软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(11):2587−2601 [doi: 10.13328/j.cnki.jos.004574]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

时态拟序数据结构研究及应用*

叶小平, 汤 庸, 林衍崇, 陈钊滢, 张智博

(华南师范大学 计算机学院,广东 广州 510631)

通讯作者: 汤庸, E-mail: scnuyxp2013@163.com

摘 要: 时态数据索引是实现时态数据有效管理的关键技术之一.讨论了一种时态数据结构及其在时态数据索引上的应用.常规的时态数据管理技术多基于代数框架.提出了一种基于拟序关系的时态数据结构,该结构能够像常规关系数据那样实现“一次一集合”的数据操作,并可通过多线程提高查询效率.在此基础上,研究了一种时态数据索引TQOindex.首先,提出时间期间集合上拟序关系和线序划分概念,讨论了线序划分的最优(最小)性质和构建算法,并在最小线序划分框架内研究时态拟序结构基于增量式更新的插入和删除算法.其次,研究了时态拟序结构应用——引入基于拟序扩展集的时态数据索引 TQOindex.该索引适用于磁盘(外存)数据管理,可在常规数据库平台上有效使用.其增量式更新机制可应用于“大数据”的动态索引技术.另外,对 TQOindex 进行了基本仿真,实验结果表明了该工作的可行性和有效性.提出的时态拟序数据结构着眼于新型数据,如语义数据、XML 数据和移动对象数据中时态处理与整合机制,相应的工作具有较为广泛的应用扩展性.

关键词: 时态拟序;基于线序划分数据结构;“一次一集合”查询;增量式更新;TQOindex

中图法分类号: TP311

中文引用格式: 叶小平,汤庸,林衍崇,陈钊滢,张智博.时态拟序数据结构研究及应用.软件学报,2014,25(11):2587−2601. http://www.jos.org.cn/1000-9825/4574.htm

英文引用格式: Ye XP, Tang Y, Lin YC, Chen ZY, Zhang ZB. Study and application of temporal quasi-order data structure. Ruan Jian Xue Bao/Journal of Software, 2014,25(11):2587−2601 (in Chinese). http://www.jos.org.cn/1000-9825/4574.htm

Study and Application of Temporal Quasi-Order Data Structure

YE Xiao-Ping, TANG Yong, LIN Yan-Chong, CHEN Zhao-Ying, ZHANG Zhi-Bo

(School of Computer, South China Normal University, Guangzhou 510631, China)

Corresponding author: TANG Yong, E-mail: scnuyxp2013@163.com

Abstract: Temporal index is one of key technologies for temporal data managements. This paper discusses a temporal data structure and its application for temporal data index. Most of existing temporal data methods are based on algebra framework. This paper presents a temporal data structure using quasi-order relation which can achieve multithreading data operations, and puts forward temporal index TQOindex supported by the structure. Firstly, it proposes the conception of quasi-order relationship of temporal data and studies the construct algorithm of linear order partition and its optimal property. Secondly, it dedicates application on temporal data structure with building TQOindex on expand quasi-order set which manages data on disks and can be used in common databases platform. TQOindex can dynamically index “big data” by its querying schema of “one time, one set” as well as incremental update mechanism. Finally, the simulations in the paper show the feasibility and validity for TQOindex. In conclusion, temporal quasi-order data structure pays attention to the mechanism of the processing and integration between time and application scene information for new kinds of data such as semantic data, XML data and moving objects data, and the works in this paper offers remarkable extendibility.

Key words: temporal quasi-order; data structure based on linear order partition; querying with “one time one set”; incremental updating; TQOindex

* 基金项目: 国家高技术研究发展计划(863)(2013AA01A212); 国家自然科学基金(60970044, 61272067); 广东省自然科学基金(S2011010003409, S2012030006242)

收稿时间: 2013-05-17; 修改时间: 2013-12-05; 定稿时间: 2014-01-10

时间是客观事物发展变化的基本刻画.作为客观事物的反映,计算机数据经常需要描述和处理时间问题,这在 Web 和 e-business 等网络环境下数据管理领域更为突出.在常规数据库(如关系、对象和 XML 数据库等)中,数据不显含时间,但实际上却表示相应数据的当前状态,可看做一种“快照”数据.随着数据库应用领域的扩展和数据库技术研究的深入,在管理事物过去状态和预测未来发展应用驱动之下,需将数据时间属性显式地表示并有效处理.带有时间标签的数据就是时态数据.数据查询是数据处理的基本功能,而时间本身特有的性质,如单向性(单调递增)、多维性(有效、事务和用户时间维等)和相互关系复杂性(ALLEN 时间关系[1])等,使得时态数据难以纳入传统关系数据的处理框架,时态数据索引就成为实现数据查询的基本途径.根据我们所掌握的资料,现有的时态数据索引研究主要有以下 3 种情形:

(1) 数据时态和非时态部分依次处理.这主要体现在时态关系数据查询方面[2−7].基本思想是:建立一套基于时间(区间)的索引机制,先对数据进行时间处理,再对筛选后数据进行常规处理.这种方式的特点是:基于时间本身的属性研究时态查询技术,但实现前提是要有成熟的常规数据查询技术作为支撑,使得“依次处理”能够有效地实现.

(2) 时态处理纳入非时态处理框架.这主要体现在时空数据库查询方面[8−15].其基本思想是:将时间看作一个新的空间维,例如,将一维时间和二维空间属性的数据看作三维空间数据.这种方式着眼于时态所依附的非时态数据空间特征,能够有效借鉴空间数据索引技术,但不易体现时间本身不同于空间的基本特性.

(3) 时态处理整合到非时态处理过程当中.这主要体现在时态 XML 数据和移动对象数据查询方面[16−22].其基本思想是:针对数据本身的特征,例如 XML 的结构特征和移动对象的轨迹线特征等,研究相应基于时间特性的时态索引机制;然后,将其整合到相应的非时态查询当中.这种方式不同于情形(1)的“依次处理”和情形(2)的“整体归结”而凸现时间本身的特征以及着眼于时态与非时态数据的内在关联与制约.

实际上,随着应用中所涉及的数据范围的日益扩展(时态对象数据的时态关联模型、时态 XML 的半结构化模型、移动对象数据的轨迹线模型等),基于情形(3)的时态数据索引模式已成为人们关注的研究课题.此时需要探讨两个基本问题:一是能够较好地体现时间本身特征的一般时态数据结构和时态索引框架;二是时态查询与非时态查询的整合机制[21,22].

本文的主要贡献是:

- 提出一种基于拟序关系的时态数据结构,其特征是按照时间期间集合组织时态数据,通过线序划分形成一种新的结构体系,并可有效地应用于各类新型时态数据,因此具有良好的拓展空间;
- 同时,由于时态数据建立在“划分”的数学基础之上,能够像关系数据那样实现“一次一集合”的查询和多线程操作,适合于分布式以及 P2P 等网络环境下的数据管理,具有理想的处理效率.

本文的另一项工作是在时态拟序数据结构框架内,通过讨论线序分枝与时间数序列间映射关系,建立了基于外存的时态拟序数据索引 TQOindex.增量式更新是海量时态数据管理中一项基本挑战,TQOindex 具有关于拟序结构增量式更新机制,可以实现索引的动态管理;同时,TQOindex 采用基于 B+树的时间数据磁盘存储管理,能够在各种常规数据库平台上使用,具有实际应用的可操作性和扩展性.

本文第 1 节引入时态拟序关系概念,研究最小线序构建算法与基本性质,建立了基于拟序关系的时态数据结构,讨论了线序划分的查询与删除技术.第 2 节通过拟序数据结构建立时态索引 TQOindex,采用线序分枝与时间期间数序列一一映射的机制实现了相应数据查询.第 3 节对 TQOindex 进行仿真评估,实验结果表明了 TQOindex 的可行性和有效性.

1 时态拟序数据结构

1.1 LOB与LOP

时态数据(temporal data)是二元组 $Td=\langle D,Tstamp\rangle$,其中,$D$ 是非时态数据;$Tstamp$ 是时间标签,不失一般性,本文假设 $Tstamp$ 为有效时间期间 VT(valid time period),$VT=[VTs,VTe)$,VTs 和 VTe 分别表示 VT 时间始点和终点($VTs\leqslant VTe$);若 $VTs=VTe$,则定义 $VT=[VTs,VTe)$为时刻(instant).设 Td 是时态数据,Td 的有效时间期间记为 $VT(Td)$.

集合 E 上满足自反性和传递性的关系 R 称为 E 上的一个拟序(quasi-order).

定义 1(时态拟序). 设 E 是时态数据集合,定义 E 上关系$\preccurlyeq$:$Td_1,Td_2\in E,Td_1\preccurlyeq Td_2\Leftrightarrow VT(Td_1)\subseteq VT(Td_2)$,则"$\preccurlyeq$"是 E 上的一个拟序.

设Γ是时间期间集合.$\forall u\in\Gamma,u=[VTs,Vte)$,则 u 在 VTs-VTe 平面上对应点 $P(u)=(VTs,VTe)$,可验证这是 1-1 对应,称 $P(u)=[VTs,VTe)$为 u 对应的(二维)时间点(2-dimension time point).此时,u 对应 VTs-VTe 平面上一个点 $P(u)$,Γ 对应 VTs-VTe 平面上一个点集 $P(\Gamma)$.本文以下将不区分Γ和 $P(\Gamma)$.

设 $P_0=(\min\{VTs(P)\},\max\{VTe(P)\}),P\in\Gamma$.由 P_0 为始点,由 $P(\Gamma)$得到的"从上到下"再"从左到右"的遍历序列称为 $P(\Gamma)$的深度优先序列.

例 1:深度优先序列实例 $S(\Gamma)$如图 1 所示.

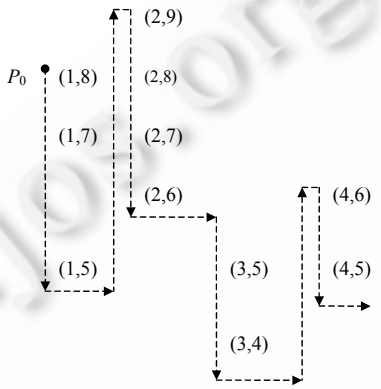

Fig.1 Depth-Prior sequence $S(\Gamma)$

图 1 深度优先序列 $S(\Gamma)$

定义 2(线序分枝与线序划分). 设Γ具拟序"$\preccurlyeq$"的时间期间集合.Γ的一个全序分枝为Γ的一个线序分枝(linear order branch,简称 LOB 或 L).记Ω是Γ上所有 LOB 的集合,若$\forall LOB_i,LOB_j\in\Omega,i\neq j,LOB_i\cap LOB_j=\varnothing$,且$\bigcup LOB=\Gamma$,则称$\Omega$是$\Gamma$上的一个线序划分(linear order partition,简称 LOP),并记为 $LOP(\Gamma)$.

算法 1. LOP 下优先算法.

输入:深度优先序列 $S(\Gamma)$,u_0 为 $S(\Gamma)$中首元素,$LOP=LOB=\varnothing,i=0,j=0,k=1$.

输出:$LOP(\Gamma)$.

Step 1. 如果$|(S(\Gamma)|=0$,转 Step 4;否则,do {将 u_i 放入 LOB;$S(\Gamma)=S(\Gamma)\backslash u_i$;} while $(i<|(S(\Gamma)|-1\wedge Vs(u_{i+1})=Vs(u_i))$.

Step 2. 令 $j=i$; while $(j<|(S(\Gamma)|-1\wedge Ve(u_i)<Ve(u_j))\{j++;\}$ 如果$(j!=|(S(\Gamma)|-1)$,令 $i=j$,转 Step 1;否则,转 Step 3.

Step 3. 令 $LOB_k=LOB$;将 LOB_k 放入 LOP; $LOB=\varnothing$; $k++$; $i=0$,返回 Step 1.

Step 4. 返回 $LOP(\Gamma)=LOP=\{LOB_1,LOB_2,\ldots\}$.

设 $Vs(\Gamma)=\max\{Vs(u)|u\in\Gamma\},Ve(\Gamma)=\max\{Ve(u)|u\in\Gamma\}$,当$\Gamma=\{u|0\leqslant Vs(u)\leqslant Vs(\Gamma)\leqslant Ve(u)\leqslant Ve(\Gamma)\}$,此时,算法 1 最大时间复杂度为$(Vs(\Gamma)\times Ve(\Gamma))/2$.

例 2:对于例 1 中 $S(\Gamma)=\langle[1,8],[1,7],[1,5],[2,9],[2,8],[2,7],[2,6],[3,5],[3,4],[4,6],[4,5]\rangle$,算法 1 的实现如图 2 所示,得到如下两条 LOB:$LOB_1=\langle[1,8],[1,7],[1,5],[3,5],[3,4]\rangle]$,$LOB_2=\langle[2,9],[2,8],[2,7],[2,6],[4,6],[4,5]\rangle$.

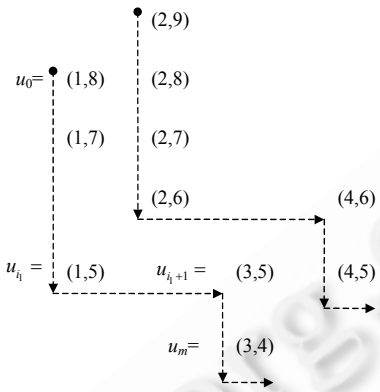

Fig.2 Implementing down prior algorithm

图 2 下优先算法实现

1.2 MLOP和EMLOP

1.2.1 MLOP

定义 3(最小线序分枝 MLOP). 设 LOP_0 是Γ上的线序划分,如果对于Γ上任意 LOP 都有$|LOP_0|\leqslant|LOP|$,则称 LOP_0 是Γ上的最小线序划分(minimum linear order partition,简称 MLOP).

定义 4(时序矩阵). 设 $i_1=\min_\Gamma\{VTs(u)\}$,$i_n=\max_\Gamma\{VTs(u)\}$,$j_1=\min_\Gamma\{VTe(u)\}$,$j_m=\max_\Gamma\{VTe(u)\}$.$\Gamma$中时间期间 $u=[Vs,Ve]=[i,j]$,简记为 $i_1j(i_1\leqslant i\leqslant i_n,j_1\leqslant j\leqslant j_m)$.设平面上横轴和纵轴分别表示$[i,j]$的始点和终点,由$\{i,j|i_1\leqslant i\leqslant i_n, j_1\leqslant j\leqslant j_m\}$确定的如图 3 所示的格点集合称为基于$\Gamma$的时序矩阵并记为 $TOM(\Gamma)$.

对 $u_{i_0j_0}\in TOM(\Gamma)$,通过 $u_{i_0j_0}$ 将 $TOM(\Gamma)$分为 4 个区域:

$$UL(u_{i_0j_0})=\{v_{ij}\mid i\leqslant i_0,j_0\leqslant j\},$$
$$UR(u_{i_0j_0})=\{v_{ij}\mid i_0\leqslant i,j_0\leqslant j\},$$
$$DL(u_{i_0j_0})=\{v_{ij}\mid i\leqslant i_0,j\leqslant j_0\},$$
$$DR(u_{i_0j_0})=\{v_{ij}\mid i_0\leqslant i,j\leqslant j_0\}.$$

在上述各式中,如果只是“<”成立,则称相应区域为开区域,分别记为 OUL,OUR,ODL 和 ODR.在 $TOM(\Gamma)$中,例如时态节点“23”的“上左”区域 $UL(23)$和下右区域 $DR(23)$,如图 4 所示.

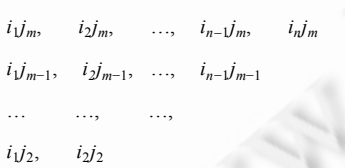

Fig.3 Quasi-Order matrix $TOM(\Gamma)$

图 3 线序矩阵 $TOM(\Gamma)$

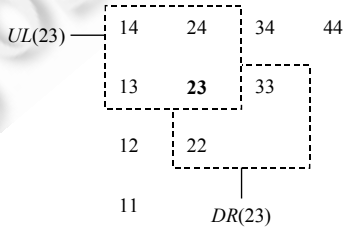

Fig.4 $UL(23)$ and $DR(23)$

图 4 $UL(23)$和 $DR(23)$

定理 1(时序矩阵与拟序). $\forall u_0\in HTM(\Gamma)$,以下各式成立:

① $u_0\subseteq v_0\Leftrightarrow v_0\in UL(u_0)$;

② $v_0\subseteq u_0\Leftrightarrow v_0\in DR(u_0)$;

③ $\neg(u_0\subseteq v_0\vee v_0\subseteq u_0)\Leftrightarrow v_0\in OUR(u_0)\vee v_0\in ODL(u_0)$.

证明:

① 设 $u_0=[i_0,j_0],v_0=[k_0,l_0],u_0\leqslant v_0\Leftrightarrow u_0\subseteq v_0\Leftrightarrow k_0\leqslant i_0,j_0\leqslant l_0\Leftrightarrow v_0\in UL(u_0)$.

同理可证②和③. □

定理 2(下优先算法基本性质). 由下优先算法(算法 1)得到的 LOP 是 MLOP.

证明:设 $LOP_0=\langle L_1,L_2,\ldots,L_n\rangle$由算法 1 得到,其中,$L_i$ 按计算顺序排序.

$$\forall u_{i_0}\in L_i(1<i\leqslant n),\exists u_{k_0}\in L_{i-1},\neg(u_{i_0}\subseteq u_{k_0}\vee u_{k_0}\subseteq u_{i_0}).$$

事实上,只需说明 L_{i-1} 中存在元素位于 $DL(u_{i_0})$. 反设这样的元素不存在,L_{i-1} 就位于 $UL(u_{i_0})$, 则 $ODR(\min L_{i-1})$ 包含 L_i,这与算法 1 中 LOB 的取法矛盾.设由此得到的节点分别为 $u_1,u_2,,\ldots,u_n$,此时就有 $u_{i-1}\in ODL(u_i)(1<i\leqslant n)$, 即$\neg(u_i\subseteq u_{i-1}\vee u_{i-1}\subseteq u_i)$.因此,任何线序划分 LOP 至少有 n 个 LOB.定理得证. □

1.2.2 EMLOP

定义 5(扩展的线序分枝 ELOB 和扩展的线序划分 EMLOP). 设 $LOB=\langle u_1\ldots,u_i,u_{i+1},\ldots,u_m\rangle,u_i$ 和 $u_{i+1}\in LOB$. 按照下述方法得到折线段称为 LOB 在 VTs-VTe 平面上 LOB 扩充(extended LOB),并记为 ELOB.

① 对于 L 中任意两个相邻的 u_i 和 u_{i+1},

- 当 $VTs(u_i)=VTs(u_{i+1})\vee VTe(u_i)=VTe(u_{i+1})$时,将 u_i 和 u_{i+1} 用线段连接;
- $\neg(VTs(u_i)=VTs(u_{i+1})\vee VTe(u_i)=VTe(u_{i+1}))$在 u_i 和 u_{i+1} 间加入点 v,其中,$v=[VTs(u_{i+1}),VTe(u_i))$,用线段连接 u_i 与 v 以及 v 与 u_{i+1}.

② 用线段连接 L 最小时间期间 $\min L=u_m$ 对应点 $P(u_m)$与 VTs-VTe 平面上对角线上点 $P(v_0)$,其中,

$$v_0=\{VTs(u_m),VTe(u_m)\}.$$

在 ELOB 中,属于Γ的点 u 称为实时间点(real instant),否则称为添加时间点(fill instant),分别记为 $u(r),u(f)$.

给定 $MLOP(\Gamma)$中,所有 LOB 对应的 ELOB 构成的集合称为 MLOP 扩充集(extended MLOP),记为 $EMLOP(\Gamma)$.

例 3:对于例 2 中的 $MLOP=\langle L_1,L_2\rangle,EMLOP=\langle ELOB_1,ELOB_2\rangle$,如图 5 所示,圆点和带阴影的节点对应 $u(f)$:

- $ELOB_1=\langle[1,9),[1,8),[1,7),[1,6),[1,5),[2,5),[3,5),[3,4),[4,4)\rangle$;
- $ELOB_2=\langle[2,9),[2,8),[2,7),[2,6),[3,6),[4,6),[4,5),[5,5)\rangle$.

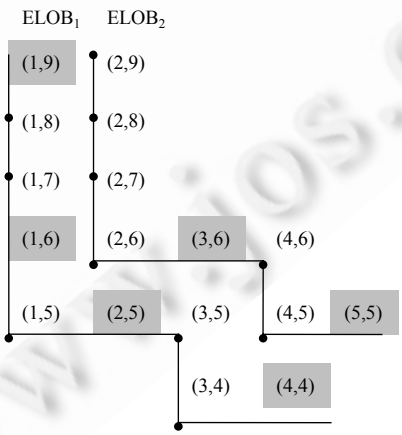

Fig.5 $EMLOP(\Gamma)$

图 5 $EMLOP(\Gamma)$

$EMLOP(\Gamma)$可与 $S(\Gamma)$同时构造.$EMLOP(\Gamma)$可用二维数组 $EMLOP[VTs][VTe]$进行存储管理,数组中元素为三元组 $N=(no,flag,[VTs,VTe])$,其中,

- no 为时间期间 $u=[VTs,VTe]$所在 ELOB 序号.
- $flag$ 用于标识 u 是否属于 MLOP,当 $u\in MLOP$ 时,$flag$=t(true);当 $u\in EMLOP\backslash MLOP$ 时,$flag$=f(false).

例如,在图 5 中,

- 若 $u=[1,5]\in ELOB_1$,则 $no(u)=1$,$u=[1,5]\in MLOP$,$flag$=t;
- 若 $v=[3,6)\in ELOB_2$,则 $no(v)=2$,$v=[3,6)\in EMLOP\backslash MLOP$,$flag$=f.

1.3 LOP动态管理

1.3.1 插入管理

定义 6(时间期间 *u* 最近邻 LOB). 设 u 为时间期间,并给定 LOB_i.若下述条件成立:

$$(\exists v\in LOB_i(u\subseteq v))\wedge(\exists w(u\subseteq w)\rightarrow(w\in LOB_i\vee w\in LOB_{i+k}))),$$

其中,$k>0$,则称 LOB_i 是 u 的最近邻 LOB.

定义 7(时间期间 *u* 上确界和下确界). 设有时间期间 u,且 $v\in LOB_0$:

- 若$(u\subseteq v)\wedge(\exists w\in LOB_0(u\subseteq w)\rightarrow v\subset w)$,则称 v 为 u 在 LOB_0 中的上确界,记为 $v=\sup LOB_0(u)$;
- 若$(v\subseteq u)\wedge(\exists w\in LOB_0(w\subseteq u)\rightarrow w\subset v)$,则称 v 为 u 在 LOB_0 中的下确界,记为 $v=\inf LOB_0(u)$.

算法 2. LOP 插入算法.

输入:Lop,插入新时间期间 u.

输出:更新后的 $LOP(\Gamma)$.

Step 1. 令 $LOP=Lop(\Gamma)$.

Step 2. 对于$\forall LOB\in LOP$,如果 sup(u)和 inf(u)都不存在:

- 若 $LOP\subseteq DL(u)$,则 $Lop(\Gamma)=Lop(\Gamma)\cup\{\langle u\rangle\}$;
- 若 $LOP\subseteq UR(u)$,则 $Lop(\Gamma)=\{\langle u\rangle\}\cup Lop(\Gamma)$,转 Step 4.

Step 3. $\exists$最近邻 $LOB_{i_0}\in LOP$,如果仅存在 sup(u):

- 若 $LOB_{i_0}\subseteq UL(u)$:

$$Lop(\Gamma)=Lop(\Gamma)\setminus\{LOB_{i_0}\},$$
$$LOB_{i_0}=LOB_{i_0}\cup\{\langle u\rangle\},$$
$$Lop(\Gamma)=Lop(\Gamma)\cup\{LOB_{i_0}\},$$

  则转 Step 6;
- 否则,$Lop(\Gamma)=Lop(\Gamma)\setminus\{LOB_{i_0}\}$,$LOB_{i_0}\cap UR(u)$作为新的插入点:

$$LOB_{i_0}=(LOB_{i_0}\setminus(LOB_{i_0}\cap UR(u)))\cup\{\langle u\rangle\},$$
$$Lop(\Gamma)=Lop(\Gamma)\cup\{LOB_{i_0}\},$$
$$Lop=Lop\setminus LOB_{i_0},$$

  对每个新插入点,依次返回 Step 2.

Step 4. $\exists$最近邻 $LOB_{i_0}\in LOP$,如果仅存在 inf(u):

- 若 $LOB_{i_0}\subseteq DR(u)$:

$$Lop(\Gamma)=Lop(\Gamma)\setminus\{LOB_{i_0}\},$$
$$LOB_{i_0}=\{\langle u\rangle\}\cup LOB_{i_0},$$
$$Lop(\Gamma)=Lop(\Gamma)\cup\{LOB_{i_0}\},$$

  则转 Step 6;
- 否则,$Lop(\Gamma)=Lop(\Gamma)\setminus\{LOB_{i_0}\}$,$LOB_{i_0}\cap UR(u)$作为新的插入点:

$$LOB_{i_0}=\{\langle u\rangle\}\cup(DR(u)\cap LOB_{i_0}),$$
$$Lop(\Gamma)=Lop(\Gamma)\cup\{LOB_{i_0}\},$$

  对每个新插入点,返回 Step 2.

Step 5. $\exists$最近邻 $LOB_{i_0}\in LOP$,如果 sup(u)和 inf(u)都存在,设 $LOB_{i_0}=\langle v_1,\ldots,u_0,\ldots,u_1,\ldots,v_m\rangle$,其中,

$$u_0=\sup(u),u_1=\inf(u).$$

Step 5.1. 如果 LOB_{i_0} 片段$\langle u_0,\ldots,u_1\rangle\backslash\{u_0,u_1\}$为空,$Lop(\Gamma)=Lop(\Gamma)\backslash\{LOB_{i_0}\}$.$LOB_{i_0}=\langle v_1,\ldots,u_0,u,u_1,\ldots,v_m\rangle$, $Lop(\Gamma)=Lop(\Gamma)\cup\{LOB_{i_0}\}$,则转 Step 6;

Step 5.2. 否则,$LOP=LOP\backslash\{LOB_1,\ldots,LOB_{i_0-1},LOB_{i_0}\}$,返回 Step 2.

Step 6. 返回 $Lop(\Gamma)$.

插入过程可能会沿 LOB 传递,最极端的情形是由首条 LOB 传递到最后一条,此时,算法 2 的时间复杂度为

$$|LOP(\Gamma)|\times\max\{|LOB|\}.$$

例 4:在如图 5 所示的 $EMLOP(\Gamma)$中插入点 u=[2,4),其最邻近 LOB 为 LOB_1,在 LOB_1 中,u_0=sup(u)=[1,5), u_1=inf(u)=[3,4].按照“下优先”得到新的 $ELOB_1'$, 如图 6 所示.此时,LOB_1 片段$\langle u_0,u_1\rangle\backslash\{u_0,u_1\}=\{[3,5]\}$.$v$=[3,5]作为新插入点继续上述过程,如图 7 所示,最终结果如图 8 所示.

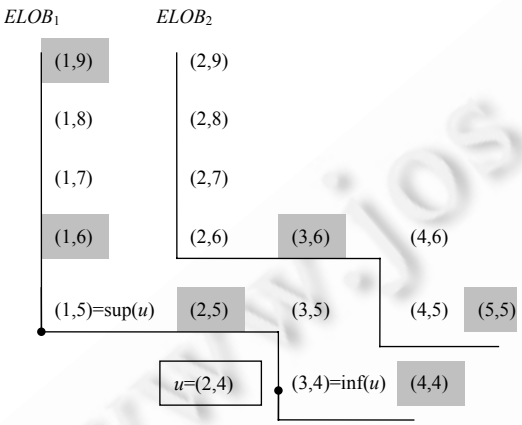

Fig.6 Inserting u=[2,4)

图 6 插入 u=[2,4)

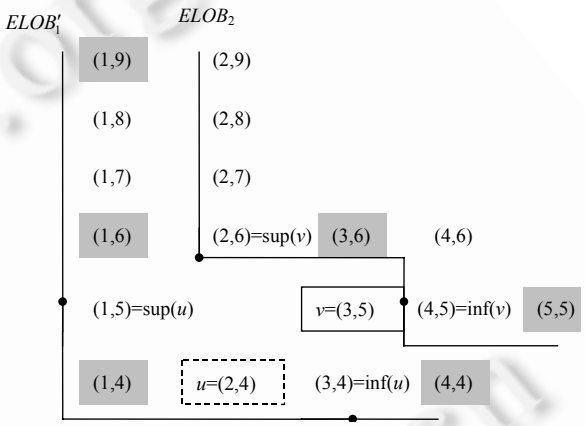

Fig.7 Getting $ELOB_1'$ after reconstruction

图 7 重构后得到的 $ELOB_1'$

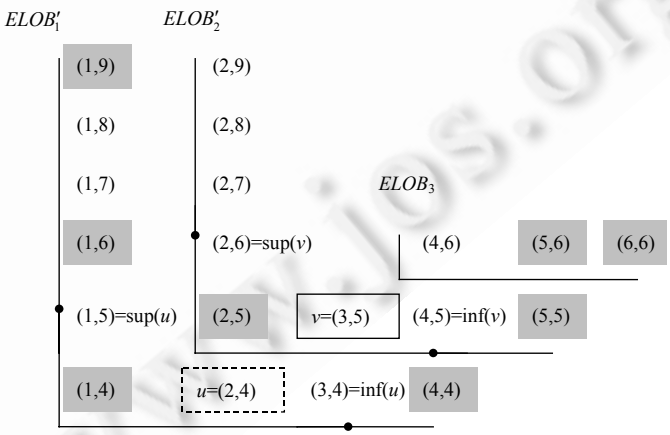

Fig.8 Getting $ELOB_1'$, $ELOB_2'$ and $ELOB_3$ after reconstruction

图 8 重构后得到 $ELOB_1'$,$ELOB_2'$ 和 $ELOB_3$

1.3.2 删除管理

算法 3. LOP 删除算法.

输入:$LOB_k=\langle v_1,\ldots,v_{i-1},v_i,v_{i+1},\ldots,v_m\rangle$,$u$ 是 LOB_k 中待删除点.

输出:更新后的 $LOP(\Gamma)$.

Step 1. 令 $top=v_1,tail=v_m,del=\{u\},Lop(\Gamma)=Lop(\Gamma)\backslash LOB_k;LOP=Lop(\Gamma)\backslash\{LOB_1,LOB_2,\ldots,LOB_k\}$.

Step 2. 如果 $u=top,LOB_k=LOB_k\backslash del$,令 top 成为 LOB_k 中第 1 个点:

Step 2.1. 如果 $UL(top)\cap LOB_{k+1}=\varnothing,Lop(\Gamma)=Lop(\Gamma)\cup LOB_k$,则转 Step 5;

Step 2.2. 如果 $UL(top)\cap LOB_{k+1}\neq\varnothing,LOB_k=(UL(top)\cap LOB_{k+1})+LOB_k,Lop(\Gamma)=Lop(\Gamma)\cup LOB_k.del=LOB_{k+1}\cap UL(top)$,令 top 成为 del 中第 1 个点,k++,返回 Step 2.

Step 3. 如果 $u=tail,LOB_k=LOB_k\backslash del$,令 $tail$ 成为 LOB_k 中最后一个点:

Step 3.1. 若 $DR(tail)\cap LOB_{k+1}=\varnothing,Lop(\Gamma)=Lop(\Gamma)\cup LOB_k$,转 Step 5;

Step 3.2. 若 $DR(tail)\cap LOB_{k+1}\neq\varnothing,LOB_k=(DR(tail)\cap LOB_{k+1})+LOB_k,Lop(\Gamma)=Lop(\Gamma)\cup LOB_k.del=LOB_{k+1}\cap DR(tail)$,令 $tail$ 成为 del 中最后一个点,k++,返回 Step 3.

Step 4. 否则,令 $w_1=\sup(u),w_2=\inf(u),Reg(w_1,w_2)=UL(w_1)\cap DR(w_2),top$ 成为 LOB_{k+1} 中第 1 个点,$tail$ 成为 LOB_{k+1} 中最后一个点,$LOB_k=LOB_k\backslash\{\langle u\rangle\}$:

Step 4.1. 若 $Reg(top,tail)\cap LOB_{k+1}=\varnothing,Lop(\Gamma)=Lop(\Gamma)\cup LOB_k$,则转 Step 5;

Step 4.2. 若$(top\in Reg(top,tail)\cap LOB_{k+1})\wedge(tail\in Reg(top,tail)\cap LOB_{k+1})$,根据算法 1,连接 LOB_k 和 LOB_{k+1},$LOB_k=LOB_k\cup LOB_{k+1},Lop(\Gamma)=Lop(\Gamma)\backslash LOB_{k+1}$,则转 Step 5;

Step 4.3. 若只存在 $top\in Reg(top,tail)\cap LOB_{k+1},LOB_k=LOB_k\cup(Reg(top,tail)\cap LOB_{k+1}),u=top,del=LOB_{k+1}\backslash(Reg(top,tail)\cap LOB_{k+1})$,$k$++,返回 Step 2;

Step 4.4. 若只存在 $tail\in Reg(top,tail)\cap LOB_{k+1},LOB_k=LOB_k\cup(Reg(top,tail)\cap LOB_{k+1}),u=tail,del=LOB_{k+1}\backslash(Reg(top,tail)\cap LOB_{k+1})$,$k$++,返回 Step 3.

Step 5. 返回 $LOP(\Gamma)$.

删除过程可能沿 LOB 传递,极端情形是由首条 LOB 传递到最后一条,此时,算法 3 时间复杂度为

$$|LOP(\Gamma)|\times\max\{|LOB|\}.$$

例 5:在如图 5 所示的 $EMLOP(\Gamma)$中删除点[1,5].如图 9 所示,$[1,5]\in ELOB_1$,在 LOB_1 中得 sup([1,5])=[1,7], inf([1,5])=[3,5].按照“下优先”连接[1,7]和[3,5]得到新的 $ELOB_1'$ 如图 9 所示. $ELOB_1'$ 的构建相当于在 LOB_2 中删除点[2,7]到[2,6]之间的一个片段,即,[2,7]和[2,6]可被看作 $EMLOP(\Gamma)\backslash ELOB_1'$ 中新的删除点.在 $ELOB_2$ 中, sup([2,7])=[2,8],inf([3,6])=[4,6].按照算法 3,连接[2,8]和[4,6]得到新的 EMLOP,如图 10 所示.

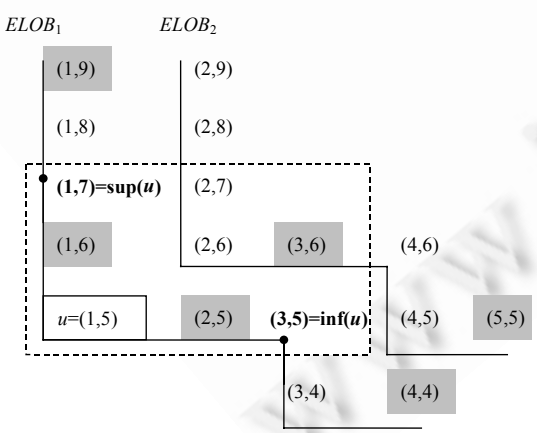

Fig.9 Rebuilt $ELOB_1'$ after deleting

图 9 删除后重构得到 $ELOB_1'$

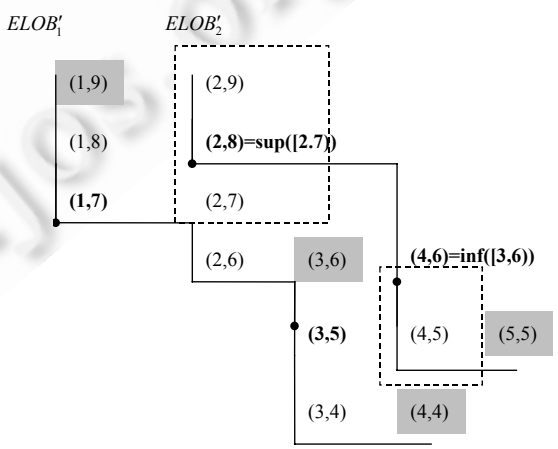

Fig.10 Rebuilt $ELOB_2'$ after cascade

图 10 级联操作后重构得到 $ELOB_2'$

2 基于 LOP 索引 TQOindex

2.1 辅助查询点集AQS

定义 8(MLOP 辅助查询点集合). 对于给定的 $EMLOP(\Gamma)$,设 $Q_0=[VTs,VTe]$为查询时间期间,$\forall ELOB\in EMLOP(\Gamma)$.按照基于时间期间拟序关系"$\preccurlyeq$",EMLOP 中各个 ELOB 包含 Q_0 的最小时间点定义为 Q_0 的辅助查询点.Q_0 关于 $EMLOP(\Gamma)$中所有辅助查询点构成的集合记为 $AQS(Q_0)$.

例 6:对于例 3 中的 $EMLOP(\Gamma)$来说,查询点 Q_0=[2,4]关于 $ELOB_1$ 的辅助点为填充点[2,5],关于 $ELOB_2$ 的辅助查询点为实点[2,6).如图 11 所示,$AQS([2,4))$={[2,5),[2,6)}.

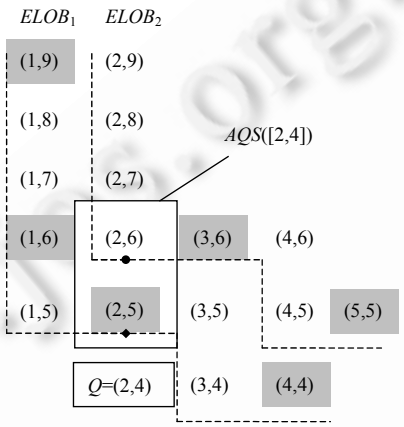

Fig.11 $AQS(Q)$ of query point Q=[2,4)

图 11 查询点 Q=[2,4)辅助点集 $AQS(Q)$

2.2 时态拟序索引树

定义 9(基于 LOB 时间数). 设 $u\in LOB,u=[VTs,VTe),LOB\in MLOP$,其序号为 $no(LOB).u$ 基于 LOB 的时间数(time member)定义如下,其中,r 是 MLOP 中所有时间期间中最大时间端点数的位数:

$$Tm(u,LOB)=no(LOB)\times 10^{2r}+Ve\times 10^{r}-Vs.$$

定理 3(时间数基本性质):

① 设 $u,v\in LOB,u\neq v$ iff $TN(u,LOB)\neq TN(v,LOB)$;

② 设 $u,v\in LOB,u\preccurlyeq v$ iff $TN(u,LOB)\leqslant TN(v,LOB)$.

证明:

① 设 $u=[VTs(u),VTe(u)),v=[VTs(v),VTe(v)).u\neq v\Leftrightarrow VTs(u)-VTs(v)\neq 0\vee VTe(u)-VTe(v)\neq 0$,

而 $TN(u,LOB)-TN(v,LOB)=(VTe(u)-VTe(v))\times 10^{r}-(VTs(u)-VTs(v))$,由此证得所需结论.

② 设 $u\preccurlyeq v$,即 $u\subseteq v\Leftrightarrow VTs(v)\leqslant VTs(u)\wedge VTe(u)\leqslant VTe(v)\Leftrightarrow VTs(u)-VTs(v)\geqslant 0\wedge VTe(u)-VTe(v)\leqslant 0$,

但 $TN(u,LOB)-TN(v,LOB)=(VTs(u)-VTs(v))\times 10^{r}-(VTs(u)-VTs(v))$,由此证得所需结论. □

由定理 3 中①可知:对于给定 LOB,其中 u 对应的 $TN(u,LOB)$是唯一确定的;由②可知:对于给定 LOB,其中元素之间的拟序关系可以通过相对应的时间数描述.

定义 10(基于时态拟序索引,TQOindex). 时间期间集合Γ的时态拟序索引(temporal quasi-order index):

$$TQOindex(\Gamma)=\langle EMLOP(\Gamma),MLOPB^{+}\text{-}tree(\Gamma)\rangle,$$

其中,MLOPB^{+}-tree 是存储偏序集Γ的 B^{+}-tree 结构,其索引对象为Γ中元素 u 对应的时间数 $Tm(u,LOB)$.

例 7:对于例 3 中的 EMLOP,相应的 $TQOindex(\Gamma)$如图 12 所示.

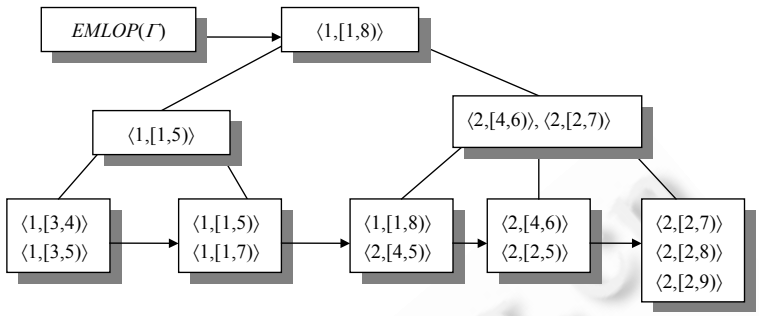

Fig.12 *TQOindex*(*Γ*)

图 12 *TQOindex*(*Γ*)

2.3 数据操作

在进行数据查询时,将 *EMLOP*(*Γ*)调入内存.设需查询 Q_0=[*Qs*,*Qe*).由 *EMLOP*(*Γ*)得到 $AQS(Q_0)$={(*no*,*Vs*,*Ve*)}. $\forall P_0 \in AQS(Q_0)$,将 P_0 作为查询目标进入 *MLOPB*$^+$*-tree*(*Γ*).按照常规 B+树操作进行查询,在叶节点中查找到大于等于 $Tm(P_0)$的最小数 $Tm(v_0)$,则在同一 LOB 中的所有大于等于 $Tm(v_0)$的 $Tm(v)$都是查询结果.

算法 4. 基于 TQOindex 查询.

输入:查询要求 Q_0=[$Vs(Q_0)$,$Ve(Q_0)$].

输出:基于 Q_0=[$Vs(Q_0)$,$Ve(Q_0)$)时间区间结果集 *Rs*,*Rs*=∅.

Step 1. Q_0 转换为 $Tm(Q_0)$,在 *EMLOP*(Q_0)中查找 $AQS(Q_0)$:$AQS(Q_0)$=∅,转 Step 4;否则,转 Step 2.

Step 2. 对于每个 $P_0 \in AQS(Q_0)$,多线程进入 *MLOPB*$^+$*-tree*(*Γ*)进入查询,得到相应的{*Tm*}.

Step 3. 将执行 Step 2 得到的{*Tm*}转化为时间期间,放入 *Rs*.

Step 4. 返回 *Rs*.

在 *EMLOP*(*Γ*)中查找 $AQS(Q_0)$的时间复杂度为|*EMLOP*(*Γ*)|×max{|*ELOB*|},算法 4 的时间复杂度为

$$|EMLOP(\Gamma)|\times\max\{|ELOB|\}+\log\lceil n/2\rceil(K),$$

其中,$\log\lceil n/2\rceil(K)$为 B+-tree 查询时间复杂度,*n* 为每个节点能够存储的最大关键字数目,*K* 为总的关键字数目.

基于 *TQOindex*(*Γ*)的时态查询的基本流程如图 13 所示.

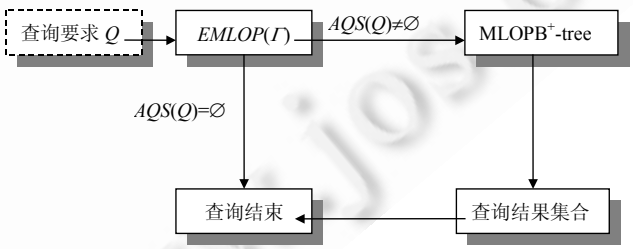

Fig.13 Query based on *TQOindex*(*Γ*)

图 13 基于 *TQOindex*(*Γ*)查询

基于 TQOindex 的时态查询具有如下特点:

① 若 $AQS(Q_0)$=∅,则 Q_0 在*Γ*中不存在查询结果,查询结束,无需调用 *MLOPB*$^+$*-tree*(*Γ*).此时,查询只在内存进行,可提升查询效率;若 $AQS(Q_0)$≠∅,则进入 *MLOPB*$^+$*-tree*(*Γ*).按照 B$^+$-tree 对 $AQS(Q_0)$中的元素依次查询.当满足条件的 LOB 片段分布于多个叶节点时,由相应指针找到后继点,由 LOB 序号得到所需结果.

② 由 LOB 性质,只要在 *MLOPB*$^+$*-tree*(*Γ*)的某叶节点中 LOB_0 内查找到一个结果 $v_0 \in LOB_0$,则 LOB_0 内 v_0 之前片段中的所有元素都是查询结果,统计得到查询结果集合,即"一次一集合".

③ MLOP 是Γ的线序划分,其中,LOB 相互并无关联,当$|AQS(Q_0)|>1$时,在 $MLOPB^+$-*tree*(Γ)中可对不同的辅助查询点进行多线程并发处理,$|AQS(Q_0)|$越大,多线程效率就越高.

例 8:设 Q_0=[2,4],由图 5 中的 *EMLOP*(Γ)得 $AQS(Q_0)$={(1,(2,5)),(2,(2,6))}.按照图 12 的 *TQOindex*(Γ),由辅助点(1,(2,5))得到⟨1,(1,5)⟩,⟨1,(1,8)⟩,⟨1,(1,8)⟩;由辅助点(1,(2,6))得到⟨2,(2,7)⟩,⟨2,(2,8)⟩,⟨2,(2,9)⟩.最终的查询集合为

{⟨1,(1,5)⟩,⟨1,(1,8)⟩,⟨1,(1,8)⟩;⟨2,(2,7)⟩,⟨2,(2,8)⟩,⟨2,(2,9)⟩}.

3 数据仿真与评估

3.1 数据查询

本文选取 Map21-tree[23]进行比较评估.实验数据涉及的参数设定为:随机生成包含在[0,*maxTime*]内的时间期间和相应集合Γ,其中,*maxTime* 为所生成时间期间最大的时间端点.磁盘物理块大小为 1 024 KB.每次查询由 50 条操作语句组成,相应的 I/O 开销为这 50 次执行操作产生的平均值.

3.1.1 基于数据量变化

取 *maxTime*=2000,最大时间期间跨度为 *maxTime*×10%,随机产生的时间区间数据量分别为 1×10^5,2×10^5,3×10^5,4×10^5,5×10^5,6×10^5,7×10^5,8×10^5,9×10^5,1×10^6.在图 14 中,横轴表示数据量(时间期间个数),纵轴表示访问磁盘块的 I/O 次数.由图 14 可知:对于相同查询跨度,随着数据量的增加,索引节点数相应地增加,需要访问的节点数也越来越多,TQOindex 和 Map21-tree 查询 I/O 次数均呈现上升趋势.但与 Map21-tree 相比,TQOindex 所需 I/O 次数上升趋势更为缓慢,性能更优.

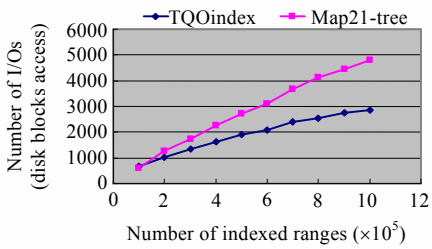

Fig.14 Changing of I/O based on increasing amounts of data

图 14 基于数据量增加时 I/O 的变化

3.1.2 基于磁盘块大小变化

取 *maxTime*=2000,查询跨度为 *maxTime*×10%,随机生成 5×10^5 个时间区间,磁盘物理块(*blockSize*)分别取值为 2^9B,2^{10}B,2^{11}B,2^{12}B.在图 15 中,横轴表示 *blockSize*,纵轴表示查询所需 I/O 次数.随着 *blockSize* 的增大,TQOindex 和 Map21-tree 查询所需 I/O 次数都呈现出下降趋势.这是由于对同样数量的时间期间,*blockSize* 值越大,索引节点数越少,所需访问的节点数也越少.但此时,TQOindex 比 Map21-tree 性能更优.

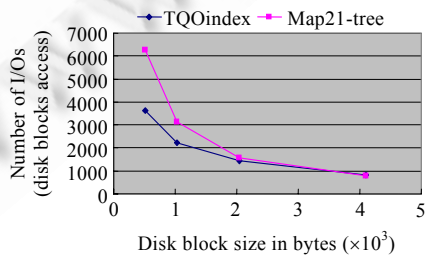

Fig.15 Changing based on different disk block's capacity

图 15 基于不同磁盘块大小变化

3.1.3 基于查询期间跨度变化

取 *maxTime*=2000,随机生成 5×10^5 个时间期间构成时间期间集合Γ,查询期间的跨度分别为 *maxTime* 的 1%,5%,10%,15%,20%,25%,30%,35%,40%,45%,50%.在图 16 中,横轴表示查询跨度,纵轴表示 I/O 次数.由图 16 可知:当查询起降跨度逐渐增大时,索引数据减少,TQOindex 和 Map21-tree 查询的 I/O 操作次数也减少;但此时,前者性能优于后者.

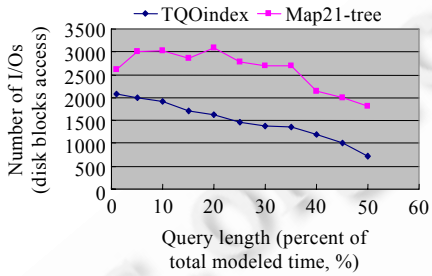

Fig.16 Changing based on different query span

图 16 基于不同查询跨度变化

3.2 数据更新

3.2.1 数据插入

取 *maxTime*=2000,时间期间集合Γ由随机生成的 5×10^5 个时间区间构成.通过算法 1 在Γ中生成 1 277 个 LOB 而构成 *MLOP*(Γ).随机生成 5×10^3 个时间区间加入到Γ后,将导致 *MLOP*(Γ)中部分 LOB 需要重组,即,产生增量式更新.此时,仿真的基本思想是:考察在 5 000 个新插入的时间期间中,导致 1 个 LOB 重建的有多少个,导致 2~5 个 LOB 重建的有多少个,等等,实验结果如表 1 和图 17 所示.

Table 1 Changing of the rebuilt LOB number after inserting

表 1 插入后需重构 LOB 数目变化

需要重建的 LOB 数	1	2~5	6~10	11~25	25 以上
5 000 个时间区间中引起需重建的 LOB 时间区间数	3 623	698	220	220	239
在 5 000 个时间区间中所占比率(%)	72.46	13.96	4.40	4.40	4.78

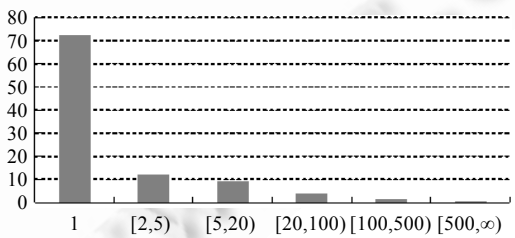

Fig.17 Percentage of the rebuilt LOB number in total inserted data

图 17 重构 LOB 数在插入总数中所占比率变化

3.2.2 数据删除

实验所用数据:*maxTime*=2000,随机生成 500 000 个时间区间,共有 1 277 个线序分枝.从随机生成的 500 000 个时间区间中选取 5 000 个时间区间进行删除测试.测试结果表明:删除 5 000 个时间区间所影响的时间区间总数为 5 014,其中,删除 1 个时间区间所影响的平均分枝数为 1.003 条左右.实验结果如图 18 所示.

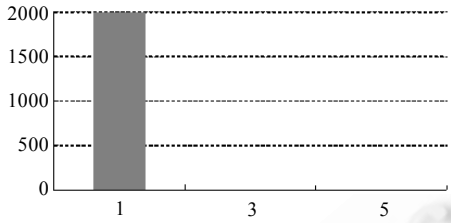

Fig.18 Average number of rebuild LOB after deleting

图 18 删除后需重构 LOB 的平均数目

3.3 时态关系数据查询

作为一种时态数据索引方法,前述仿真表明:TQOindex 能够有效地实现对外存数据信息查询,可在数据库平台上使用.本文随机产生较大规模的有效时间期间(*VTs*,*VTe*)集合,建立关系模式 *TR*(主键、有效时间始点 *VTs*、有效时间终点 *VTe*),在 SQL Server 2008 使用如下常规 SQL 语句进行有效时间 Q_0 查询:SELECT *VTs*,*VTe* FROM *TR* WHERE *VTs*<=*VTs*(Q_0) and *VTe*(Q_0)<=*VTe* and *VTs*<=*VTe*;同时,在 MAP21 和 TQOindex 进行 Q_0 的相应索引查询,仿真结果如图 19 所示.当数据量较小时,SQL 查询和 MAP21 以及 TQOindex 效果大体相当;但当数据量加大时,TQOindex 优于 SQL 和 MAP21.

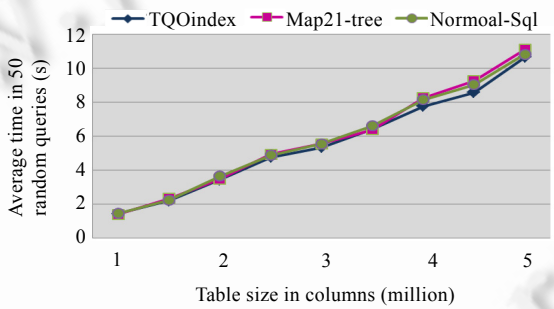

Fig.19 Query test based on temporal relation

图 19 基于时态关系的查询测试

关于时间信息的处理可分为基于代数和基于关系两种情形.关系数据操作建立在选择、投影和连接等代数运算上,数据处理本质上是将时间数据归结为常规关系数据,通过加有时间约束的代数运算进行操作;TQOindex 则是基于 Allen 提出的时间关系组织时态数据,通过拟序关系进行相应的操作.因此,上述仿真从一个侧面表现了基于关系的查询优于基于代数的查询.同时,由于 TQOindex 带有更为精细的结构,可用于如 XML、面向对象数据等非传统数据,而 Map21 和常规 SQL 查询则不易进行拓展,而且即使对于传统关系数据,其时态查询效率也优于后两者,因此,本文工作具有较好的适应性与可用性.

4 结 语

本文研究一种不同于常规代数方式的序关系时态拟序数据结构.首先,根据时态数据中时间信息和非时态数据内在联系,提出时态拟序关系概念,建立线序分枝“下优先算法”,得到基于拟序关系的时态数据结构,并证明结构在一定意义下的最优性.在拟序结构框架内,研究了时态数据查询算法和数据插入及删除的增量式算法.在本文中,时态数据结构具有数学支撑,能够实现“一次一集合”的查询模式和多线程的优化机制,适合于内存与外存两种方式,同时具有与相应的非时态数据结构(例如 XML 中的结构信息和移动对象中轨迹线信息等)协同的内禀特征,能够适用于多种新型时态数据(网络、移动对象和语义数据等)管理需要,具有良好的扩展空间.本文还讨论了拟序数据结构在外存数据索引上的应用,建立时态索引 TQOindex.通过较为精细的分析,在常规数据库

系统支持的 B+树平台上实现了时态数据索引.本文还对 TQOindex 进行仿真评估,通过与已有工作的比较,表明了本文工作的可行性与有效性.本文工作来自研究时态数据中时间与非时间信息整合处理的驱动,主要讨论其在时态关系数据管理方面的应用.实际上,时态拟序结构能够应用于对象、XML 和移动对象数据索引,相关工作将另文讨论.

References:

[1] Allen JF. Maintaining knowledge about temporal intervals. Communications of the ACM, 1983,26(11):832−843. [doi: 10.1145/182.358434]

[2] Bozkaya T, Ozsoyoglu M. Indexing transaction time databases. Information Sciences, 1998,112(1):85−123. [doi: 10.1016/S0020-0255(98)10024-5]

[3] Bliujute R, Jensen CS, Šaltenis S, Slivinskas G. R-Tree based indexing of now-relative bitemporal data. In: Proc. of the 24th VLDB Conf. New York: ACM Press, 1998. 345−356.

[4] Bliujute R, Jensen CS, Šaltenis S, Slivinskas G. Light-Weight indexing of bitemporal data. In: Proc. of the 12th Int'l Conf. on Scientific and Statistical Database Management. Berlin: IEEE Computer Society, 2000. 125−138. [doi: 10.1109/SSDM.2000.869783]

[5] Stantic B, Khanna S, Thornton J. An efficient method for indexing now-relative bitemporal data. In: Proc. of the 15th Conf. of Australasian Database. Australian Computer Society, Inc., 2004. 113−122.

[6] Moro MM, Tsotras VJ. Transaction-Time indexing. Technical Report, TR-90, New York: Springer-Verlag, 2008. [doi: 10.1007/978-0-387-39940-9_3 99]

[7] Lomet D, Hong MS, Nehme R, Zhang R. Transaction time indexing with version compression. Proc. of the VLDB Endowment, 2008,1(1):870−881. [doi: 10.14778/1453856.1453951]

[8] Tao Y, Papadias D. MV3R-Tree: A spatio-temporal access method for timestamp and interval queries. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB). San Francisco: Morgan Kaufmann Publishers Inc., 2001. 431−440.

[9] Pfoser D, Jensen CS, Theodoridis Y. Novel approaches to the indexing of moving object trajectories. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB). San Francisco: Morgan Kaufmann Publishers Inc., 2000. 395−406.

[10] Chakka VP, Everspaugh AC, Patel JM. Indexing large trajectory data sets with SETI. In: Proc. of the Conf. on Innovative Data Systems Research (CIDR). 2003. 164−175.

[11] Abdelguerfi M, Givaudan J, Shaw K, Ladner R. The 2-3 TR-tree, A trajectory-oriented index structure for fully evolving valid-time spatio-temporal datasets. In: Proc. of the 10th ACM Int'l Symp. on Advances in Geographic Information Systems. 2002. 29−34.

[12] Lee ML, Hsu W, Jensen CS, Cui B, Teo KL. Supporting frequent updates in R-trees: A bottom-up approach. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB). 2003. 608−619.

[13] Tao Y, Papadias D, Sun J. The TPR*-Tree: An optimized spatio-temporal access method for predictive queries. In: Proc. of the Int'l Conf. on Very Large Data Bases (VLDB). 2003. 790−801.

[14] Procopiuc CM, Agarwal PK, Har-Peled S. STAR-Tree: An efficient self-adjusting index for moving objects. In: Proc. of the Workshop on Algorithm Engineering and Experimentation. Berlin, Heidelberg: Springer-Verlag, 2002. 178−193. [doi: 10.1007/3-540-45643-0_14]

[15] Šaltenis S, Jensen CS. Indexing of moving objects for location-based services. In: Proc. of the Int'l Conf. on Data Engineering (ICDE). 2002. 463−472. [doi: 10.1109/ICDE.2002.994759]

[16] Mendelzon AO, Rizzolo F, Vaisman A. Indexing temporal XML documents. In: Proc. of the 30th VLDB Conf., Vol.30. VLDB Endowment, 2004. 216−227.

[17] Wang FS, Zaniolo C. Publishing and querying the histories of archived relational databases in XML. In: Proc. of the 4th Int'l Conf. on Web Information Systems Engineering (WISE 2003). IEEE, 2003. 93−102. [doi: 10.1109/WISE.2003.1254473]

[18] Rizzolo F, Vaisman AA. Temporal XML: Modeling, indexing, and query processing. The VLDB Journal, 2008,17(5):1179−1212. [doi: 10.1007/s00778-007-0058-x]

[19] Ye XP, Chen KY, Tang Y. Technology on temporal XML indexing. Chinese Journal of Computers, 2007,30(7):1074−1085 (in Chinese with English abstract).

[20] Ye XP, Guo H, Tang Y, Chen LW, Zhou C, Liao QY. Index of mobile data based on phrase points analysis. Chinese Journal of Computers, 2011,34(2):256−274 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.00256]

[21] Ye XP, Tang Y, Chen LW, Guo H, Zhu J, Chen KY. Study and application of temporal index technology. Science in China Series F: Information Sciences, 2009,52(6):899−913. [doi: 10.1007/s11432-009-0115-8]

[22] Guo H, Ye XP, Tang Y, Chen LW. Temporal XML index based on temporal encoding and linear order partition. Ruan Jian Xue Bao/Journal of Software, 2012,23(8):2042−2057 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4161.htm [doi: 10.3724/SP.J.1001.2012.04161]

[23] Nascimento M, Dunham M. Indexing valid time database via B+-tree, The MAP21 approch. Technical Report, CSE-97-08, Dallas: School of Engineering and Applied Sciences, Southern Methodist University, 1997.

附中文参考文献:

[19] 叶小平,陈凯原,汤庸.时态 XML 索引技术.计算机学报,2007,30(7):1074−1085.

[20] 叶小平,郭欢,汤庸,陈罗武,周畅,廖青云.基于相点分析的移动数据索引技术.计算机学报,2011,34(2):256−274. [doi: 10.3724/SP.J.1016.2011.00256]

[22] 郭欢,叶小平,汤庸,陈罗武.基于时态编码和线序划分的时态 XML 索引.软件学报,2012,23(8):2042−2057. http://www.jos.org.cn/1000-9825/4161.htm [doi: 10.3724/SP.J.1001.2012.04161]

叶小平(1955－),男,陕西安康人,博士,教授,博士生导师,主要研究领域为时态数据库技术,NoSQL 数据库技术,计算机病毒.
E-mail: scnuyexp@163.com

汤庸(1964－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为时态数据库,协同软件,云服务与信息安全,移动互联网应用,学术信息检索与推荐系统,面向学者的社交网络.
E-mail: scnuyxp2013@163.com

林衍崇(1990－),男,硕士生,CCF 学生会员,主要研究领域为时态数据库,NoSQL 数据库技术.
E-mail: jimslin1990@163.com

陈钊滢(1990－),女,硕士生,CCF 学生会员,主要研究领域为移动对象数据库.
E-mail: 13429637@qq.com

张智博(1990－),男,硕士,CCF 学生会员,主要研究领域为时态与移动对象数据管理.
E-mail: paolo7@qq.com