一种基于时态中间件的高效双时态索引模型*)

康向锋 汤 庸 叶小平 汤 娜
(中山大学计算机科学系 广州 510275)

摘 要 当前的时态数据库中间件不支持包含事务时间和有效时间的双时态数据索引,通过使用适当的数据变换和查询变换,可将双时态数据转化为R树可索引的数据。基于4R技术,提出了作为时态中间件TimeDB组件的双时态索引模型B4Rindex。实验证明,利用该模型对双时态数据进行索引是高效的。

关键词 时态数据库,双时态索引,TimeDB中间件,R树

An Efficient Bitemporal Index Model Based on the Temporal Middleware

KANG Xiang-Feng TANG Yong YE Xiao-Ping TANG Na
(Department of Computer Science, Zhongshan University, Guangzhou 510275)

Abstract The temporal middleware does not support the index of bitemporal data, which include the transaction time and valid time. By using the proper data transformation and query transformation, the bitemporal data can be transferred to the data of 4R index. Based on the 4R index technique, a B4Rindex model was proposed, which can be made as a component of the temporal middleware——TimeDB. The experiment proves that this model is efficient to index the bitemporal data.

Keywords Temporal database, Bitemporal index, TimeDB middleware, R-tree

1 引言

时间是自然界无所不在的客观属性,时态数据库一直伴随着数据库技术的发展而产生和发展。由于当前的主流商业DBMS缺乏对时态数据的支持,文[1]引入了一种嵌入式时态应用模式(Embedding Temporal Application Mode),它通过一些时态处理开发工具(时态中间件)来实现应用系统中的时态信息处理功能,底层数据库系统仍采用传统的RDBMS,从而提供了一种基于传统RDBMS实现时态数据库的方式。当前的时态中间件模型均没有对时态数据进行索引优化,仍然依赖于传统的数据库索引技术(例如B^+树)。文[2]比较了多种常规的索引技术,它表明B^+树不适合索引时态数据。

文[3,4]提出借鉴广泛应用于空间索引技术的R树来索引时态数据,但在解决带有效时间变元NOW和事务时间变元UC的双时态数据问题上,所采用的最大时间戳R树存在死空间(Dead Space)过大、重叠(Overlap)过多等缺陷,从而直接导致查询、更新操作的效率低下。文[5]提出的GR树扩充了R树的功能,使其既可以处理带变元NOW和UC的双时态数据,又可以降低死空间和重叠等问题,是一种较好的双时态索引技术,但目前尚未有DBMS支持该索引技术。如果要将GR树索引技术应用到主流数据库系统Oracle、DB2或Sybase中,必须扩充DBMS的内核。文[6]提出的4R树,其索引效果可与GR树相当,可直接应用于支持R树索引的DBMS上层,如Informix、PostgreSQL等数据库系统。该方法的实现比GR树简单且以较成熟的R树为基础,所以4R树是一种较优的双时态索引技术。

本文基于较优的4R树双时态索引技术,提出基于嵌入式时态中间件TimeDB的双时态索引B4Rindex(Bitemporal 4R index,双时态4R索引)模型。通过实验使用大量的数据集模拟真实的数据操作,统计查询、更新操作所引起的磁盘I/O次数来衡量该模型的效能。实验证明,基于B4Rindex的索引模型大大提高了双时态数据查询与更新速度(平均提高因子为3~5),有效地解决了当前时态数据库中间件索引双时态数据的性能瓶颈。

2 4R索引

2.1 双时态数据类型

双时态数据由两个时间维组成:有效时间(Valid Time)和事务时间(Transaction Time)[7],其中有效时间是对象在现实世界中为真(存在)的时间,事务时间指一个事实存储在数据库中的时间。分别用TT^+,TT^-,VT^+,VT^-代表事务时间的起始、终止和有效时间的起始、终止。当有效时间的终止值不确定时,用NOW表示有效时间的终止;事务时间终止值不确定时,用UC表示。其中,NOW和UC均为与当前时间(Current Time, CT)相关的时间变元。双时态数据可分为四种类型,如表1所示。

表1 四种双时态数据类型

类型	TT^+	TT^-	VT^+	VT^-
A	tt_1	UC	vt_1	NOW
B	tt_1	UC	vt_1	vt_2
C	tt_1	tt_2	vt_1	NOW
D	tt_1	tt_2	vt_1	vt_2

双时态数据采用Snodgrass提出的TQuel 4TS表示数据

*)基金项目:国家自然科学基金项目(60373081);广东省自然科学基金重点项目(04105503)。康向锋 硕士研究生,主要研究方向为时态数据库、数据索引技术。汤 庸 博士,教授,博士生导师,主要研究方向为数据库与知识库、CSCW等。叶小平 博士,副教授,硕士生导师,主要研究方向为数据库与知识库、Rough集理论等。汤 娜 博士研究生,主要研究方向为数据库与知识库、CSCW等。

模型(Representational Data Model)[8]，即双时态数据可表示为$[TT^{\vdash},TT^{\dashv};VT^{\vdash},VT^{\dashv}]$。

2.2 数据变换

如何对四种类型的双时态数据建立索引是问题的关键。4R索引的核心思想是：先消除双时态数据中的变元，再利用R树建立索引。具体可分为以下三步：(1)将双时态数据分为四类如表1；(2)对每一类数据执行数据变换，消除变元；(3)将变换后的四类数据分别建立R树索引。

数据变换的目的在于消除双时态数据中的时态变元NOW和UC，从而可以利用R树建立索引。在定义数据变换之前，分别给出带变元、无变元的双时态数据域的定义。

定义1 令时间戳域为T、元组标识域为ID，带变元双时态域D^B和无变元双时态域D^S定义如下：

$$D^B=\{(TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r)\in T\times T\cup\{UC\}\times T\times T\cup\{NOW\}\times ID\mid(TT_r^{\vdash}=UC\vee TT_r^{\vdash}\leqslant TT_r^{\dashv})\wedge(VT_r^{\vdash}=NOW\vee VT_r^{\vdash}\leqslant VT_r^{\dashv})\} \quad (1)$$

$$D^S=\{(T_r,TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r)\in T^4\times ID\mid TT_r^{\vdash}\leqslant TT_r^{\dashv}\wedge VT_r^{\vdash}\leqslant VT_r^{\dashv}\} \quad (2)$$

由定义1可知D^B是双时态域四种时态数据类型的集合，包括增长楼梯形、增长矩形、固定楼梯形和固定矩形。而D^S仅由固定的矩形区域组成。

定义2 令$R\subseteq D^B$，类型Type$=\{1,2,3,4\}$，数据变换$\Gamma_D:2^{D^B}\to 2^{D^S\times Type}$定义如下：

$$\Gamma_D(R)=\{\Gamma_r((T_r,TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r))\mid\langle TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r\rangle\in R\}, \quad (3)$$

其中

$$\Gamma_r((TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r))=\begin{cases}TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r,1)\text{if } TT_r^{\vdash}=UC\wedge VT_r^{\dashv}=NOW\\TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r,2)\text{if } TT_r^{\vdash}=UC\wedge VT_r^{\dashv}\neq NOW\\TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r,3)\text{if } TT_r^{\vdash}\neq UC\wedge VT_r^{\dashv}=NOW\\TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r,4)\text{if } TT_r^{\vdash}\neq UC\wedge VT_r^{\dashv}\neq NOW\end{cases}$$

由Γ_r函数知：经数据变换后可得到四种类型的双时态数据区域，分别对应图1所示的R1～R4四棵R树。

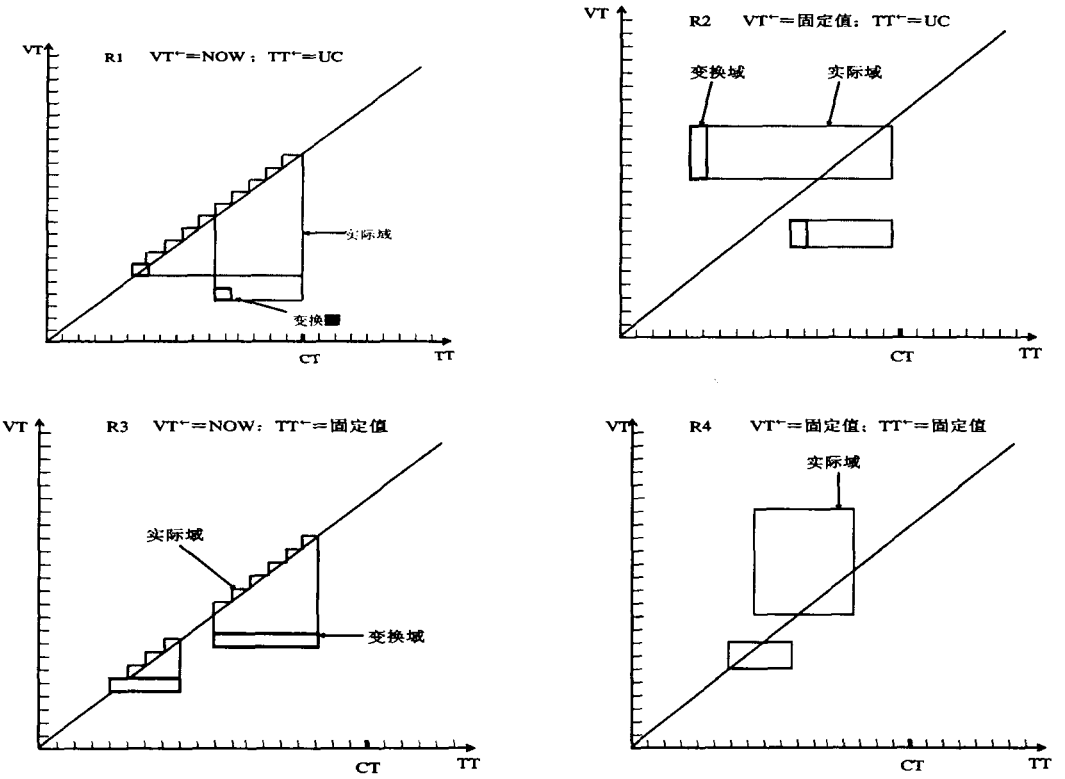


图1 四种类型的双时态R树

2.3 查询变换

双时态数据经过适当的数据变换后，消除了时态变元NOW和UC，使得带有增长楼梯形和增长矩形的双时态数据区域变为规则的矩形，从而可以利用现有的R树算法来实现。经过数据变换后，必然要对查询条件做出相应的变换，才能与变换前的查询语义等价。下面给出查询变换的两个相关定义。

定义3 若$(TT_{r1}\leqslant TT_{r2})\wedge(TT_{r1}\geqslant TT_{r2})\wedge(VT_{r1}\leqslant VT_{r2})\wedge(VT_{r1}\geqslant VT_{r2})$条件为真，则称数据$(TT_{r1},TT_{r1},VT_{r1},VT_{r1})$与$(TT_{r2},TT_{r2},VT_{r2},VT_{r2})$相交。令$q=(TT_q,TT_q,VT_q,VT_q)$且$R\subseteq D^B$，以矩形查询框$q$、当前时间$CT$为参数在$R$上的相交查询$Intersect^B$定义如下：

$$Intersect^B[q,CT](R)=\{id_r\mid\alpha\wedge(\beta\vee\gamma\vee\delta\vee\omega)\} \quad (4)$$

其中

$\alpha=(\langle TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv},id_r\rangle\in R)$

$\beta=(TT_r^{\vdash}=UC\wedge VT_r^{\dashv}=NOW\wedge TT_q^{\dashv}\geqslant VT_q^{\vdash}\wedge q\cap\langle TT_r^{\vdash},CT,VT_r^{\vdash},CT\rangle)$

$\gamma=(TT_r^{\vdash}=UC\wedge VT_r^{\dashv}\neq NOW\wedge q\cap\langle TT_r^{\vdash},CT,VT_r^{\vdash},VT_r^{\dashv}\rangle)$

$\delta=(TT_r^{\vdash}\neq UC\wedge VT_r^{\dashv}=NOW\wedge TT_q^{\dashv}\geqslant VT_q^{\vdash}\wedge q\cap\langle TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},TT_r^{\dashv}\rangle)$

$\omega=(TT_r^{\vdash}\neq UC\wedge VT_r^{\dashv}\neq NOW\wedge q\cap\langle TT_r^{\vdash},TT_r^{\dashv},VT_r^{\vdash},VT_r^{\dashv}\rangle)$

定义4 令$q=\langle TT_q^{\vdash},TT_q^{\dashv},VT_q^{\vdash},VT_q^{\dashv}\rangle$且$R\subseteq D^S$，在$R$

上以查询矩形 q 为参数的相交查询定义如下：

$$Intersect^s[q](R)=\{id_r \mid \langle TT_r^-, TT_r^+, VT_r^-, VT_r^+, id_r\rangle \in R \wedge q \cap \langle TT_r^-, TT_r^+, VT_r^-, VT_r^+\rangle\} \quad (5)$$

根据定义 3 和定义 4，可以定义 4R 树查询变换 Γ_q，根据上节数据变换产生的不同类型而有相应的查询变换。这个变换把在原始数据上的一个相交查询转化为在变换数据上的二个或四个相交查询。

定义 5 给出如下初始定义：

$R \subseteq D^B$

$S=\Gamma_D(R)$

$S_i=\{\langle TT_r^-, TT_r^+, VT_r^-, VT_r^+, id_r\rangle \mid \langle TT_r^-, TT_r^+, VT_r^-, VT_r^+, id_r, i\rangle \in S\}, i=1,2,3,4$

$q=\langle TT_q^-, TT_q^+, VT_q^-, VT_q^+\rangle$

$q_1=\langle 0, TT_q^+, 0, VT_q^+\rangle$

$q_2=\langle 0, TT_q^+, VT_q^-, VT_q^+\rangle$

$q_3=\langle \max(TT_q^-, VT_q^-), TT_q^+, 0, VT_q^+\rangle$

$q_4=\langle TT_q^-, TT_q^+, VT_q^-, VT_q^+\rangle$

4R 查询变换 $\Gamma_q: [2^{D^B} \to 2^{ID}] \to [2^{D^S \times Type} \to 2^{ID}]$：定义如下：

$$\Gamma_q(Intersect^B[q,CT](S)= \begin{cases} \bigcup_{i=1,2,3,4} Intersect^S[q_i](S_i) \text{ if } TT_q^- \geqslant VT_q^- \\ \bigcup_{i=2,4} Intersect^S[q_i](S_i) \text{ if } TT_q^- < VT_q^- \end{cases} \quad (6)$$

根据查询条件的不同，查询可能遍及两棵或四棵查询树，且在每棵树上有不同的操作。图 2 和图 3 描述了在 R1～R4 上的原始查询框与对应的变换查询框。

定理 1 对于任意 $q=\langle TT_q^-, TT_q^+, VT_q^-, VT_q^+\rangle$ 和任意数据集 $R \subseteq D^B$，有

$$Intersect^B[q,CT](R)=\Gamma(Intersect^B[q,CT](\Gamma_D(R)) \quad (7)$$

定理 1 说明，经过数据变换和查询变换所得的 4R 树查询与原有的矩形区域查询是等价的。文[6]给出了该定理正确性的证明。

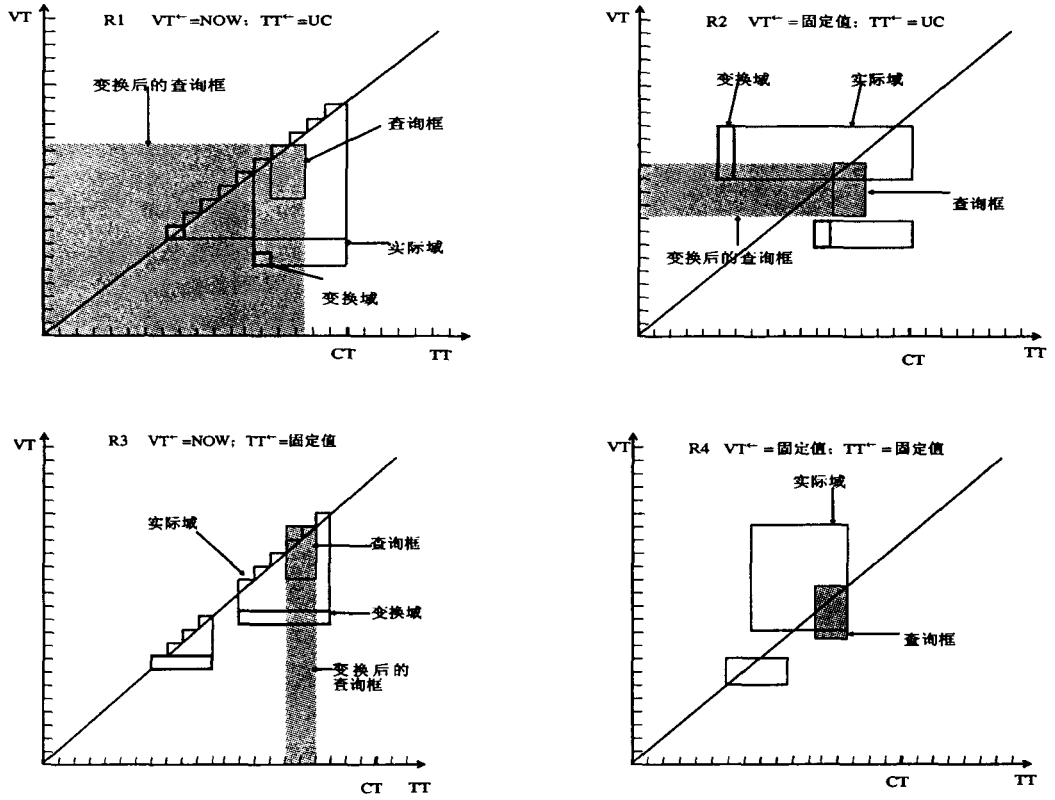


图 2 4R 索引在 4 棵树上的查询变换

3 4R 索引的实现

实现 4R 索引的方式可分为两种：

• 基于支持 R 树索引的 DBMS 由于 4R 树是由四棵 R 树组成，经过数据变换后，双时态数据区域不存在增长楼梯形和增长矩形这些不规则形状。因而可以直接使用支持 R 树索引的 DBMS，而无须扩充 DBMS 内核，只需进行适当的数据变换和查询变换即可实现双时态数据索引。

• 基于不支持 R 树索引的 DBMS 为了避免扩充该类 DBMS 的内核，可以采用文后描述的 B4Rindex 索引模型。即自行实现 4R 算法并作为时态数据库中间件 TimeDB 的一个组件，形成一个扩充的 TimeDB 中间件。

目前直接支持 R 树的主流商业 DBMS 较少，因而第二种方法更具有现实意义。为了避免从最低层的索引文件做起，本文引用了由美国 Berkeley 大学提出的通用搜索树(Generalized Search Tree，GiST)来实现与数据索引密切相关的底层工作[9, 10]，从而可以更专注于如何实现上层的 4R 树变换以及双时态数据语义等处理工作。

4 B4Rindex 索引与 TimeDB 的无缝结合

4.1 TimeDB 数据库中间件

TimeDB 是一个 ATSQL2 语句的编译器，它的主要功能是将 ATSQL2 语句翻译成语义一致的标准 SQL 语句。TimeDB 是非时态 DBMS 的一个前端构件，它作为单独的一层架构在非时态 DBMS 上。从数据模型的角度来看，它实现了在非时态数据模型上扩展成为时态数据模型。提供给用户

一个时态 DBMS 接口(即 ATSQL2 语言),用户通过该接口可以实现时态数据库的管理和时态数据的存取[11]。TimeDB 中间件作为一个非时态 DBMS 前端构件的形式,有利于使用底层主流商业 DBMS 提供的各种完善管理功能,例如数据的持久性、多用户下的并发控制、数据备份和灾难恢复等。这些功能均可以由底层商业 DBMS 实现,从而大大减少了代码的编制量。图 4 是一个典型的应用示意图。

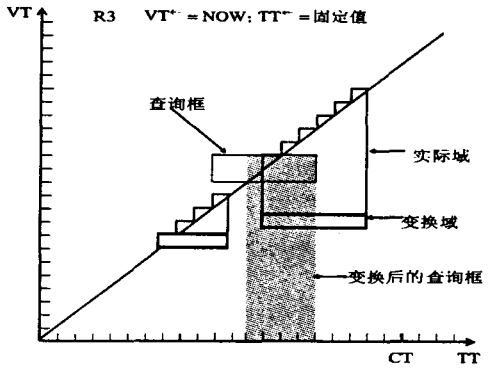


图 3 R3 树上的另一种查询变换

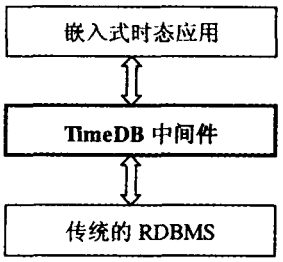


图 4 一个典型应用 TimeDB 中间件的架构

在 TimeDB 中,有效时间维和事务时间维采用"正交"的处理方式,这就意味着 TimeDB 既能处理单个有效时间或单个事务时间的单时态数据,又能处理同时具有有效时间和事务时间的双时态数据。

但是,当前的时态数据库中间件缺乏对双时态数据进行索引的有效算法。使用中间件把 ATSQL2 翻译成标准的 SQL 语句,从而导致双时态字段的索引仍然使用传统 RDBMS 的索引算法(大多数使用 B^+ 树)。然而,B^+ 树索引适合于线性的一维数据,缺乏对具有二维特性的双时态数据的支持,很难建立双时态数据字段的索引。引入第 2 节描述的 4R 树索引,通过数据变换、查询变换把不规则的二维时间区域变换成规则的矩形区域,从而可以利用成熟的 R 树索引算法来实现双时态数据的索引。

4.2 B4Rindex 索引与扩充的 TimeDB 中间件

利用 GiST 树优越的可扩展特性,结合 4R 树的数据变换、查询变换技术,可以方便地开发出 B4Rindex 索引。图 5(a)是 B4Rindex 索引模型的系统架构图。利用 B4Rindex 最上层的"TimeDB 接口层",可以与 TimeDB 中间件进行无缝连接(图 5(b)),从而在原有的 TimeDB 中间件基础上组成一个"扩充 TimeDB 中间件"组件,完成了时态中间件 TimeDB 对双时态索引的扩充。

TimeDB 接口层至少具有如下两个功能:

· 建立索引。根据 5 元组$\langle id, TT^-, TT^+, VT^-, VT^+\rangle$建立一条记录的 B4Rindex 索引,其中 id 是该记录的标识字段。

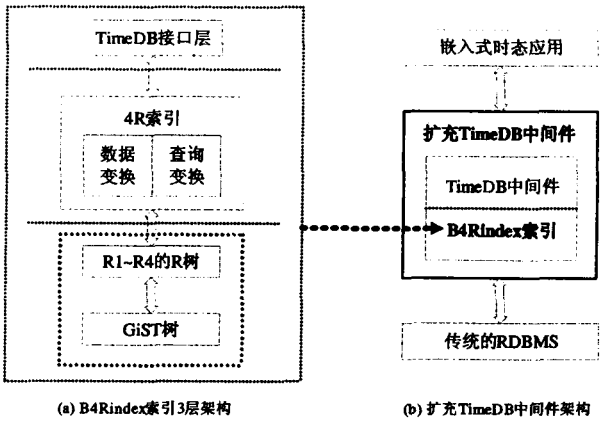


图 5 B4Rindex 索引和扩充 TimeDB 中间件的系统架构

· 查询。给出查询条件 $q=\langle TT_q^-, TT_q^+, VT_q^-, VT_q^+\rangle$,利用 B4Rindex 索引返回包含 id 的搜索结果集。如果需要双时态数据之外的字段,在 B4Rindex 返回结果后,TimeDB 中间件可根据 id 字段集向底层 RDBMS 查询。由于 id 字段在 RDBMS 有相应的 B^+ 树索引,因而此查询是非常快速的。

5 实验结果

本实验根据若干参数随机生成实验数据,所采用的主要参数如下:样本量、含有时态变元 NOW 数据的比例、Insert/Delete 的比例以及查询语句的比例。表 2 是不同参数的描述及其使用值。

表 2 实验参数值

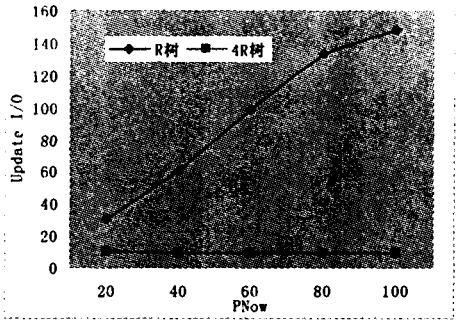
参数	描述	使用值
Total	实验样本量	50000,100000
PNow	带变元 NOW 数据的百分比	20,40,60,80,100
PIns	插入语句的百分比 (PDel =1 - PIns)	10~100
PQuery	查询语句的百分比	10~100

B4Rindex 最主要的优点是它可以很好地处理带时态变元 NOW 和 UC 的双时态数据,图 6 给出 PNow 对使用最大时间戳的 R 树[4]与使用变换技术的 4R 树索引的比较结果(由于 B^+ 树的一维线性特性难以改造为支持双时态数据的高效索引,故本实验忽略 B^+ 树的比较)。实验所采用的参数值如下:Total = 100000, PNow = 20% ~ 100%, PIns = 70%, PQuery=30%。

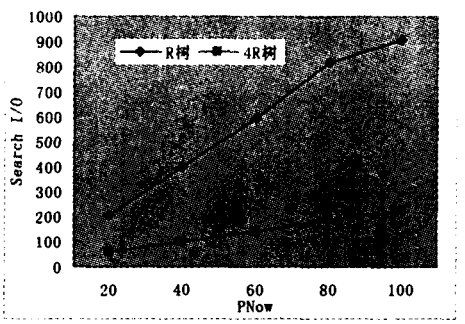
由实验结果可知,PNow 值越大,采用 4R 技术的 B4Rindex 模型在 I/O 次数上优势越明显。当 PNow≥80% 时,B4Rindex 的索引效率是采用最大时间戳 R 树的 4 倍以上。

结束语 传统的数据库索引技术(如 B^+ 树)不能很好地对双时态数据进行索引:一方面,B^+ 树不能直接索引带时态变元 NOW 和 UC 的数据,且 B^+ 树很难有效地改造以支持时态变元[2];另一方面,B^+ 树索引的数据是一维线性的数据,而双时态数据具有二维特性,故不能很好地利用该特性的自然语义进行优化。借鉴应用于空间索引的 R 树技术,经过适当的数据变换、查询变换成为 4R 树,它可以查询带时态变元 NOW 和 UC 的双时态数据。结合时态数据库中间件 Time-

DB.把 B4Rindex 作为 TimeDB 的一个组件而形成一个新的"扩充 TimeDB"组件。实验证明:对比 B^+ 树、R 树索引技术,B4Rindex 模型是一种高效的双时态索引技术。



(a) Update I/O 次数



(b) Search I/O 次数

图 6 实验结果

由于变元 NOW 语义的复杂性,关于 NOW 查询语义的过去、现在和将来的绑定问题仍处在研究中,目前尚未有统一的定论[12, 13]。因而,一旦 NOW 查询语义本身出现较大的变动,可能需要修改索引模型以适应新的需求。对此,我们将会在另文进行单独的讨论。

参 考 文 献

1 Tang Yong, et al. Bitemporal extensions to non-temporal RDBMS in distributed environments. In: The 8th Intl. Conf. on Computer Supported Cooperative Work in Design Proceedings, Volume 2, Xiamen, China, 2004. 370～373

2 Salzberg B, Tsotras V J. A Comparison of Access Methods for Temporal Data. TimeCenter Publications:[TR18]. 1997

3 Beckmann N, et al. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proc. of ACM SIGMOD, 1990. 322～331

4 Kumar A, Tsotras V J, Faloutsos C. Designing access methods for bitemporal databases. IEEE Transactions on Knowledge and Data Engineering, 1998, 10(1): 1～20

5 Bliujute R, Jensen C S, Saltenis S, et al. R-tree based indexing of now-relative bitemporal data. In: Proc. of the 24th VLDB Conf. New York, USA, 1998. 345～356

6 Bliujute R, Jensen C S, Saltenis S, et al. Light-weight indexing of bitemporal data. In: Proc. of the 12th International Conf. on Scientific and Statistical Database Management, Berlin, 2000. 125～138

7 汤庸, 汤娜, 叶小平. 时态信息处理技术研究综述. 中山大学学报(自然科学版), 2003, 42(4): 5～9

8 Snodgrass R T. The Temporal Query Language TQuel. ACM-TODS, 1987, 12(2): 247～298

9 Hellerstein J M, Naughton J F, Pfeffer A. Generalized Search Tree for Database Systems. In: Proc. of the 21st VLDB Conf. Zurich, Switzerland, 1995. 562～573

10 PostgreSQL 8.1devel 文档. http://www.pgsqldb.org/pgsqldoc-cvs/gist.html

11 汤庸, 汤娜, 等. 时态知识数据模型研究及应用. 中山大学学报(自然科学版), 2004, 43(6): 62～66

12 叶小平, 汤庸. 时态变量"NOW"语义及相应时态关系运算. 软件学报, 2005, 16(5): 838～845

13 王路邦, 叶小平, 等. 时间变元 now 的绑定算法. 中山大学学报(自然科学版), 2004, 43(s1): 152～154

(上接第 44 页)

6 Kleinrock L. Queuing System. New York: Wiley, 1975

7 Leland L E, Taqqu M S, Willinger W. On the Self-Similar Nature of Ethernet Traffic(extend version). IEEE/ACM Trans. Networking, 1994, 2(1): 1～15

8 Forest V S, Melamed B. Traffic Model for Telecommunications Networks. IEEE Communication Magazine, March 1994. 70～81

9 Cruz K L. A Calculus for Network Delay. I. Network Elements in Isolation. IEEE Trans. Theory, 1991, 37: 114～131

10 Cruz R L. A Calculus for Network Delay. II. Network analysis. IEEE Trans. Theory, 1991, 37: 132～141

11 Shaikh A, Rexford J, Shin K G. Evaluating the impact of stale link state on quality-of-service routing. IEEE/ACM Transactions on Networking, 2001, 9(2): 162～176

12 Shreedhar M, Varghese G. Efficient fair queueing using deficit round-robin [J]. IEEE/ACM Transactions on Networking, 1996, 4(3): 375～385

13 Georgiadis L, Guerin R, Parekh A. Optimal multiplexing on a single link: delay and buffer requirements. In: Proc. IEEE INFOCOM'97, also in IEEE Trans. Infor. Theory, 1997, 43(5)

14 Demers A, Keshav S, Shenker S. Analysis and simulation of a fair queueing algorithm. Journal of Internetworking: Research and Experience, Oct. 1990. Also In: Proc. ACM SIGCOMM' 89, 1989