一种基于线序划分的时态数据索引技术

叶小平 朱峰华 汤 庸 周 畅 廖青云
(华南师范大学计算机学院 广州 510651)

摘 要 讨论一种基于线序划分的时态数据索引技术。首先,讨论了时间期间集合上线序划分概念及其构建算法;其次,提出了基于常规磁盘方式进行数据管理的时态数据库索引技术TQOindex;另外,对TQOindex进行的基本实验仿真表明了其可行性和有效性。TQOindex的基本特征是基于"序关系"数学框架,能够实现"一次一集合"的数据操作。

关键词 时态数据库,线序划分,基于外存数据索引

中图法分类号 TP311.13 **文献标识码** A

Temporal Data Index Based on Linear Order Partition

YE Xiao-ping ZHU Feng-hua TANG Yong ZHOU Chang LIAO Qing-yun
(School of Computer Science, South China Normal University, Guangzhou 510651, China)

Abstract The paper presented the temporal data index technology based on linear order partition. Firstly, the paper discussed the conception of linear order partition as well as its constructing algorithm on a given period set. Secondly, it put forward the TQOindex which is capable of indexing temporal database in external storages. Besides, the simulation results of the index show its practicability and effectiveness. The basic property of TQOindex is that it is built on the math frame of order relationship and its data operation can be completed once a set in temporal query.

Keywords Temporal database, Linear order partition, Data index based on external storage

1 引言

世界上客观事物都在发展变化,发展变化需要通过时间进行描述,时态数据库就是对带有时间标签的数据进行有效管理的技术。数据查询是数据管理的基本要求,时间本身特有的单向性(单调递增)、多维性(有效、事务和用户时间维等)和相互关系的复杂性(ALLEN时间关系[1])等,使得时态数据难以直接纳入传统关系数据框架,相应索引就成为实现数据查询的基本途径。计算机数据管理通常有基于外存和基于内存两种方式。时态数据索引也可按这两种方式进行分类。据我们所掌握的资料,基于外存时态数据主要有基于B+树的有效时间索引MAP21[2]、事务时间索引[3]和版本管理[4];基于R树的双时态索引Light-weight index[5]、基于变量语义双时态索引[6]、时空索引MV3R-Tree[7]及TPR*-Tree[8]。基于内存主要是基于新型数据模型的时态数据索引,例如基于结构摘要[9]和基于时态拟序[10,11]的有效时间XML索引,基于B+树的事务时间XML索引[12]、基于R树[13-15]或相平面分析[16]的移动对象数据索引。就时态数据索引来说,基于内存方式适用于时间标签和相应数据模型的"强耦合"情形(如XML和移动对象数据);基于外存方式多适用于"弱耦合"情形(如时态关系)或将时间转化为空间(如时空数据)情形。研究基于"弱耦合"情形下的索引技术对于提高系统效率和扩大应用都是必要的、有价值的。本文研究一种"弱耦合"框架下基于外存的时态数据索引TQOindex,其具有明显的数学基础,在某种意义下可以看作一种时态索引框架;另外,对时态数据集合上等价类进行数据操作,每次查询均可返回一个集合的结果集,其具有"一次一集合"的特征;同时,通过时间标签和相应时间数之间一一对应,能够采用基于常规数据库的B+树索引方式,其具有较为广泛和有效的实际应用场景。

本文第2节是本文工作的原理基础,主要通过线序划分LOP建立了时间期间集合上的基本数据结构和相应构建算法;第3节是本文工作的技术实现,主要是通过在LOP框架内引入"时间期间数"来建立基于B+树的索引TQOindex;第4节是对TQOindex的大数据量仿真评估,表明了本文工作的可行性和有效性。

2 时间期间集合基本数据结构LOP

时态数据(Temporal data)是一个非时态数据与一个时间

到稿日期:2012-03-26 返修日期:2012-07-14 本文得到国家自然科学基金(60970044,61272067),国家科技支撑计划项目(SQ2011GX07E01500),广东省自然基金团队研究项目(S2012030006242),广东省重大科技专项计划项目(2012A080104019),广东省部产学研结合项目(2011A090100003),广东省战略新兴产业项目(2011A010801007,2011168005),广东省科技计划项目(2011A091000036,2011B080100031)和广东省自然科学基金(05003348,9151027501000054,s2011010003409)支持。

叶小平(1955-),男,博士,教授,主要研究方向为移动对象数据管理、网络数据库技术和计算智能;**朱峰华**(1987-),男,硕士生,主要研究方向为时态数据库技术,E-mail:zfh-cs@163.com;**汤 庸**(1964-),男,博士,教授,主要研究方向为时态数据与信息管理、计算机协同工作技术和信息服务;**周 畅**(1987-),男,硕士生,主要研究方向为移动对象数据库技术;**廖青云**(1987-),男,硕士生,主要研究方向为时态数据库技术。

标签构成的二元组 $Td=\langle D, Tstamp\rangle$，本文中时间标签 *Tstamp* 是有效时间期间(valid time period, VT)，并记为 $VT=[VTs, VTe)$，*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，*E* 上有关系 $\lesssim$：$Td_1, Td_2 \in E, Td_1 \lesssim Td_2 \Leftrightarrow VT(Td_1) \subseteq T(Td_2)$，则称"$\lesssim$"是 *E* 上一个时态拟序。设 Γ 是 *E* 中时间期间集合。$\forall u \in \Gamma$，$u=[VTs, VTe)$，记 *u* 在 *VTs-VTe* 平面上对应的点为 $P(u)=[VTs, VTe)$，这样是一个 1-1 对应。此时，称 $P(u)=[VTs, VTe)$ 为 *u* 对应的(二维)时间点(2-dimension time point)。即是说，*u* 对应 *VTs-VTe* 平面上的一个点 $P(u)$，Γ 对应 *VTs-VTe* 平面上的一个点集 $P(\Gamma)$。

设 $P_0=(\min\{VTs(P)\}, \max\{VTe(P)\}), P \in \Gamma$。由 P_0 开始得到的 $P(\Gamma)$ 深度优先遍历序列 $S(\Gamma)$ 称为 $P(\Gamma)$ 的"深度优先序列"。

定义 2(线序划分) 设 Γ 是具有如上定义拟序"$\lesssim$"的时间期间拟序集合。Γ 的一个全序分枝称为 Γ 的一个线序分枝(Linear Order Branch, *LOB* 或 *L*)。设 *LOP* 是 Γ 上 *LOB* 的集合，若 $\forall LOB_i, LOB_j \in LOP, i \neq j, LOB_i \cap LOB_j = \varnothing$，且 $\bigcup LOB=\Gamma$，则称 *LOP* 是 Γ 上一个线序划分(Linear Order Partition, *LOP*)，并记为 $LOP(\Gamma)$。

算法 1(LOP 的下优先算法) 设有深度优先序列 $S(\Gamma)$。

Step1 由 $S(\Gamma)$ 中首元素 u_0 开始至 $u_{i0} \in S(\Gamma)$，$VTs(u_{i0}) = VTs(u_0) \wedge (VTs(u_{i0+1}) \neq VTs(u_0))$。

Step2 由 u_{i0} 开始至 u_{i1}：$VTe(u_{i1}) = VTe(u_{i0}) \wedge VTe(u_{i1}) = \min\{VTs(u_j)\}$，其中，$u_j \in S(\Gamma) \wedge (\exists u_k \in S(\Gamma), VTs(u_k) = VTs(u_j) \wedge VTe(u_k) < VTe(u_j))$

Step3 由 u_{i1} 开始，继续"step1"和"step2"，直至 $u_m \in S(\Gamma)$，$\exists u_m' \in S(\Gamma)$ 使得 $(VTs(u_m') > VTs(u_m) \wedge VTe(u_m') < VTe(u_m))$，$S(\Gamma)$ 中由 u_0 至 u_m 的子序列即是一个 LOB_1。

Step4 从 $(\Gamma) \backslash LOB_1$ 中首元素开始，继续上述 Step1－Step3，计算可得 LOB_2，…，如此，即可得 $LOP(\Gamma)$。

设 $VTs(\Gamma)=\max\{VTs(u) \mid u \in \Gamma\}$，$VTe(\Gamma)=\max\{VTe(u) \mid u \in \Gamma\}$，则算法 1 时间复杂度为 $(VTs(\Gamma) \times VTe(\Gamma))/2$。

3 时态索引 TQOindex

时间期间集合 Γ 上的基本数据结构就是在其上建立 *LOP*。为了使用 B+树技术范畴，需要将时间期间之间某些拓扑关系(如基本的包含关系)转换为整数之间的代数关系(如"大小"关系)。一般而言，这种转换不能保证"一一对应"[2]，但本节工作表明，在 *LOP* 框架内能够实现"区间"与"整数"的一一对应。

为建立所需时态索引，还需要引入下述概念。

定义 3(扩展的线序分枝 *ELOB* 和扩展的线序划分 *ELOP*) 设 $LOB=\langle u_1 \cdots, u_i, u_{i+1}, \cdots, u_m\rangle$ 是按照算法 1 得到的线序分枝，u_i 和 u_{i+1} 是 *L* 中的时间期间，$P(u_i)$ 是 u_i 在 *VTs-VTe* 平面上对应的时间点。按照下述方法得到的折线段称为 *L* 在 *VTs-VTe* 平面上的 *LOB* 扩充(extended LOB)，并记为 *ELOB*：① 对于 *L* 中任意两个"相邻"的时间 u_i 和 u_{i+1}，当 $P(u_i)$ 和 $P(u_{i+1})$ *VTs* 坐标或 *VTe* 坐标相同时，就直接将这两点用线段相连接；② 否则，就在 $P(u_i)$ 和 $P(u_{i+1})$ 间加入新的点 $P(v)$，其中 $v=[VTs(u_{i+1}), VTe(u_i))$，此时用线段连接 u_i 与 v，v 与 u_{i+1}；③ 另外，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* 平面上对角线上的点 $P(v_0)$，其中 $v_0=[VTs(u_m), VTe(u_m))$。

EL 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 $u(r)$ 和 $u(f)$。

给定 $LOP(\Gamma)$ 中所有 *LOB* 对应的 *ELOB* 构成的集合称为 *LOP* 的扩充集(extended LOP)，并记为 $ELOP(\Gamma)$。$ELOP(\Gamma)$ 实际上可与 $S(\Gamma)$ 同时构造。$ELOP(\Gamma)$ 用二维数组 *ELOP*[*VTs*][*VTe*] 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 $u=[VTs, VTe)$ 所在的 *ELOB* 序号，*flag* 用于标识 *u* 是否属于 *LOP*，当 $u \in LOP$，$flag=t$(true)；当 $u \in ELOP \backslash LOP$，$flag=f$(false)。

定义 4(基于 *LOB* 时间数) 设 $u \in LOB$，$u=[VTs, VTe)$，$LOB \in ELOP$，其序号为 $no(LOB)$。时间期间 *u* 基于 *LOB* 的时间数(time member)定义如下：

$$TM(u, LOB)=no(LOB) \times 10^{2r} + VTe \times 10^{r} - VTs$$

其中，*r* 是 Γ 中所有时间期间中"最大"时间端点数的"位数"。

由 *LOB* 概念和上述定义可以证明下述定理。

定理 1(时间数基本性质)

① 在同一 *LOB* 内，时间期间 *u* 和 $TM(u, LOB)$ 之间的对应是一个 1-1 映射。

② $\forall u, v \in LOB_0$，$u \prec v \Leftrightarrow TM(u, LOB) \leqslant TM(v, LOB)$，其中 LOB_0 是给定的线序分枝。

定义 5(*ELOP* 辅助查询点集合) 对于给定 $ELOP(\Gamma)$，设 $Q_0=[VTs, VTe)$ 为查询时间期间，$\forall ELOB \in ELOP(\Gamma)$。按照拟序关系"$\prec$"，*ELOP* 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 $ELOP(\Gamma)$ 中所有 *ELOB* 的所有辅助查询点构成的集合称为 Q_0 的辅助查询点集合(auxiliary query set)，记为 $AQS(Q_0)$。

定义 6(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) $TQOindex(\Gamma)=\langle ELOP(\Gamma), LOPB^+\text{-tree}(\Gamma)\rangle$。

① $ELOP(\Gamma)$ 是定义 4 中给出的二维数组。

② $LOPB^+$-tree 是存储偏序集 $TM(LOP)$ 的 B^+-tree 结构，其索引对象为中元素 *u* 对应的时间数 $TM(u, LOB)$。

进行数据查询时，将 $ELOP(\Gamma)$ 调入内存。设需查询时间期间 $Q_0=[VTs, VTe)$。由二维数组 $EMLOP(\Gamma)$ 顺序存储得到 $AQS(Q_0)=\{(no, VTs, VTe)\}$。$\forall P_0 \in ASQ(Q_0)$，将 P_0 作为查询目标进入 $LOPB^+$-tree(Γ)。按照常规 B+树操作进行查询，在叶结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。基于 $TQOindex(\Gamma)$ 时态查询的基本流程如图 1 所示。

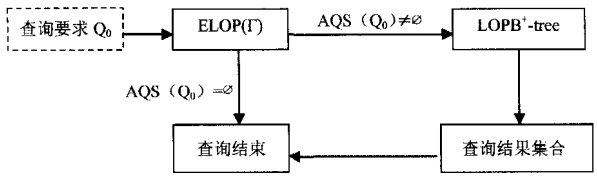

图 1 基于 $TQOindex(\Gamma)$ 查询

基于 TQOindex 的时态查询具有如下特点：

① 若 $AQS(Q_0)=\varnothing$，则 Q_0 在 Γ 中不存在查询结果，查询结束，无需调用 $LOPB^+$-tree(Γ)。此时查询只在内存进行，不进入磁盘，因此提升查询效率。若 $AQSQ_0)\neq\varnothing$，则进入 $LOPB^+$-tree(Γ)。按常规 B^+-tree 对 AQS (Q_0)元素进行依次查询。当满足条件 LOB 片段分布于多个叶结点时，由叶结点指针找到后继叶结点，再由 LOB 序号得到所需结果。

② 由 LOB 性质，只要在 $LOPB^+$-tree(Γ)某叶结点中 LOB_0 内查找到一个结果 $v_0\in LOB_0$，则 LOB_0 内 v_0“之后”片段中所有都是查询结果，即得查询结果集合，即“一次一集合”。

③ LOP 是 Γ 的线序划分，其中 LOB 相互并无关联，当 $|AQS(Q_0)|>1$ 时，在 $LOPB^+$-tree(Γ)中可对不同辅助查询点进行多线程并发处理，$|AQS(Q_0)|$愈大，多线程效率愈高。

例 1　设深度优先序列实例 $S(\Gamma)$如图 2 所示。

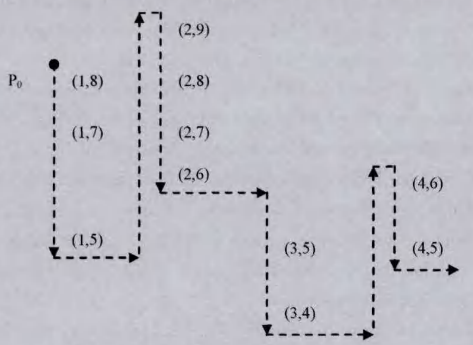

图 2　深度优先序列 $S(\Gamma)$

此时，对于 $S(\Gamma)=\langle[1,8),[1,7),[1,5),[3,5),[3,4),[2,9),[2,8),[2,7),[2,6),[4,6),[4,5)\rangle$，算法 1 的实现如图 3 所示，由此得到 LOB：$LOB_1=\langle[1,8),[1,7),[1,5),[3,5),[3,4)\rangle$，$LOB_2=\langle[2,9),[2,8),[2,7),[2,6),[4,6),[4,5)\rangle$。

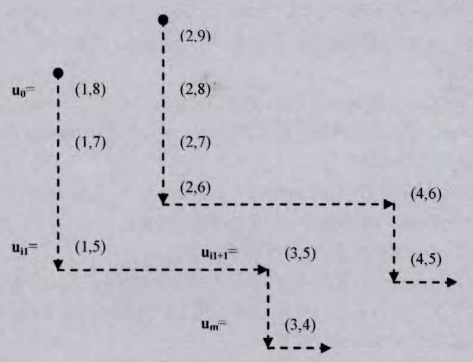

图 3　下优先算法实现

对于 $LOP=\langle LOB_1,LOB_2\rangle$，相应 $ELOP=\langle ELOB_1,ELOB_2\rangle$：

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

$ELOB_2=\langle[2,9),[2,8),[2,7),[2,6),[3,6),[4,6),[4,5),[5,5)\rangle$

$ELOB_1$ 和 $ELOB_2$ 如图 4 所示，其中不带“阴影”结点和带阴影结点分别对应 $u(r)$和 $u(f)$。

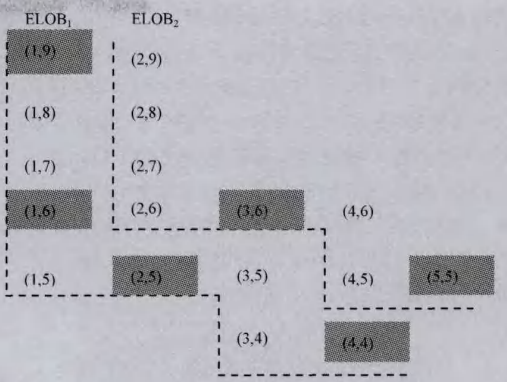

图 4　$ELOP(\Gamma)$

对于 $ELOP(\Gamma)$，查询点 $Q_0=[2,4)$ 关于 $ELOB_1$ 和 $ELOB_2$ 的辅助点分别为[2,5)和[2,6)，如图 5 所示。$AQS([2,4))=\{[2,5),[2,6)\}$，[2,5)为填充点，[2,6)为实点。$ELOP(\Gamma)$相应的 $TQOindex(\Gamma)$如图 6 所示。

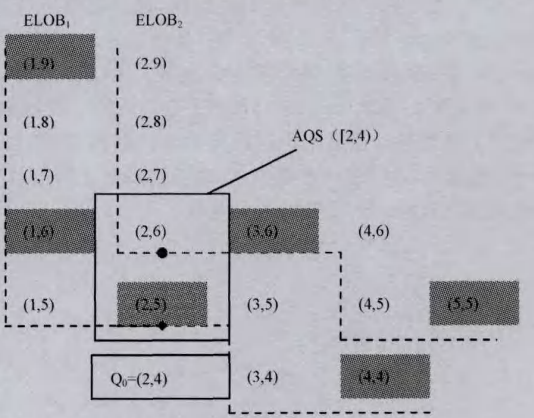

图 5　查询点 $Q_0=[2,4)$辅助点集 $AQS(Q_0)$

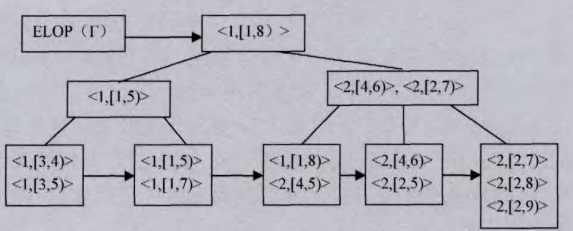

图 6　$TQOindex(\Gamma)$

设 $Q_0=[2,4)$，由 $ELOP(\Gamma)$可得 $AQS(Q_0)=\{(1,(2,5)),(2,(2,6))\}$。按照 TQOindex($\Gamma$)，由 (1,(2,5))得查询结果$\langle1,(1,5)\rangle$，$\langle1,(1,7)\rangle$，$\langle1,(1,8)\rangle$；由 (1,(2,6))得查询结果$\langle2,(2,6)\rangle$，$\langle2,(2,7)\rangle$，$\langle2,(2,8)\rangle$，$\langle2,(2,9)\rangle$。最终查询结果集合为$\{\langle1,(1,5)\rangle,\langle1,(1,7)\rangle,\langle1,(1,8)\rangle;\langle2,(2,6)\rangle,\langle2,(2,7)\rangle,\langle2,(2,8)\rangle,\langle2,(2,9)\rangle\}$。

4　实验仿真

本文选取 MAP21[2] 进行比较评估，主要是由于 MAP21 实现了时间期间和时间数的单项对应，这是使用 B+树的基本前提，而这与本文工作具有技术可比性；其次，MAP21 是使用 B+树实现有效时间索引的经典工作，后续工作多有借鉴和扩充，比较具有一般意义。仿真过程随机生成包含在 [0, maxTime)内的时间期间和相应集合 Γ，maxTime 为所生成时间期间中最大时间终点。磁盘块大小为 1024kB。每次测试

查询由 50 条语句组成，I/O 开销为 50 次操作的平均值。取 maxTime=2000，最大时间跨度为 maxTime * 10%，随机数据量分别为 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 7 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 7 可知，对于相同查询跨度，随着数据量增加，索引结点数相应增加，需访问结点数增多，TQOindex 和 MAP21 查询 I/O 次数均呈现上升趋势。但与 MAP21 相比，TQOindex 所需 I/O 次数上升趋势更为缓慢，性能更优。

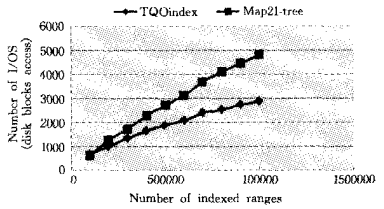

图 7　基于数据量增加时 I/O 的变化

另外，随机生成 5×10^5 个时间区间，磁盘物理块（Block Size）分别取为 2^9 B，2^{10} B，2^{11} B，2^{12} B。如图 8 所示，横轴表示 Block Size，纵轴表示查询所需 I/O 次数。由图 8 可知，随 Block Size 增大，TQOindex 和 MAP21 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，Block Size 取值越大，索引结点数越少，查询需访问结点数也越少，但此时 TQOindex 较 MAP21 性能更优。

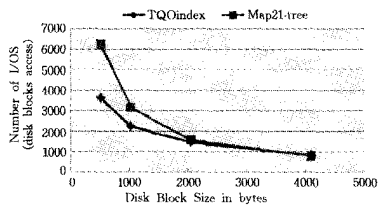

图 8　基于不同磁盘块大小变化

TQOindex 是一种基于外存文件中的时态数据索引技术，因此能够在常规数据库平台上使用。本文使用 MySQL 数据库平台，随机产生基于有效时间期间[*VTs*，*VTe*）的时态关系数据文件，其中将 *VTs* 和 *VTe* 分别作为常规属性处理，通过直接使用 SQL 查询语句、基于 MAP21 的时态数据索引和 TQOindex 对满足相应时间条件的元组进行筛选，相应结果如图 9 所示。

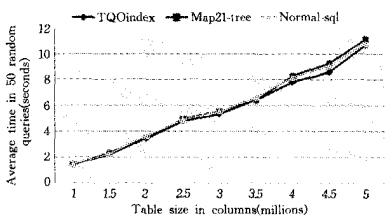

图 9　基于时态关系数据的时态查询测试

TQOindex 带有更为复杂的结构，能够用于各种非经典数据（时态对象、时态 XML 和移动对象）情形。而常规 SQL 语句和 MAP21 查询则难以进行相应拓展，但对于传统的关系数据来说，其时态查询效率也优于后两者，这说明本文工作具有多环境的适应性与可用性。

结束语　本文主要工作是研究一种新的基于外存的时态数据索引技术 TQOindex。首先，与传统时态查询基于“代数”的情形不同，TQOindex 是基于“（序）关系”的；其次，TQOindex 虽然具有比较精细的数据结构，但能够和传统查询一样具有“一次一集合”查询结果和多线程管理模式。由于 TQOindex 具有基本数学支撑，从某种角度来看，它可以作为一种时间数据处理框架，能够用于较为广泛的应用领域，例如时态对象数据、时态 XML 数据和移动对象数据等。另外，TQOindex 还可以作为一种动态索引，实现时间数据的增量式更新（数据插入和数据删除），从而用于各种海量数据管理与应用，由于篇幅限制，TQOindex 更新机制将另文讨论或参见文献[10,11]。

参 考 文 献

[1] Allen J F. Maintaining knowledge about temporal intervals[J]. Communications of the ACM, 1983, 26(11): 832-843

[2] Nascimento M, Dunham M. Indexing Valid Time Database via B+-Tree, The MAP21 Approch [R]. CSE-97-08. Dallas, USA: School of Engineering and Applied Sciences, Southern Methodist University, 1997

[3] Bliujute R, Jensen C S, Saltenis S, et al. Light-weight indexing of bitemporal data[C]// Proceedings of the 12th International Conference on Scientific and Statistical Database Management. Berlin: IEEE Computer Society, 2000: 125-138

[4] Stantic B, Khanna S, Thornton J. An efficient method for indexing now-relative bitemporal data[C]// Proceedings of the fifteenth conference of Australasian database, ACM International Conference Proceeding Series. New Zealand: Dunedin, 2004: 113-122

[5] Moro M M, Tsotras V J, Transaction-time indexing [R]. volume Time Center Technical Report TR-90. Temporal Database Entries for the Springer Encyclopedia of Database Systems, US, New York: Springer 2008

[6] Lomet D, Hong M, Nehme R, et al. Transaction time indexing with version compression [J]. Proceedings of the VLDB Endowment, 2008, 1(1): 870-881

[7] Tao Y, Papadias D. MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries[C]// Proc. of the Intl. Conf. on Very Large Data Bases, VLDB. Sept. 2001: 431-440

[8] Tao Y, Papadias D, Sun J. The TPR*-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries[C]// Proc. of the Intl. Conf. on Very Large Data Bases, VLDB. Sept. 2003

[9] Rizzolo F, Vaisman A. Temporal XML: modeling, indexing, and query processing[J]. The VLDB Journal, 2008, 17(5): 1179-1212

[10] Ye Xiao-ping, Chen Kai-yuan, Tang Yong. Technology on temporal XML indexing[J]. Chinese journal of computer, 2007, 30(7): 1074-1085

[11] Guo Huan, Ye Xiao-ping, Tang Yong, et al. Temporal XML Index Based on Temporal Encoding and Linear Order Partition [J]. Journal of Software, 2012, 23(8)

[12] Mendelzon A, Vaisman A. Indexing temporal XML documents [C]// Proceedings of the 30rd VLDB Conference. Toronto, Canada, ACM, 2004: 216-227

[13] Pfoser D, Jensen C S, Theodoridis Y. Novel Approaches in Query Processing for Moving Object Trajectories[C]// Proc. of the Intl. Conf. on Very Large Data Bases, VLDB. Sept. 2000: 395-406

[14] Saltenis S, Jensen C S. Indexing of Moving Objects for Location-Based Services[C]// Proc. of the Intl. Conf. on Data Engineering, ICDE. Feb. 2002

[15] Chakka V P, Everspaugh A, Patel J M. Indexing Large Trajectory Data Sets with SETI[C]// Proc. of the Conf. on Innovative Data Systems Research, CIDR. Asilomar, CA, Jan. 2003

[16] Ye Xiao-ping, Guo Huan, Tang Yong, et al. Index of Mobile Data Based on Phrase Points Analysis[J]. Chinese journal of computer, 2011, 34(2): 256-274