第 36 卷 第 9 期 计 算 机 工 程 2010 年 5 月
Vol.36 No.9 Computer Engineering May 2010
·软件技术与数据库· 文章编号：1000—3428(2010)09—0077—03 文献标识码：A 中图分类号：TP311

基于 CB+-tree 的时态 XML 索引

徐海燕，蒋夏军
(南京航空航天大学信息科学与技术学院，南京 210016)

摘 要：针对时态查询与时间属性紧密相关的特点，利用时间区间作为改进后 B+-tree 的索引关键字建立索引，改进后的 B+-tree 命名为 Changing B+-tree(CB+-tree)。实验证明，在 CB+-tree 上进行时态查询比 B+-tree 及基于 DOM 的 XML 文档的查询效率有所提高。
关键词：时态 XML；B+-tree 索引；数据模型；时态查询

Temporal XML Index Based on CB+-tree

XU Hai-yan, JIANG Xia-jun
(College of Information Science & Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

【Abstract】Focused on the close relative between temporal queries and time attribute, this paper uses temporal interval as an index key of the modified B+-tree to create indexes. The modified B+-tree is denoted Changing B+-tree(CB+-tree). Through the experiments, new index method processes several typical queries in temporal XML better than querying in B+-tree index and DOM method.
【Key words】temporal XML; B+-tree index; data model; temporal queries

1 概述

XML(eXtensible Markup Language)已成为 Web 上信息表示与交换的一个标准数据格式。随着 XML 数据的大量出现，如何有效地存储、索引和查询 XML 数据成为当前研究的热点。而时态 XML 文档很少是静态的，往往随着时间的变化而变化，提高时态 XML 的查询效率具有很高的价值。

目前对于非时态的 XML 文档的索引已经有了比较多的研究成果，根据索引的方法可以分 3 大类[1]：基于路径的索引，基于串的索引和基于结点的索引。Nrvag K 研究利用 V2 时态文档数据库系统实现时态文档的存取和查找，该索引仅仅针对某些特殊情况提高了查询效率而且维护代价也高。文献[2]提出了 TempIndex 索引机制，它包含 3 种结构：时态模式，时态深度表，cp 和 cp+value 表。文献[3]提出了一种时态 XML 查询数据模型 TXQDM 和时态 XML 索引数据模型 TXIDM。文献[4]基于扩展的 XPath 模型，定义了一种时间连通性的等价关系，并且在该关系的等价类的基础上建立索引。

以上的索引方法大多是借助于非时态 XML 的索引方法，然后加入时间的处理，并没有针对时态查询的特点和时态 XML 的特征来考虑索引，这些索引不能高效地处理典型的时态查询。

以时间区间作为索引的关键字，提出 Changing B+-tree (CB+-tree)索引。该索引关键字中包含实体信息的地址指针，地址用于实体的快速定位。CB+-tree 索引主要用于处理实体轨迹查询、快照查询和时间段查询，效率优于 DOM 方法和 B+-tree 索引查询。

2 时态XML

XML 文档可以建模为一个基于 XPath 的有序而且具有边标记的结点层次图形结构。为了将文档本身作为数据进行查询，以这种结点层次结构为基础，建立起相应的 XML 查询数据模型。时态查询 XML 数据模型则是在常规模型上添加时态信息。

时态 XML 文档是具有时态标签的 XML 文档，时态标签通常分为有效时间和事务时间，这些标签在 XML 数据库中是以属性或元素的方式出现的。时态 XML 文档的片段代码如下：

```
<employee tend="1993-08-22" tstart="1992-09-19">
<empno tend="1993-08-22" tstart="1992-09-19">10 015</empno>
<firstname tend="1993-08-22" tstart="1992-09-19"> Guoxiang </firstname>
…
<salary tend="1993-08-22" tstart="1992-09-19">40 000</salary>
</employee>
```

当时态标签以属性的形式表示时称为属性时戳模型，其一般形式为：

```
<element0 tend="element0-end-time" tstart="element0-start-time">
<element1_1 tend="element1_1-end-time" tstart="element1_1-start-time">
Element1_1 value</element1_1>
<element1_2 tend="element1_2-end-time" tstart="element1_2-start-time">
Element1_2 value</element1_2>
…
<element1_j tend="element1_j-end-time" tstart="element1_j-start-time">
Element1_j value</element1_j>
```

这里的事务时间区间是用[tend, tstart]表示的，其中，tend 和 tstart 分别表示事务的结束和起始时间。

基金项目：南京航空航天大学引进人才科研基金资助项目(S0677-042)
作者简介：徐海燕(1984－)，女，硕士，主研方向：时空数据库索引；蒋夏军，讲师、博士
收稿日期：2009-12-07 **E-mail：**xhy_309@163.com

XML 文档模型中结点之间具有一定层次关系，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。当需要取该元素的属性时，缺失的时间属性可以回溯到父结点中获取。未建立索引之前 XML 文档大小对查询效率的影响比较大，所以首先利用该特点将 XML 数据进行冗余消除。

3 时态XML文档的CB+-tree索引

利用 B+-tree 对时态 XML 文档进行索引时态查询效果不是很好，为了进一步提高典型的时态查询的效率，提出新的索引方法 CB+-tree。

3.1 CB+-tree原理

CB+-tree 是对 B+-tree 的改进，CB+-tree 的定义如下：

(1)CB+-tree 是一个平衡的树。

(2)根结点至少有 2 个孩子，最多有 m 个孩子(除非是叶结点)。

(3)包含 m 个关键字的中间结点有 m+1 个指针 $p_j(0\leqslant j\leqslant m)$，每个指针指向树中下一层次的孩子结点，$p_i$ 所指向的结点中的关键字都小于 k_i 而 p_{i+1} 所指向的结点中的关键字都大于或等于 k_i。

(4)叶子结点中的第 1 种指针是用于指向同一层次结点，除有指向同一层次下一个结点的后向指针 m_pNextNode，还有指向前一个结点的前向指针 m_pPrevNode，即叶子结点之间形成一个双向链表如图 1 所示。

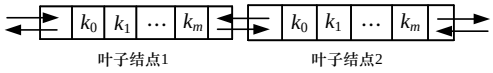

图 1 双向链表

(5)叶子结点中包含关键字信息为(id, addr, length, filepath)(实体轨迹查询)或(tend, tstart, id, addr, length, filename)(快照和时间段查询)。其中，id 是实体号；addr 为实体在文档中的地址；length 为实体的信息长度；filepath 为文档的路径；tend 和 tstart 分别为实体的事务时间的结束时刻和开始时刻。

(6)叶子结点包含的第 2 种指针，该类指针不是指向数据或下一个结点而是指向处理重复键值的查询链表(QList)，每个叶子结点中包含一个指针数组 m_pointers[m]，添加 QList 之后的叶子结点如图 2 所示。

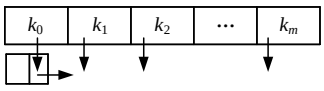

图 2 添加 QList 之后的叶子结点

该链表用于处理重复键值，当出现插入的键值与叶子结点中的某个键值相同时就在其后面分配一个链表来存储刚插入的新的键值，且按照 tend 从大到小插入。该链表将键值相同的信息聚簇在一起便于查询，其次减少索引结点的分裂次数和索引的深度从某种程度上可以提高查询的效率。

(7)叶子结点中的第 3 种指针 MPointer 指向的仍然是链表，不过该链表的作用是将该叶子结点中(除 m_pointer[m]链表中的数据)所有的数据按照 tend 从大到小依次插入单链表中，链表的作用是当快照或时间段查询时可以减少与 tend 的比较时间以提高查询效率。

通过定义，CB+-tree 本质上仍然带有复杂关键字 B+-tree，它的叶子结点中的前向指针和后向指针有利于进行快照查询和时间段查询。

3.2 基于CB+-tree的查询

针对时态数据库中比较典型的 3 类查询[5]：实体轨迹查询，快照查询和时间段查询，采用职工数据集作为实验数据，定义 3 种查询。

(1)Q1：实体轨迹查询——查询给定职工号的某个职工的所有历史信息和当前信息。

(2)Q2：快照查询——查询给定时刻所有职工的信息。

(3)Q3：时间段查询——查询给定时间段的所有实体的信息。

3.2.1 实体轨迹查询

实体轨迹查询以实体号 id 作为索引关键字。

首先根据给出的查询 id 到 CB+-tree 中去查找，当查找到符合条件的实体后，直接通过 addr 定位到实体在文档中的位置，并根据 length 准确地将实体的全部信息取出。

3.2.2 快照和时间段查询

以下建立的索引可以同时满足时间段查询和快照查询。因为时间段和快照查询都跟时间密切相关，所以选择使用实体事务时间的 tstart 作为索引关键字。

快照查询的过程：用户给出快照时间 snapshot，以 snapshot 与 CB+-tree 中的中间结点进行比较，找到第 1 个 tstart 等于或大于 snapshot 的关键字，沿着该关键字的左指针下的结点继续比较查找直到叶子结点，则该叶子结点中的部分关键字的 tstart 及其之前的所有叶子结点中的 tstart 都是满足条件的，而沿着叶子结点中的前向指针可以查询到所有满足条件的叶子结点，而当前叶子结点之后的都是不满足条件的。到达叶子结点后并不在叶子结点中进行查询而是继续到叶子结点的查询链表中查询，因为查询链表中数据按照 tend 从大到小进行了排序，与 B+-tree 索引相比可以降低比较时间。

时间段查询的过程：时间段查询与快照查询的过程类似，找到 tstart 满足条件的实体之后，将给出的时间段的结束时间与各实体的 tend 再进行比较，不同的是快照查询直接用快照时间与 tend 进行二次筛选，而时间段查询是将给出的时间段的结束时间与 tend 进行比较。

实体轨迹查询、快照和时间段查询都是通过实体地址随机定位到实体信息，将实体的全部信息取出。

4 实验结果与结论

实验环境：CPU 为 Intel Pentium D 930(1 GHz)，系统为 Windows XP Professional，生成和查询 XML 文档的开发环境为.NET2005 中的 C#，实验数据来自于 TimeCenter 的职工数据库。该数据库的 3 个文件夹共有 15 004 个 XML 文件、300 000 个职工的历史和当前信息。

4.1 实验过程

将源数据进行冗余消除。如果孩子结点的 tend 和 tstart 这 2 个时间属性与父结点的时间属性值有相同的，则孩子结点中不再存储与父结点相同的时间属性，从而消除冗余的时间信息。表 1 用于说明消除冗余前后文档的压缩情况。

表 1 合并前属性时戳模型的 XML 文件大小

平均文件长度/KB	消除时间冗余后文件平均长度/KB	平均压缩比/(%)
23.46	16.36	69.98

分别取 5 002, 10 004, 15 004 个 XML 文档进行合并，合并后大小依次为 85 MB, 169 MB 和 256 MB。

在合并后的 3 个文档上分别实现上文定义的 3 种查询。

4.2 查询效率分析

文档大小分别为 85 MB, 169 MB 和 254 MB，基于 DOM

的实体轨迹查询时间分别为 7 804 ms, 15 719 ms 和 67 490 ms，而基于 B+-tree 和 CB+-tree 的查询时间几乎为 0。由此可知，随着 XML 文档的增大基于 CB+-tree 的实体轨迹查询效率的提高更显著。表 2、图 3~图 6 表明，对于快照查询和时间段查询，XML 文档越大，基于 CB+-tree 的查询比基于 DOM 的查询效率提高得越显著。图 3 的快照时间为 1980-02-20，图 4 的快照时间为 1980-03-10，图 5 的时间段为[1985-02-20, 1985-06-01]，图 6 的时间段为[1985-03-10, 1985-04-10]。查询结果比较小时效率提高得更明显。

表 2　时间段和快照查询结果

文档大小/MB	实体总数	时间段	快照时间	满足条件的实体数(Q3)	满足条件的实体数(Q2)
85	100 024	[1985-02-20, 1985-06-01]	1985-02-20	358	360
		[1985-03-10, 1985-04-10]	1985-03-10	691	692
169	200 024	[1985-02-20, 1985-06-01]	1985-02-20	701	703
		[1985-03-10, 1985-04-10]	1985-03-10	1 366	1 367
254	300 024	[1985-02-20, 1985-06-01]	1985-02-20	1 067	1 069
		[1985-03-10, 1985-04-10]	1985-03-10	2 048	2 050

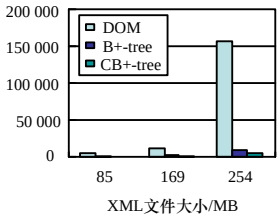

图 3　快照查询 1(Q2)

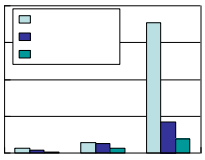

图 4　快照查询 2(Q2)

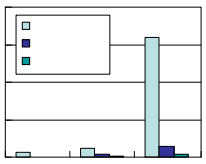

图 5　时间段查询 1(Q3)

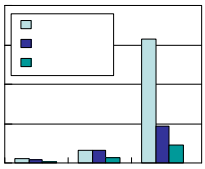

图 6　时间段查询 2(Q3)

基于 CB+-tree 索引比基于 B+-tree 的查询效率也有所提高，由于本数据集的 tend 时间几乎都满足查询条件，因此使用 B+-tree 与 CB+-tree 的比较效果还不是很显著。基于 DOM 模型的思想是把 XML 文档整个调入内存，查询是基于整个文档的遍历，这样当处理数据量大的 XML 的时候，其性能往往较差，因为存在大量无谓的数据遍历和磁盘访问的问题。基于 B+-tree 查询减少对 XML 文档的遍历，但是 B+-tree 未对实体的 tend 及重复键值进行处理，所以快照查询和时间段查询需要跟 tstart 满足条件的所有关键字的 tend 继续比较，而 CB+-tree 中的 MPointer 和 m_pointer[*m*]链表对 tend 进行了从大到小的排序，解决了这个缺陷，进一步减少了大量的遍历，并能准确地随机定位到实体，从而可以提高查询的效率。

5　结束语

本文提出的 CB+-tree 索引利用时态属性或实体号作为索引关键字，关键字中实体信息的地址指针方便实体的快速定位。结合理论和实验数据分析，本索引对查询结果适中的查询比 DOM 和 B+-tree 索引查询效率有明显提高，但对大结果查询仍然有待提高。下一步的工作是对 B+-tree 索引进行改进，使之适合更多查询类型，提高查询效率，并进一步考虑索引的动态更新。

参考文献

[1] Zou Qinghua, Liu Shaorong, Chu W W. Ctree: A Compact Tree for Indexing XML Data[C]//Proc. of the 6th Annual ACM International Workshop on Web Information and Data Management. [S. l.]: ACM Press, 2004: 39-46.

[2] Mendelzon A O, Rizzolo F, Vaisman A. Indexing Temporal XML Documents[C]//Proc. of the 30th International Conference on Very Large Data Bases. Toronto, Canada: [s. n.], 2004: 216-227.

[3] 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): 1074-1085.

[4] 陈丽冰，吉永杰，邓楚燕. 一种基于扩展时态 XML 模型的索引技术[J]. 微计算机信息, 2006, 22(5): 301-303.

[5] Salzberg B, Tsotras V J. A Comparison of Access Methods for Time-evolving Data[J]. ACM Computing Surveys, 1999, 31(2): 158-221.

编辑　顾逸斐

（上接第 76 页）

由实验结果可以看出，本文算法在处理多区域多代码问题且物理块数较多的情况下，显示了很好的自动调整能力。

本算法可用于对接网格、重叠网格和拼接网格。在软件开发过程中，考虑到 CFD 数值模拟问题的复杂性、要增大分块算法的灵活性、提高并行效率等因素，允许用户对部分物理块指定分配处理器数。

5　结束语

本文算法在并行求解 CFD 数值模拟问题中，可实现全自动计算空间的划分并将浮点计算量分配到处理器。实验结果表明该算法能使一些物理块数较多的多区域多代码问题取得很好的并行效果，并能提高它们的并行度。

参考文献

[1] Nicol D M, Saltz J H. An Analysis of Scattered Decomposition[J]. IEEE Transactions on Computers, 1990, 39(11): 1337-1345.

[2] Salmon J K. Parallel Hierarchical N-body Methods[D]. Los Angles, USA: California Institute of Technology, 1990.

[3] Baiardi F, Bonotti A, Ferrucci L, et al. Load Balancing by Domain Decomposition: The Bounded Neighbours Approch[C]//Proc. of the 17th European Simulation Multi Conference. Nottingham, UK: [s. n.], 2003.

[4] 卢开澄，卢华明. 图论及其应用[M]. 2 版，北京：清华大学出版社, 1995.

[5] Berger M J, Bokhari S H. A Partitioning Strategy for Nonuniform Problems on Multiprocessors[J]. IEEE Transactions on Computers, 1987, 36(5): 570-580.

[6] 刘　鑫，陆林生. 数据不规则问题并行计算的负载平衡策略的研究[J]. 计算机应用, 2004, 24(10): 108-111.

编辑　陆燕菲