间信息相关性很大, 则可能导致大量 <trans> 元素的删除, 从而获得更好的压缩效果. 对两类查询操作的影响与属性时戳模型类似.

3 文本前缀时戳模型

文本前缀时戳模型没有引入新的变量, 只是在元素值的前面加上时间信息作为前缀.

3.1 模型定义

文本前缀时戳模型的一般形式为:

```
<element 0>
  element 0-end-time_ element 0-start-time
  <element 1_1>(element 1_1-end-time_ element 1_1-start-time)Element 1_1 value </element 1_1>
  <element 1_2>(element 1_2-end-time_ element 1_2-start-time)Element 1_2 value </element 1_2>
  ...
  <element 1_j>(element 1_j-end-time_ element 1_j-start-time)Element 1_j value </element 1_j>
  ...
</element 0>
```

并且满足如下条件(其中, i, j, k 是正整数.):

(1) element i_j value(i 表示层次, j 表示兄弟序号)可能的两种取值:

- A. 文本, 则 element i_j 在树结构中为叶子结点;
- B. 元素, 其孩子结点(一个或多个)形式如下:

```
<element i+1_k>(element i+1_k-end-time_ element i+1_k-start-time)Element i+1_k value</element i+1_k>
```

(2) 与 1.1 中条件(2)类似.

3.2 时间冗余信息的消除

文本前缀时戳模型中, 由于没有引入新的属性和元素, 得到的数据文件在没有消除时间冗余信息的情况下最小. 利用文档中元素的层次关系以及时间信息的相关性, 同样可以消除某些结点中多余的时间信息.

由于文本前缀时戳模型的数据文件相对最小, Q_1 类查询的效率最高, 经消除时间冗余信息后还能进一步提高其查询效率, 可能增加的几次比较操作对查询性能的影响同样较小. Q_2 类查询在前面两种模型的基础上有了进一步的改善.

4 结果与结论

实验在 CPU 为 Pentium dualE2160(1 8G)的机器上进行, 系统为 W inXP Professional 开发环境为 . NET2003中的 C#, 实验数据来自于 TimeCenter的职工数据库.

表 1是实验结果. 未消除时间冗余信息时, 文本前缀时戳模型的平均文件长度明显减小. 消除时间冗余信息后, 文本前缀时戳模型仍然具有最小的文件长度. 从平均压缩比来看, 属性时戳模型最好, 元素时戳模型次之, 文本前缀时戳模型最大. 但文中计算压缩比的方法是首先计算每个文件的压缩比, 再计算其平均值, 考虑到文本前缀时戳模型的原始文件长度已经最小这一情况, 就 平均压缩比 而言, 文本前缀时戳模型同样具有最好的性能.

表 1 3种模型文件大小比较
Table 1 The comparison of document length for the three models

	平均文件长度 /k	消除时间冗余信息后 文件平均长度 /k	平均压缩比 %
属性时戳模型	23 46	16 36	69 98
元素时戳模型	22 69	16 56	72 87
文本前缀时戳模型	17 71	13 50	76 12

在数据模型的基础上, 进一步的研究工作包括: 以时间为基准的时态数据的分页存储策略、查询优化及时态 XML索引技术、时态 XML数据模型的安全机制、时空 XML数据模型等方面^[9].

[参考文献] (References)

[1] Wayne Randolph, Dany King. Scenario generation: XML to the rescue[C] // 2002 SPRING SW. Orlando, USA: SISO, 2002. 02S-SW-075.

[2] Gultekin Zsoyoglu, Richard T Snodgrass. Temporal and real-time database a survey[J]. IEEE Transactions on Knowledge and Data Engineering, 1995, 7(4): 513-532.

[3] Clifford J, Croker A, Grandi E, et al. On temporal grouping[C] // Proceedings of the International Workshop on Temporal Databases. Recent Advances in Temporal Databases. London, UK: Springer Verlag, 1995: 194-213.

[4] Richard Snodgrass. The temporal query language tqrel[J]. ACM Transactions on Database Systems, 1987, 12(2): 247-298.

[5] Fusheng Wang, Carlo Zaniolo, Xin Zhou. Temporal XML SQL strikes back [C] // Proceedings of the 12th International Symposium on Temporal Representation and Reasoning. Vermont, USA: IEEE Computer Society, 2005: 47-55.

[6] Fusheng Wang, Xin Zhou, Carlo Zaniolo. Temporal information management using XML[J]. ER, 2004, 3288: 858-859.

[7] Fusheng Wang, Xin Zhou, Carlo Zaniolo. Using XML to build efficient transaction-time database systems on relational databases[C] // Proc of EDE 2006. Atlanta, USA: IEEE Computer Society, 2006: 131-134.

[8] Grandi E, Mandreoli E. The valid web: an XML/XSL infrastructure for temporal management of web documents[C] // ADVIS, 2000, 1909: 294-303.

[9] 叶小平, 陈铠原, 汤庸, 等. 时态 XML索引技术[J]. 计算机学报, 2007, 30(7): 1 074-1 085.
Ye Xiaoping, Chen Kaiyuan, Tang Yong, et al. Technology on temporal XML indexing[J]. Chinese Journal of Computers, 2007, 30(7): 1 074-1 085. (in Chinese)

[责任编辑: 刘 健]