一种健全和有效的双时态索引技术①

A Robust and Efficient Bitemporal Index Technology

杨玉军　杨夷梅　(怀化学院 计算机科学与技术系 湖南 怀化 418008)

摘　要： 能为时态数据库设计出一种比现有的时态索引技术更健全且更有效的索引技术是当今时态数据库研究的主要问题，本文结合 2R-tree 索引、GR-tree 索引和 G4R-tree 索引，提出了一种新的索引技术，称为 G2R-tree 索引。它能有效地处理所有类型的双时态数据，而且性能与 4R-tree 索引和 GR-tree 索引相当。

关键词： 双时态数据库 双时态索引 查询性能

1　引言

为了提高检索时态数据的速度，最有效的途径就是研制出一种可实现的高效的适合时态特性的索引技术。自 1984 年 A.Guttman 第一个提出可应用于检索时态数据的索引技术 R-tree 以来，时态索引技术经过了 20 多年的发展，同时对时态索引的评价标准也得到了发展，针对时态索引技术的查询速度受 I/O 操作影响的规律和实际对查询速度的要求高于索引创建时的创建速度的要求，本文对时态索引的查询速度和存储速度进行科学地处理，以降低小量的存储速度为代价来换取更高的查询速度，这样更符合实际应用的要求。本文以 2R-tree 为基础来形成索引的理论，以 R*-tree 优良的数据结构和 GR-tree 先进的分裂算法来构建索引的数据结构和相应的算法，以 G4R-tree 对将来时态数据的检索思想来实现索引对将来时态数据检索的支持，这样刚好结合了上述三种时态索引技术的优点，摒弃了它们的缺点，在实际应用中提高了对时态数据查询的速度，由于该索引是基于 2R-tree 的，为了叙述的方面，本文称之为 G2R-tree(General 2R-tree)索引 。

2　索引的数据结构

G2R-tree 索引的结构与 B-tree 和 R-tree 类似，只不过 B-tree 用于一维数据的索引，R-tree 用于二维数据的索引，而 G2R-tree 把数据存储在 2 棵普通的 R-tree 之上，结构完全与 2R-tree 相同，但 G2R-tree 是 2R-tree 在处理带变元的时态数据方面的扩充。G2R-tree 索引中的 R-tree 也是一种平衡树结构，每个叶子节点的深度是一样的，非叶子节点不存储具体的实际数据(数据的地址)，只有叶子节点才存储实际数据。

3　索引的数据转换

在双时态数据元组中，一般都含有一个双时态域(即 4 个时间戳)和一个指针域，数据变换的主要目的是消除元组中的变元 UC 和 Now，以便能用 R-tree 索引进行处理，因此双时态数据域($TT^{\rightarrow}$,$TT^{\leftarrow}$,$VT^{\rightarrow}$,$VT^{\leftarrow}$)按和是否为固定时间值将双时态元组分为 4 种类型，如表 1 所示。经过变换后的数据都是无变元的，变换后的双时态数据也是静态的。

表 1　双时态数据的 4 种类型

类型号	事务时间起点	事务时间终点	有效时间起点	有效时间终点
(1)	TT1	TT2	VT1	VT2
(2)	TT1	UC	VT1	VT2
(3)	TT1	TT2	VT1	Now
(4)	TT1	UC	VT1	Now

下面具体给出从带变元的双时态数据域到无变元的双时态域的变换，首先给出相关的定义。

① 基金项目:怀化学院科研项目(HHUY2008-17)
收稿时间:2008-08-18

定义 1．已知时间点域为 T，数据元组的指针域为 ID，双时态域D^B和无变元的双时态域D^S定义如下：

$$D^B \triangleq \{(TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r) \in T \times \{T \cup UC\} \times T \times \{T \cup Now\} \times ID \mid (TT_r^{\leftarrow} = UC \vee TT_r^{\rightarrow} \le TT_r^{\leftarrow}) \wedge (VT_r^{\leftarrow} = Now \vee TT_r^{\rightarrow} \le VT_r^{\leftarrow})\}$$

$$D^S \triangleq \{(TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r) \in T \times T \times T \times T \times ID \mid TT_r^{\rightarrow} \le TT_r^{\leftarrow} \wedge TT_r^{\rightarrow} \le VT_r^{\leftarrow}\}$$

由定义 1 可知双时态域D^B是 4 种时态数据类型的集合，包括增长矩形，增长楼梯形，固定楼梯形和固定矩形，而无变元的双时态域只包括固定的矩形。

定义 2．令$R \subseteq D^B$，类型 Type={1，2，3，4}，那么 G2R-tree 的数据变换$\Gamma^D : 2^{D^B} \to 2^{D^S \times Type}$定义如下：

$$\Gamma_D(R) \triangleq \{\Gamma_r((TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r)) \mid (TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r) \in R\},$$

$$\Gamma_r((TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r)) \triangleq \begin{cases} (TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, 1) & if\ (TT_r^{\leftarrow} \neq UC) \wedge (VT_r^{\leftarrow} \neq Now) \\ (TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, 2) & if\ (TT_r^{\leftarrow} = UC) \wedge (VT_r^{\leftarrow} \neq Now) \\ (\max(TT_r^{\rightarrow}, VT_r^{\rightarrow}), TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, 3) & if\ (TT_r^{\leftarrow} \neq UC) \wedge (VT_r^{\leftarrow} = Now) \\ (\max(TT_r^{\rightarrow}, VT_r^{\rightarrow}), \max(TT_r^{\rightarrow}, VT_r^{\rightarrow}), VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, 4) & if\ (TT_r^{\leftarrow} = UC) \wedge (VT_r^{\leftarrow} = Now) \end{cases}$$

根据双时态数据有效时间和事务时间的结束值不同，定义 2 中的数据变换函数分别将其变换到 R1 和 R2 树上，如图 1,其中虚线区域为变换前的值，实线区域为变换后的值，通过变换消除了变元 Now 和 UC，表 2 中的(a1)、(b)和(c10 类数据的变元变换后的值等于相应的有效时间和事务时间的起始值,若是(a2)和(c2)类型数据,通过$\max(TT_r^{\rightarrow}, VT_r^{\rightarrow})$变换处理后,则有$TT_r^{\rightarrow} = VT_r^{\rightarrow}$，其余变换相同。

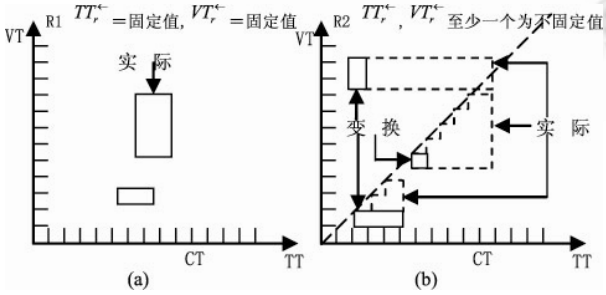

图 1　G2R 树数据变换前后图(CT 表示当前时间)

R1 树是静态矩形，所以不需要变换，可以直接利用 R 树进行索引。

R2 树实际上是 G2R-tree 对 G4R-tree 的 R1，R2 和 R3 的科学合并，同时完全吸收了 G4R-tree 能处理将来数据的优点。

G4R-tree 的 R1 树对双时态数据变换前为增长楼梯形的数据进行索引，其处理过程在 G2R-tree 的 R2 中完全实现了，若时态数据为表 1 中的(a1)类($TT_r^{\rightarrow} \ge VT_r^{\rightarrow}$)，经过变换后和分别映射到相应时间的起始值，即$TT_r^{\leftarrow} = TT_r^{\rightarrow}$和$VT_r^{\leftarrow} = VT_r^{\rightarrow}$，则变换后的数据结构为$(TT_r^{\rightarrow}, TT_r^{\rightarrow}, VT_r^{\rightarrow}, VT_r^{\rightarrow}, id_r, 2)$，若为(a2)类($TT_r^{\rightarrow} < VT_r^{\rightarrow}$),变换后的双时态数据为$(VT_r^{\rightarrow}, VT_r^{\rightarrow}, VT_r^{\rightarrow}, VT_r^{\rightarrow}, id_r, 2)$，在变换中用$\max(TT_r^{\rightarrow}, VT_r^{\rightarrow})$来处理(a1)和(a2)这两种情况，这里的“2”代表此数据类型是由(a)类双时态数据变换而来，把它归到第“2”类时态数据类型，在建立索引时是插入到 R2 中，同时第 3 类和第 4 类双时态数据类型的数据也插入到 R2 中。这里用变换后为点的数据代替变换前为楼梯形的数据也大大节省了存储空间。

G4R-tree 的 R2 树对双时态数据变换前为增长矩形的数据进行索引，其处理过程也在 G2R-tree 的 R2 中实现了，经过变换后为$(TT_r^{\rightarrow}, TT_r^{\rightarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, 3)$，变换后$TT_r^{\leftarrow} = TT_r^{\rightarrow}$，变换前为矩形的数据在变换后变为了与 TT 时间轴垂直的直线了，这样也节省了存储空间。

G4R-tree 的 R3 树对双时态数据变换前为沿着 VT＝TT 增长的楼梯形数据进行索引，其处理过程同样也在 G2R-tree 的 R2 中完全实现了，根据$TT_r^{\rightarrow} \ge VT_r^{\rightarrow}$和$TT_r^{\rightarrow} < VT_r^{\rightarrow}$的关系，双时态数据经过变换后分别为$(TT_r^{\rightarrow}, TT_r^{\rightarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, 4)$和$(VT_r^{\rightarrow}, TT_r^{\rightarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, 4)$，变换后$VT_r^{\leftarrow} = VT_r^{\rightarrow}$，变换前为矩形的数据在变换后变为与 TT 时间轴平行的直线了，这样也可以节省存储空间。

经过变换将带有 Now 和 UC 变元的值映射到固定值，便于存储在支持 R-tree 的数据库中，也便于进行 R-tree 检索，同时也克服了 R-tree 不能处理变元的不足和 2R-tree 只有处理 UC 变元的不足，却继承了 4R-tree 和 G4R-tree 对所有变元都能处理以及对将来时态数据也能处理的优点。对这些变换后的值的查询，G2R-tree 与 4R-tree 和 G4R-tree 一样要执行查询变换，以便能得到正确的查询结果，下面将介绍查询变换的整个过程。

4　索引的查询转换

在 4R-tree 和 G4R-tree 索引查询中，对查询的条件除了满足$TT_q^{\rightarrow} \le TT_q^{\leftarrow}$和$VT_q^{\rightarrow} \le VT_q^{\leftarrow}$外，还必须满足$TT_q^{\leftarrow} \le CT$(其中 CT 表示当前时间)，即查询只限于当前的和历史的数据，表 2 中的(a2)类数据在未生效之前是不可查询的，将来的情况也是不可查询的，这就给时态查询带来

了很大的局限性，其实(a1)、(a2)和(b)类数据在将来时间里是可能生效的，如果仅把查询局限在当前时间以前，不符合客观实际，所以能查询将来的数据有助于预测将来的情况，但是查询的结构是一个大概的结果，带有一定的概率性，不能保证完全是正确的，查询结果是根据先前的数据预算出来的，这些结果可以由一些算法，比如神经算法和遗传算法等等，来提高它们的可靠性。在查询将来数据时，G2R-tree 通过将事务时间的变元 UC 转换为将来时间来查询，将来数据的查询结果是表 2 中的(a1)、(a2)和(b)类数据满足条件的元组集合，另外 G4R-tree 也是这么实现的。

在二维数据的索引查询中，通常是通过判断查询条件 q 和双时态数据是否相交来取得查询结果，下面先给出查询条件 q 与双时态数据相交的定义。

定义 3.已知 r1 和 r2 为双时态数据的任意两个元组，若条件$(TT_{r_1}^{\rightarrow} \le TT_{r_2}^{\leftarrow}) \wedge (TT_{r_1}^{\leftarrow} \ge TT_{r_2}^{\rightarrow}) \wedge (VT_{r_1}^{\rightarrow} \le VT_{r_2}^{\leftarrow}) \wedge (VT_{r_1}^{\leftarrow} \ge VT_{r_2}^{\rightarrow})$为真，则称双时态数据$(TT_{r_1}^{\rightarrow}, TT_{r_1}^{\leftarrow}, VT_{r_1}^{\rightarrow}, VT_{r_1}^{\leftarrow}) \cap (TT_{r_2}^{\rightarrow}, TT_{r_2}^{\leftarrow}, VT_{r_2}^{\rightarrow}, VT_{r_2}^{\leftarrow})$相交。又令$q=(TT_q^{\rightarrow}, TT_q^{\leftarrow}, VT_q^{\rightarrow}, VT_q^{\leftarrow})$且$R \subseteq D^B$，那么以查询矩形 q 和当前时间 CT(Current Time)为参数，在 R 上的相交查询 IntersectB 定义如下：

$$
\begin{aligned}
Intersect^B[q,CT](R) \triangleq \{id_r \mid (&TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r) \in \\
&R \wedge ((TT_r^{\leftarrow} \ne UC \wedge VT_r^{\leftarrow} \ne Now \wedge (q \cap \\
&(TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}))) \vee (TT_r^{\leftarrow} = UC \wedge VT_r^{\leftarrow} \ne Now \wedge \\
&(q \cap (TT_r^{\rightarrow}, \max(TT_r^{\leftarrow}, CT), VT_r^{\rightarrow}, VT_r^{\leftarrow}))) \vee \\
&(TT_r^{\leftarrow} \ne UC \wedge VT_r^{\leftarrow} = Now \wedge (q \cap (\max(TT_r^{\rightarrow}, VT_r^{\rightarrow}), \\
&TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow})) \wedge TT_q^{\leftarrow} \ge VT_q^{\rightarrow}) \vee (TT_r^{\leftarrow} = UC \wedge VT_r^{\leftarrow} \\
&= Now \wedge (q \cap (\max(TT_r^{\rightarrow}, VT_r^{\rightarrow}), \max(TT_r^{\leftarrow}, CT), VT_r^{\rightarrow}, \\
&max(TT_r^{\leftarrow}, CT)) \wedge TT_q^{\leftarrow} \ge VT_q^{\rightarrow}))\}
\end{aligned}
$$

其中第一行限制了查询结果的元组必须在元组集 R 中，以定义 2 定义的四种数据类型为序，接下来的每一行分别判断四类数据中的一种数据类型是否与查询框相交，若相交则返回查询结果的元组集合，第一个逻辑值(第二行)判断了静态矩形是否与查询框相交，第二个逻辑值判断了增长的矩形，第三个和第四个逻辑值分别判断了静态楼梯形和增长的楼梯形是否与查询框相交。其中在判断增长的楼梯形时，查询框右下角必须处在 TT = VT 线之下或在线上，条件就是对此作出的限制，另外操作使查询 q 既可以查询过去的时态数据又可以查询将来的时态数据。

与定义 3 类似，同样可以定义查询 q 与无变元的时态数据域相交的定义，其查询结果将不再与当前时间 CT 有关，下面给出具体的定义。

定义 4. 令$q=(TT_q^{\rightarrow}, TT_q^{\leftarrow}, VT_q^{\rightarrow}, VT_q^{\leftarrow})$且$S \subseteq D^S$，在 S 上的相交查询且以查询矩形 q 为参数的定义如下：

$$Intersect^S[q](S) \triangleq \{id_r \mid (TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r) \in S \wedge (q \cap (TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}))\}$$

在有了定义 3 和定义 4 后，现在可以定义 G2R-tree 的查询变换了，它将随着表 1 中的数据类型的不同而有不同的查询变换，这个将查询条件分别映射到四类双时态数据上，其中q_1, q_2, q_3, q_4分别为在(1)、(2)、(3)、(4)类数据上执行的查询经过变换以后的查询条件，S_1, S_2, S_3, S_4分别为双时态数据(1)、(2)、(3)、(4)类变换以后的相应类型的数据。

由于双时态数据的(1)类数据存储在 G2R-tree 的 R1 树上，而(2)、(3)、(4)类数据存储在 G2R-tree 的 R2 树上，所以查询都涉及到两棵 R-tree 树，且在每棵 R-tree 树上有不同的操作，图 2 描述了每棵 R-tree 树上的查询和相应的查询变换，图 3 描述了 G2R-tree 的 R2 树上对第(4)数据的查询过程，从图可以看出，在没有条件的情况下，查询框将会与两个变换后的数据框相交，但是在考虑条件的情况下，下面那个数据框在变换后不可能超出 TT = VT 线，因而不会和查询框相交，这样做的目的是使查询结果更符合实际和更加精确。

表 2　双时态数据类型的 6 种情况

类　型		事务时间起点	事务时间终点	有效时间起点	有效时间终点	条　件
(a)	(a1)	TT_1	UC	VT_1	VT2	VT1≤TT1
	(a2)	TT_1	UC	VT_1	VT2	VT1>TT1
(b)		TT_1	TT2	VT_1	VT2	
(c)	(c1)	TT_1	UC	VT_1	Now	VT1≤TT1
	(c2)	TT_1	UC	VT_1	Now	VT1>TT1
(d)		TT_1	TT2	VT_1	Now	

定义 5.一些初始化定义如下：

$R \subseteq D^B$

$S = \Gamma_D(R)$

$S_i = \{(TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r) \mid (TT_r^{\rightarrow}, TT_r^{\leftarrow}, VT_r^{\rightarrow}, VT_r^{\leftarrow}, id_r, i) \in S\}$

其中 $i = 1,2,3,4$

$q = (TT_q^{\rightarrow}, TT_q^{\leftarrow}, VT_q^{\rightarrow}, VT_q^{\leftarrow})$

$q_1 = (TT_q^{\rightarrow}, TT_q^{\leftarrow}, VT_q^{\rightarrow}, VT_q^{\leftarrow})$

$q_2 = (0, TT_q^{\leftarrow}, VT_q^{\rightarrow}, VT_q^{\leftarrow})$

$q_3 = (\max(TT_q^{\rightarrow}, VT_q^{\rightarrow}), TT_q^{\leftarrow}, 0, VT_q^{\leftarrow})$

$q_4 = (0, TT_q^{\leftarrow}, 0, VT_q^{\leftarrow})$

G2R-tree 相交查询变换 $\Gamma_q:[2^{D^B}\to 2^{ID}]\to[2^{D^S\times Type}\to 2^{ID}]$ 定义如下：

$$\Gamma_q(Inter\sec t^B[q,CT])(S)\triangleq\begin{cases}\bigcup_{i=1,2,3,4} Inter\sec t^S[q_i](S_i) & if\ TT_q^{\leftarrow}\geq VT_q^{\rightarrow}\\ \bigcup_{i=2,4} Inter\sec t^S[q_i](S_i) & if\ TT_q^{\leftarrow}<VT_q^{\rightarrow}\end{cases}$$

现在通过一个定理来说明 G2R-tree 查询变换和 G2R-tree 数据变换的结合会得到精确的查询结果和两个变换的等价性。

定理 1.对于任意查询 $q=(TT_q^{\rightarrow},TT_q^{\leftarrow},VT_q^{\rightarrow},VT_q^{\leftarrow})$，和任意数据集 $R\subseteq D^B$，有 $Inter\sec t^B[q,CT](R)=\Gamma(Inter\sec t^B[q,CT])(\Gamma_D(R))$。

下面给出定理 1 的具体证明过程可以参考文献[1]，这里从略。

下面介绍如何在变换后的 G2R-tree 上执行相应的查询操作：

(1)R1 树不涉及数据变换和查询变换，可以直接利用 R 树进行结果查询。

(2)R2 树索引所有带变元的双时态数据，首先，把查询矩形在 R2 树上从原来的大小扩大到 $(0,TT_q^{\leftarrow},0,VT_q^{\leftarrow})$，再查询 $(0,TT_q^{\leftarrow},0,VT_q^{\leftarrow})$ 是否和 R2 树上的数据相交来取得查询结果。

(3)根据不同的双时态数据类型，对查询结果进行检查，检查是否包含一些错误的数据：对图 2(b)类双时态数据，去除所有符合条件 $VT_r^{\leftarrow}<VT_q^{\rightarrow}$ 和 $VT_r^{\rightarrow}<VT_q^{\leftarrow}$ 的数据；对图 2(c)类双时态数据，去除所有符合条件 $TT_r^{\leftarrow}<TT_q^{\rightarrow}$ 的数据；对图 2(d)类双时态数据，去除所有符合条件 $VT_r^{\rightarrow}>TT_r^{\rightarrow}$ 的数据，即去除 TT＝VT 线以上的数据。

(4)结合 R1 和 R2 的查询结果，得到最后的查询结果。

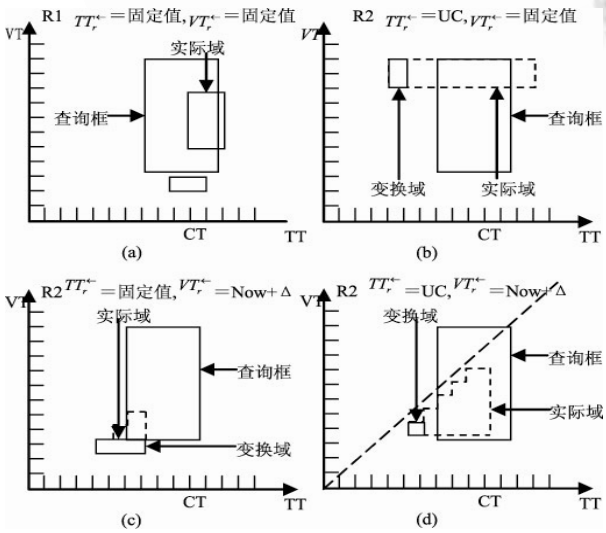

图 2G2R 索引在 R1 和 R2 树上查询时四种时态数据图

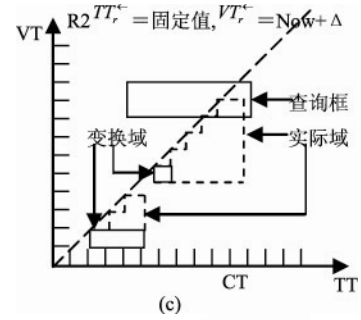

图 3　G2R 的 R2 树上对第(4)数据的查询

5　提高索引的查询速度的策略

影响一种索引的查询速度的原因是多方面的，相对而言，以下五个方面的影响是主要的和明显的：(1)时态数据所形成的覆盖面积、重叠面积和死区面积的大小；(2)查询时 I/O 操作次数；(3)查询时同时需要访问 R-tree 的数目；(4)索引本身产生的数据的多少；(5)缓冲区的大小和分配方法。

为了尽量提高索引的查询速度，G2R-tree 索引对这五个方面的任何一个方面都进行了考虑和优化，下面针对(4)和(5)进行详细的分析和说明。

查询时同时需要访问 R-tree 的数目也是影响查询速度的又一重要因素，4R-tree 索引和 G4R-tree 索引都是采用 4 棵 R-tree 实现的索引技术，查询时一般都会同时访问 4 棵 R-tree，相对内存储器来说，访问磁盘的速度是很慢的，同时访问 4 棵 R-tree 实际上就是进行了 4 次磁盘的读取操作，这样就增加了 I/O 操作次数，降低了查询速度。G2R-tree 索引是采用 2 棵 R-tree 实现的索引技术，因此在最坏的情况下，进行查询操作也只要同时访问 2 棵 R-tree，这样大大减少了 I/O 操作的次数，提高了查询速度，这也是 G2R-tree 索引提出的主要原因所在。

缓冲区的大小从理论上是可以无限大的，但实际上是有限的，固定的，特别在对不同种类的索引进行性能比较实验时，是严格规定的。为了测试更加公平，实验中为每种索引分配一样大的缓冲区，对基于多 R-tree 的索引，如 4R-tree 索引，每棵 R-tree 的缓存区也是平均分配的。据统计，缓冲区的利用率一般在 95％左右，都有部分缓冲区的浪费，在基于多 R-tree 的索引中，浪费的现象特别严重，一般情况下每棵 R-tree 的缓冲区不可能都得到很好的利用，最坏情况，只会利用到一个缓冲区，比如 4R-tree 索引就

可能有 3 棵 R-tree 的缓冲区是浪费的。由于基于多 R-tree 索引的缓冲区没有充分利用，其查询速度就会受到严重影响，而 G2R-tree 索引是基于 2 棵 R-tree 的，最大限度地降低了 R-tree 的数目，尽量提高了缓冲区的利用率，以提高查询速度。

6 结束语

本文详细地介绍了 G2R-tree 索引的双时态数据结构、数据转换、查询转换和一些优化策略，并通过定理说明了数据转换的正确性。G2R-tree 索引结合了 2R-tree、GR-tree、4R-tree 和 G4R-tree 四种索引，以 2R-tree 为基础来形成索引的理论依据，以 R*-tree 优良的数据结构和 GR-tree 先进的分裂算法来构建索引的数据结构和相应的算法，利用 G4R-tree 对将来时态数据的检索思想来实现 G2R-tree 索引对将来时态数据检索的支持，结合了上述四种时态索引技术的优点，摒弃了它们的缺点，提高了在实际应用中对时态数据查询的速度。

参考文献

1 杨玉军.时态索引技术及算法的研究.中南林业科技大学,2007.

2 汤庸.时态数据库导论. 北京:北京大学出版社,2004.1－64,123－145.

3 周风华,汤庸,康向锋.一种有效的双时态索引技术.计算机工程与应用, 2005,13:166－171.

4 Salzberg B, Tsotras V J. A Comparison of Access Methods for Temporal Data. TimeCenter TR-18.1997,To appear in ACM Computing Surveys.

5 Kumar V, Tsotras J, Faloutsos C. Designing Access Mothods for Bitemporal Databases. TR-3764, Dept of Computer SCI University of Maryland, 1997.