第 32 卷 第 19 期 计 算 机 工 程 2006 年 10 月
Vol.32 № 19 Computer Engineering October 2006
·博士论文· 文章编号：1000—3428(2006)19—0017—03 文献标识码：A 中图分类号：TP311

# 基于时态数据类型的时态数据索引方法

丁国芳[1]，汤 庸[2]，章 云[1]
(1. 广东工业大学计算机学院，广州 510090；2. 中山大学计算机系，广州 510275)

**摘 要：**在系统中引入时态数据类型，使得在关系数据库系统中能对时态数据信息进行方便有效的管理。该文给出了一个基于时态数据类型的时态索引方法：MAP21*3B$^+$-Tree方法，通过对时态数据的各个域分别建立索引，实现双时态数据库的各种时态查询。
**关键词：**时态数据类型；时态索引；B$^+$-Tree

# Index Method of Temporal Data Based on Temporal Date Type

DING Guofang[1], TANG Yong[2], ZHANG Yun[1]
(1. Faculty of Computer, Guangdong University of Technology, Guangzhou 510090;
2. Department of Computer, Sun-Yat Sen University, Guangzhou 510275)

**【Abstract】**Adding temporal data type to system, the temporal data can be managed conveniently and effectively in relational database system. This paper presents a temporal index method based on temporal data type, named MAP21*3B$^+$-Tree, and accomplishes various temporal queries by creating temporal index for every domains of temporal data.
**【Key words】** Temporal data type; Temporal index; B$^+$-Tree

时态数据库研究经过 30 多年的发展，目前处于应用期阶段。学者们围绕时态概念模型、时态数据存储、时态索引、时态查询语言等方面作了大量的研究，取得了丰硕的成果。在实际应用上，还存在不少问题：数据冗余量大，查询效率低和数据存储问题等。

时态数据库查询种类多，而且复杂。对于只保存一维的时间信息的时态数据库，学者们提出了许多索引方法。相对而言，现有文献对双时态数据库的时态索引方法却不多见，而且在双时态数据库中，存在Now和Uc两个时间变元，使时态索引变得更加困难，常用的索引方法有时态 2R-Tree[1]、R*-Tre[e][2]、和时态间隔树[3]等。文献[4]总结了目前常见的时态索引方法，这些方法都只对部分类型的查询有效。本文提出了一种基于时态数据类型的时态索引方法：MAP21*3B$^+$ -Tree，能够实现所有的时态查询。

## 1 时态数据类型

假设：在 t0 时刻，插入记录 R1，在 t1 时刻插入 R2 和 R3，在 t2 时刻修改 R1 的值，在 t3 时刻修改 R2 的值，在 t4 删除 R2，在 t5 时刻修改 R1 的值，在 t6 时可插入 R4，在 t7 时刻修改 R1 和 R4 的值，在 t8 时刻修改 R4 的值。如图 1，圆点处表示插入、修改或删除操作。

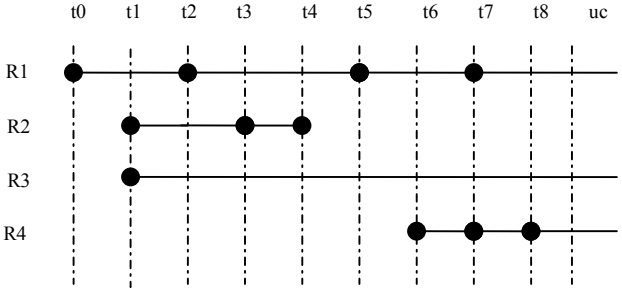



**图 1 记录操作情况**

通常的方法是用元组表示时态数据[5]：每个记录有一个时变键（surrogate）和一些时变属性；根据不同的应用，每个记录包含 1 个或 2 个时间间隔，时间间隔表示为两个属性：Start_time和End_time。可以得到如表 1，这里没有考虑有效时间的变化。

**表 1 元组表示法时态数据**

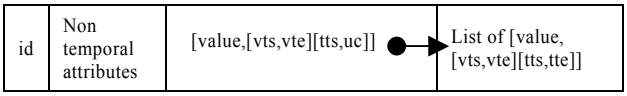
| id | CAS | TA | VTS | VTE | TTS | TTE |
|---|---|---|---|---|---|---|
| R1 | CA1 | TA1 | Ts | Te | T0 | T2 |
| R2 | CA2 | TA2 | Ts | Te | T1 | T3 |
| R3 | CA3 | TA3 | Ts | Te | T1 | UC |
| R1 | CA1 | TA11 | Ts | Te | T2 | T5 |
| R2 | CA2 | TA21 | Ts | Te | T3 | T4 |
| R1 | CA1 | TA12 | Ts | Te | T5 | T7 |
| R4 | CA4 | TA4 | Ts | Te | T6 | T7 |
| R1 | CA1 | TA13 | Ts | Te | T7 | UC |
| R4 | CA4 | TA41 | Ts | Te | T7 | T8 |
| R4 | CA4 | TA42 | Ts | Te | T8 | UC |

每次对记录更新，都会增加一条新的记录。随着时间的推移，数据的冗余量将非常大，而且时态数据的变化时间分布在不同的记录中，时态数据查询困难、效率低。

为了克服上述困难，本文采用文献[6]定义的时态数据类型来表示时态数据：TYPEDEF bt_DataType LIST of bt_DataType_Base。因此，数据记录具有如下形式，其中 ●▶

**作者简介：**丁国芳(1966－)，男，博士生、讲师，主研方向：数据库技术，模式识别，智能控制；汤 庸、章 云，教授、博导
**收稿日期：**2005-12-28 **E-mail：**dingguofang@126.com

表示链接指针：




如果该记录被删除，则数据记录成为：


对时态数据库的上述操作可得如表 2。

**表 2 当前数据和历史数据**

| id | CAS | TA（当前快照） | TA（历史数据） |
|---|---|---|---|
| R1 | CA1 | (TA13,[VTs,VTe][t7,uc]) | (TA12,[VTs,VTe][t5,t7]),<br>(TA11,[VTs,VTe][t2,t5]),<br>(TA1,[VTs,VTe][t0,t2]) |
| R2 | CA2 | | (TA21,[VTs,VTe][t3,t4]),<br>(TA2,[VTs,VTe][t1,t3]) |
| R3 | CA3 | (TA3,[VTs,VTe][t1,uc]) | |
| R4 | CA4 | (TA42,[VTs,VTe][t8,uc]) | (TA41,[VTs,VTe][t6,t7]),<br>(TA4,[VTs,VTe][t7,t8]) |

用时态数据类型描述时态数据，对时态数据的更新都在同一条记录中进行，所以时态数据的变化情况都保存在同一条记录中，因此，数据冗余大大减少，同时也有利于对时态数据查询。

## 2 MAP21*3B$^+$-Tree索引结构模型

对双时态数据库的时态数据的查询有多种形式，它们分别满足不同的实际应用的查询需求。双时态数据的所有查询分类如表 3。

**表 3 双时态数据查询分类**

| VALUE | VALID TIME | TRANS TIME |
|---|---|---|
| V/R/* | R/P/* | R/P/* |

其中，“ * ”表示所有（值或时间），“ V ”表示数据值，“ P ”表示时间点，“ R ”表示范围（值或时间）。

为了实现对时态数据的各种可能的查询，本文给出了MAP21*3B$^+$ - Tree模型，如图 2 所示。

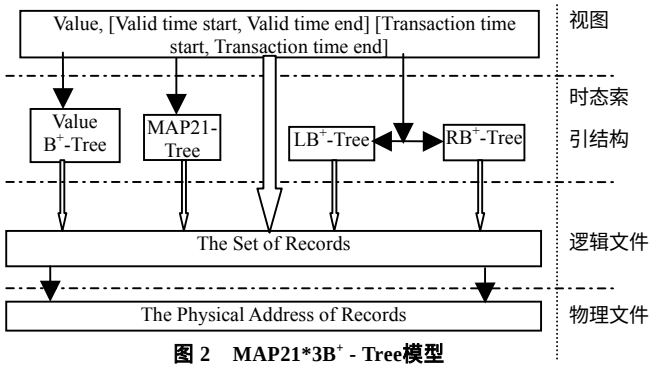



**图 2 MAP21*3B$^+$ - Tree模型**

## 3 MAP21*3B$^+$ - Tree索引结构操作

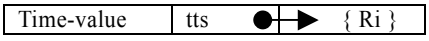
插入、修改和删除操作会更新模型中的索引结构部分，即Value B$^+$-Tree、MAP21-Tree和LRB$^+$-Tree。

### 3.1 Value B$^+$-Tree 操作

对时态属性值用B$^+$树进行索引，当插入记录或修改记录时，该值就被插入B$^+$树，在树的叶结点，保存该值所属的记录号。在结点中增加一个域，保存该操作的事务时间，叶结点结构如下：

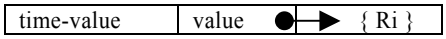

### 3.2 MAP21-Tree 操作

对时态属性的有效时间的索引，采用MAP21 树[7]。当插入记录或修改记录时，有效时间被映射为一个值插入到MAP21 树，当有效时间VTe<Now时，对MAP21 树进行操作，如果VTe=Now，则对OET树进行操作。在结点中增加一个域，保存该操作的事务时间。同样，为了保存记录有效时间的变化，在叶结点中保存记录号。叶结点结构如下：




### 3.3 LRB$^+$-Tree操作

对时态属性的事务时间的索引，分为两个部分：(1) 当前数据库快照（保存事务时间Vte=UC的记录），以事务的开始时间vts为索引值，建立一个B$^+$树（称为LB$^+$树），树的叶结点指向记录集；(2) 历史数据库快照（保存事务时间Vte<UC的记录），当删除记录或修改记录时，将该值从LB$^+$树删除，插入到RB$^+$树。例如，对表 1 中记录R1：t0 时刻，LB$^+$树中插入t0；在t2 时刻，在LB$^+$树中的插入t2 结点，同时将t0 结点删除，并插入到RB$^+$树中；t5 时刻，在LB$^+$树中插入t5 结点，同时将t2 结点删除，并插入到RB$^+$树中；同样，在t7 时刻，在LB$^+$树中插入t7 结点，同时将t5 结点删除，并插入到RB$^+$树中。为了保存时态数据值的变化，在叶结点中增加一个域，保存当前时刻的数据值。叶结点结构如下：




经过图 1 所示的操作后，MAP21*3B$^+$-Tree的索引结构图 3 所示。

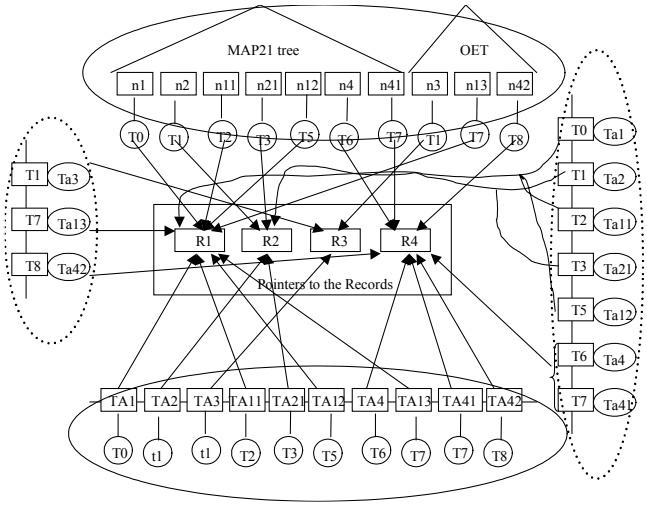



**图 3 MAP21*3B$^+$-Tree的索引结构示意图**

图 3 中上边虚线所框的表示MAP21-Tree，假定未被删除的记录的有效时间是开区间，即VTE=NOW，它们构成OET树，叶结点中的值是VTS；而被删除的记录的有效时间是闭区间，即VTE〈NOW，它们构成MAP21 树，叶结点中的值是VTS*10$^a$+VTE，其中a是满足VTS*10$^a$+VTE足够大，能表示具体应用所表示的最大值。图 3 中下面的虚线框表示B$^+$-Tree的叶结点，其值是时态属性的数据值。图 3 中左右两边的虚线框表示LB$^+$树和RB$^+$树的叶结点，结点中的值是该记录的事务时间值。所有叶结点都有指向该记录的指针，为了方便查询和提高查询效率，在叶结点中也保存相关的数据。

## 4 MAP21*3B$^+$ - Tree查询

### 4.1 单一条件查询

双时态数据库的所有时态查询如表 3 所示，显然，对于纯粹的Value、Valid time或Transaction time查询，分别对Valid B$^+$-Tree、MAP21-Tree或LRB$^+$-Tree查询，即可得到查询 结

果。

(1) 数据值查询，即 V/*/*查询，查找值等于 V 的所有记录，查找 Valid B+-Tree 即可。例如“ find the id's of all the records which the TA.value=V ”；

(2) 有效时间查询，即*/P/*查询，查找有效时间等于 P 的所有记录，例如“ find the id's of all the records which the P∈TA.valid ”，查找 MAP21-Tree 即可；

(3) 事务时间的查询，即*/*/P查询，查找LRB$^+$-Tree树。首先查找LB$^+$树，查找tts<t的节点，直到完成对LB$^+$树的查找；接着查找RB$^+$树，RB$^+$树的查询要稍微复杂一些，算法如下：

1)查找 tts<t 的叶结点，如果没找到，转 3)；否则：

2)设找到的记录号为 i，沿着叶结点链向后查找指向记录号为 i 的叶结点，直到最后，得到结点 j，如果结点 j 的 tts〉t，则记录 i 为满足条件的记录；

3)如果查找RB$^+$-Tree未完成，转 1)；否则，查询结束。

### 4.2 多条件查询

如果包含多个查询条件，只需将单个条件查询的结果进行集合运算即可。

例如，V/R/P 的查询，即查询在 P 时刻，值等于 V，有效时间范围为 R 的记录。算法如下：

(1) 在Value B$^+$-Tree查找值为V的记录，如果没找到，转(5)；否则：

(2) 设找到的记录号为 i，事务时间为 vts，如果 vts 大于 P，转(5)；否则：

(3) 沿着叶结点链向后查找指向记录号为 i 的节点，如果没找到，则把 i 添加到 SV 集中，转(5)；否则：

(4) 如果找到一个叶结点 j，可以得到 vts'，如果 vts'大于 P，把 i 添加到 SV 集中；

(5) 如果查找Value B$^+$-Tree未完转(1)；否则，得到结果集SV；

(6) 查找 MAP21-Tree，可以得到结果集 SM；

(7) 两集合求交集即可得到满足条件的记录集。

由于在查询Value B+-Tree时，利用节点的事务时间域，确定满足时间P的记录，因此不需要查询LRB$^+$-Tree即可得到正确的查询结果。

其他形式的组合条件查询，类似地可以得到查询算法（算法略）。

## 5 性能分析

MAP21*3B$^+$-Tree方法的性能，可从几个方面来考虑：

(1) 存储代价

存储代价是指包括数据记录和存取方法结构所需的物理存储空间。基于时态数据类型的数据模型，极大地减少了记录数据冗余。设表 1 中记录的长度为L（L1+L2），其中L1表示非时态属性长度之和：L1=（len(id)+len(CAS)）；L2 表示时态属性长度：L2=len(TA)+4*len(date)，则存储代价为M1=$\sum$L*$n_i$，$n_i$表示第i条记录重复的次数。在相同的情况下，用表 2 的方法，存储代价为M2=$\sum$（L1+L2* $n_i$），可以得到：

M1-M2=$\sum$L*$n_i$-$\sum$（L1+L2* $n_i$）=$\sum$（L1+L2）*$n_i$-$\sum$（L1+L2* $n_i$）=$\sum$L1*（$n_i$-1）

同时，在MAP21*3B$^+$-Tree中，用数据冗余的方式满足各类查询，提高查询的效率，这无疑会增加索引结构的物理存储量。

(2) 更新时间代价

更新操作包括插入、删除和修改记录，因此其更新时间代价是数据库所有更新操作的时间之和。假定对数据库有 n 次更新操作，则更新代价为 n 的函数，记为 O(n)。

对数据表的每次更新操作，都会更新索引文件，因此更新代价也包括对MAP21*3B$^+$-Tree索引文件的更新。MAP21*3B$^+$-Tree模型的组成是以B$^+$-Tree为基础的，即索引文件的更新操作即是对多个B$^+$-Tree的操作。B$^+$-Tree的一次操作时间为O($\log_b{}^m$+B)，这里m表示结点数。因此，MAP21*3B$^+$-Tree的更新时间代价为多个B$^+$-Tree更新时间代价之和O(5($\log_b{}^m$+B)*n)。因此总的开销为O(n+5n($\log_b{}^m$+B))。

对B$^+$树进行插入和删除操作，会引起B$^+$树结点的分裂和合并，并可能引起连锁反应，会增加一些时间开销。而且对于LRB$^+$-Tree，会进行频繁的从LB$^+$树中删除结点，而将它插入到RB$^+$树中，这会增加更新的时间代价。

(3) 查询时间代价

由于B$^+$树是平衡树，树的深度不超过$\log_b{}^n$，因此搜索代价最坏为O($\log_b{}^n$+b)，n为数据的个数，b为节点包含的数据数。最复杂的R/R/R查询，可能会搜索 5 棵B$^+$树，查询代价最坏为 5 O($\log_b{}^n$+b)，平均为 5O(b/2+$\log_b{}^n$)。对于其他情况下的索引，平均查询时间不会超过 5O(b/2+$\log_b{}^n$)。由此可见，MAP21*3B$^+$-Tree索引方法的效率比较好。

## 6 结语

本文提出了基于时态数据类型的时态关系模型的时态数据索引模型：MAP21*3B$^+$ - Tree模型，其基本思想是分别用不同的索引结构对时态属性数据的不同数据域分别建立索引，并利用数据冗余方式提高索引的效率和灵活性。

MAP21*3B$^+$ - Tree索引结构较复杂，规模也比较大，因此实现查询的I/O操作较多，因此进一步的工作是研究MAP21*3B$^+$ - Tree的物理存储结构，减少I/O操作次数，进一步提高查询效率。

### 参考文献


1 Kim J S, Kim D H, Ryu K H. A Spatiotemporal Data and Indexing[C]. Proceedings of International Conference on Electrical and Electronic Technology, Hong Kong, China, 2001: 110-113.

2 Beckmann N, Kriegel. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles[C]. Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1990-05: 322-331.

3 Kumar A, Tsotras V J, Faloutsos C. Access Methods for Bitemporal Databases[C]. Proceedings of the International Workshop on Recent Advances in Temporal Databases. New York: Springer-Verlag, 1995: 235-254.

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

5 Lorentzos N A, Johnson R G. Extending Relational Algebra to Manipulate Temporal Data[J]. Inf. Syst., 1988, 13(3): 289-296.

6 Ding Guofang, Zhang Yun, Tang Yong. Temporal Data Type and Temporal Relation Operation[C]. Proc. of IEEE ICIA, Hong Kong, China, 2005: 204-209.

7 Nascimento M A, Dunham M H. Indexing Valid Time Databases via B$^+$-Trees[J]. IEEE Transactions on Knowledge and Engineering, 1999, 11(6): 929-947.