实时数据库缓冲区管理算法的设计和实现*

国志宏, 王宏安, 王　强
(中国科学院 软件研究所, 北京 100080)

摘　要: 分析了实时数据库的事务特征,对以往的研究成果进行了总结,以满足事务的按时完成比率(Success Ratio) 为目标,为实时系统设计了一种使用反馈控制思想的基于优先级的实时数据库缓冲区管理算法 FCLRU2-dl,并将该算法与常用的实时数据库事务调度算法和并发控制策略配合进行了测试和评估,证明了算法的优越性。实验中得到的另一个结论是在特定的事务调度算法和并发控制策略下,实时数据库不需要全部位于内存中,可以不是内存数据库。

关键词: 实时数据库系统; 缓冲区管理; 反馈控制
中图法分类号: TP392　　**文献标识码:** A　　**文章编号:** 1001-3695(2005)02-0121-04

Design and Implementation of Buffer Management Algorithm in Real-time Database Systems

GUO Zhi-hong, WANG Hong-an, WANG Qiang
(Institute of Software, Chinese Academy of Sciences, Beijing 100080, *China)*

Abstract: A feedback control based, deadline cognoscible buffer management algorithm for firm real-time database system is schemed out and evaluated. The experimental results indicate that this algorithm outperforms some RTDBs buffer management policies. Another result concluded from the experiment is, under the certain conditions, a real-time database need not be a main-memory database.

Key words: Real-time Database Systems; Buffer Management; Feedback Control

随着实时数据库应用的推广,其中各项关键技术如事务调度、并发控制等问题受到了越来越多的研究,但是实时数据库缓冲区管理技术(或称为内存管理技术) 是不受研究人员关注的一个问题,专用于实时数据库系统的缓冲区管理算法也不多见。许多研究人员认为实时数据库应当或者必须是内存数据库,因此实时数据库缓冲区管理算法的研究没有意义。对于这种认识,文献[1] 进行了否定,阐述了实时数据库和内存数据库在概念、功能以及使用的调度算法上的区别;而在诸如过程生产监控、股票交易等实际应用中,实时数据库需要处理海量数据,这些系统每天产生的数据以 GB 计,将数据全部存储在内存中是不可能的,对于此类应用,将缓冲区管理算法引入实时数据库系统是非常必要的。

实时数据库应用于对数据刷新和处理要求时限性很强的系统,其事务和数据与一般数据库的事务和数据相比,一个最大的不同是具有时态约束。事务的时态约束主要指截止期约束,数据的时态约束是数据的有效时间约束。由于这些特点,用于关系数据库的调度方法,例如事务调度、I/O 调度以及缓冲区调度都不能直接用于实时数据库系统中,因为传统的调度方法很少考虑截止期[2]。尽管在实时数据库中,事务调度和并发控制策略对于系统性能影响更大一些[3],但是不论系统的瓶颈是 CPU 还是磁盘 I/O,对重要资源的调度必须由基于优先级的缓冲区调度算法配合完成,许多实验结果也证明了这一结论[4],因此缓冲区管理是影响实时数据库系统性能的因素之一,应针对实时数据库的特点进行设计。

实时数据库越来越多地用于开放式的环境下,其负载变化比较大,这要求实时数据库的调度方法必须能够适应变化的负载,保持稳定的性能。用于关系数据库的缓冲区管理方法一般都使用固定长度的缓冲区,对负载变化的适应性比较差。

我们通过对实时数据库事务和数据特征的分析,总结了以往的研究成果,在本文中提出了实时数据库缓冲区管理算法的设计原则,并依据该原则设计了一种使用反馈控制思想的基于优先级的实时数据库缓冲区管理方法 FCLRU2-dl(Feedback Control based LRU2-dl)。该方法根据系统负载的变化,动态调整缓冲区的规模,在不同负载下,将系统的事务按时完成比率(Success Ratio) 维持在一定的水平,在一定程度上保证系统性能的稳定。

1　相关的研究工作

由于内存访问时间远远低于磁盘访问时间,因此直接从内存获取所需要的数据将大大缩短事务的执行时间。但是数据库的数据量一般很大,无法把全部数据放在内存中,而只能将一部分数据放入内存,这就给事务的执行带来了不确定性。缓冲区管理的作用是把事务将要访问的数据尽量多地换入内存中,当事务访问这些数据时,使执行速度得以提高。缓冲区管理算法是用来决定将哪些数据放在内存以及存放多长时间的一组策略。缓冲区管理算法包括缓冲区分配和缓冲区置换两

收稿日期: 2004-04-10; 修返日期: 2004-06-30
基金项目: 国家“863”计划资助项目(2001AA412020);国家自然科学基金资助项目(60373055)

个过程,其中缓冲区置换策略是传统缓冲区管理技术的核心。传统的置换方法以提高事务对内存数据访问的命中率(Hit Ratio)为目的,一般基于局部性(Locality)原理和事务的倾斜访问(Skewed Access)原理[5,6],通过分析事务对数据的访问模式,针对一个固定长度的缓冲区设计一个置换算法,按照某种策略将内存数据与磁盘数据进行交换,以期在内存中保存事务将要访问的数据,降低事务执行时的缺页率(Page Fault Ratio),减少事务对磁盘的访问[5~9]。但是多数文献提出的缓冲区管理方法都没有使用优先级策略,同时也没有考虑实时环境。还有的文献提出了一些使用优先级策略的方法如 PLRU(Priority-LRU),PH(Priority Hints)等,使用的都是静态的优先级,而实时环境下的事务的优先级是随着截止期的变化而变化的。

文献[3,8]研究了实时环境下的缓冲区调度问题,并且都使用了基于优先级的调度策略。文献[3]提出的缓冲区管理算法是 LRU-dl(具有截止期的 LRU 算法),并且将该算法用于多种 CPU 调度算法、并发控制算法协同运行,形成了一些实验结果。该文同时探讨了实时数据库中所有的调度策略对于系统性能的影响,但是没有使用先进的缓冲区管理方法。文献[8]针对实时数据库事务特点提出了一种使用预取(Prefetching)策略的缓冲区管理算法 PAPER,取得了较好的实验效果。但是该算法仍然没有脱离传统方法的思维模式,使用固定长度的缓冲区应对变化的负载,影响了算法的稳定性,其预取策略基于对事务、数据、约束三者关系的分析比较复杂,实现困难,也带来较大的系统开销。

在实时事务调度和并发控制策略的研究方面,目前已经有许多研究成果。文献[10]提出了 2PL-HP 的冲突解决方法,是一种目前被广泛使用的并发控制策略,该方法偏向高优先级事务。其基本思想是:如果一个事务请求锁住某个数据对象,而这个数据对象已经被一个或多个低优先级事务锁住,则持有锁的事务被夭折;如果锁请求者的优先级低于锁持有者的优先级,则锁请求者等待。该算法的优点是避免了优先级反转和死锁问题,缺点是可能导致无效的重启。文献[11]最先综合研究了 FCFS(First Come First Serve),EDF(Earliest Deadline First),LSF(Least Slack First)三种优先级调度算法,认为就调度算法而言,EDF 表现了最好的性能。

通过对于实时数据库特征及其他相关研究成果的总结,我们认为实时数据库缓冲区算法设计应当遵循以下原则:

(1)应以事务的按时完成比率作为衡量缓冲区管理算法优劣的标准,好的算法应能提高按时完成比率。

(2)实时数据库的缓冲区管理算法必须考虑事务的信息,包括截止期和事务价值等因素,或者说是能够识别事务截止期和价值的调度算法。

(3)实时数据库的缓冲区管理算法要考虑系统在变化负载下的稳定性和效率问题。

(4)实时数据库的缓冲区管理算法需要为用户提供控制接口,除了允许用户改变缓冲区的长度、结构等信息外,还应当允许用户指定重要的、使用频繁的表和数据常驻缓冲区,使其在生存期间不被换出[12]。

2 FCLRU2-dl 算法

针对上述的实时数据库的缓冲区管理的要求,我们提出了一个新方法,使用反馈控制策略对缓冲区长度进行调整,维持系统在不同负载下性能的稳定性。

图 1 中,虚线右侧为 FCLRU2-dl 算法的组成模块,虚线左侧为实时数据库系统的功能模块。算法使用的所有缓冲区页面都位于全局缓冲区 Global Buffer 中;置换控制器(Replacement Controller)在系统稳态运行期间工作,(这里的稳态是指系统在某个固定负载条件下运行了足够的时间后达到的状态),置换控制器将使用 LRU2-dl 方法决定哪一页换出,并将这个信息发送给执行器(Executor)执行;反馈控制器(Feedback Controller)在系统负载发生较大变化时工作,用来判断缓冲区长度是否符合负载的要求,当缓冲区长度不适合当前的负载时,反馈控制器根据一定的方法计算一个新的缓冲区尺寸,将这个决定也交给执行器执行;事务管理器(Transaction Manager)计算当前的负载并将负载情况通知反馈控制器。执行器负责执行分配/释放/交换缓冲区等所有实际的操作行为。执行器需要通过 I/O 管理器的协助完成最终的读写磁盘的操作。

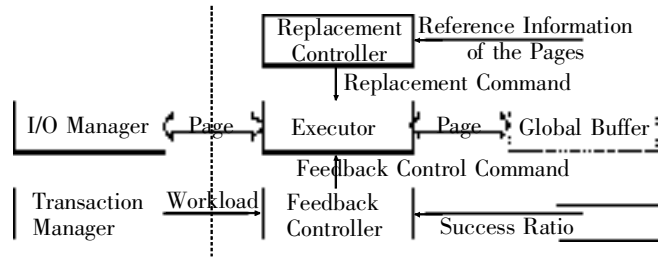

图 1 FCLRU2-dl 算法的基本结构

2.1 缓冲区页面的结构

大量的实验结果证明,事务对数据的访问满足局部性(Locality)原理,即一个数据被访问,其相邻的数据也将很快被访问。在实时数据库系统中,该规则依然成立。实时数据库中事务访问数据的另一个特点是时态倾斜,即多数事务只访问最近某个时间段内的数据,一个数据的时间标签距离当前时间越远,其被访问的概率也越小。基于这些特点,我们为实时数据库的缓冲区设计了如表 1 所示的结构。

表 1 缓冲区页面结构

	字段 1	字段 2		字段 m
时刻 1	$Value_{11}$			
时刻 2				
…			$Value_{ij}$	
时刻 n				$Value_{mn}$

$Value_{ij}$是字段 i 在 j 时刻的值,可能是一个结构型数据,包括了字段的值和其他需要的信息。一个缓冲区中的包含的字段可以是全部或部分数据库字段,时刻 1 ~n 时刻是一组连续的时间标签。参数 m 和 n 的选择与缓冲区长度、数据库的规模以及具体的应用有关。该结构可以很好地利用局部性原理和时态倾斜访问原理,提高数据访问的命中率。

缓冲区页面的控制信息包括如该页面是否被引用以及引用时间,该页面是否是“脏”页,引用该页面事务的部分信息如截止期、价值等也包含在缓冲区的页面中。页面的优先级与引用页面的事务的截止期成反比。

2.2 缓冲区的置换

当系统的事务到达速率在一定时间内基本维持恒定时,可以认为系统负载处于稳态状态,此时使用缓冲区置换策略处理事务访问时的缺页。我们使用 LRU2-dl 的置换策略。许多实验证明文献[13]提出的 LRUK 算法优于 LRU 算法(命中率最大可以提高 40%)。LRUK 算法记录每个页面最近第 K 次被引用的时间,取该值最小(离当前时间最久)的一页换出。文

献[13]同时证明了 LRUK 算法取 $K>2$ 时的效果仅仅略优于 $K=2$ 的效果,所以我们此处取 $K=2$,即使用 LRU2 算法。我们对 LRU2 增加了识别截止期的机制,建立了 LRU2-dl 算法。算法中使用的页面信息包括: Pi. dl,页面的截止期,是引用该页的事务的(如果多个事务取最短)截止期; Pi. usr,引用该页面的事务数; Pi. ts,该页的次最近(从当前时间向前推的第二次引用)访问时间。算法描述如下:

① search a proper page from the start location to the end to replace one page;
② if(Pi. dl < now) or (Pi. usr = 0)
then replace the page;
return success;
③ if (step ② fails)
then find a page with the least Pi. ts; replace it;
return success;
④ move the start location to the page which next to the replacement location

2.3　缓冲区的反馈控制

当负载发生较大变化时,系统将根据一定的规则增加(减少)缓冲区的长度,维持 Success Ratio 稳定于某个目标值。系统根据负载变化的情况确定反馈策略,增加或减少缓冲区的长度,当缓冲区的长度等于数据库的长度时,实时数据库完全位于内存中,成为一种内存数据库。此时系统表现了较好的性能,我们把系统在这种情况下的 Success Ratio 值作为反馈控制算法的目标值,对系统的缓冲区规模进行反馈控制。

一个典型的反馈系统如图 2 所示。

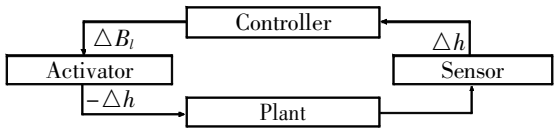

图 2　反馈控制系统

Plant 是被控制对象,有若干个变量反映它的状态,在该算法中被控制对象是 Plant 的一个属性——Success Ratio,控制的手段是调控 Plant 的缓冲区长度 B_l,从 Plant 中采集的状态是 S_r 和 MAR(Mean Arrival Rate,事务平均到达速率); Sensor 采集 Plant 的状态,计算 Success Ratio 状态的变化率和 MAR 的变化率△MAR,传送给 Controller, Controller 使用一个控制算法计算出系统需要改变的缓冲区数量的比率△B_l,将这个决策交给 Activator 执行,使 Plant 的 S_r 向目标的方向变化。反馈控制的算法如下:

① If(there is a △MAR for a time t)
② If (MAR is low level) and ($|S_r-S_{rl}|>\triangle S$) then $B_l=B_{ll}$
③ If (MAR is medium level) and ($|S_r-S_{rm}|>\triangle S$) then $B_l=B_{lm}$
④ If (MAR is high level) and ($|S_r-S_{rh}|>\triangle S$) then $B_l=B_{lh}$

t 是稳定时间,△S 是允许的调整误差。B_{ll}是低负载时的缓冲区长度的理想值,B_{lm}是中负载时的理想值,B_{lh}是高负载时的理想值,是由实验得到的。S_{rl}, S_{rm}, S_{rh}分别是系统在低、中、高负载下的 Success Ratio 的目标值,上述已经提到,该值是实时数据库全部位于内存时的 Success Ratio 测量值。

3　算法测试

我们的测试平台是集中式的实时数据库系统,包括了实时数据库系统中所有重要的功能模块,例如事务调度、并发控制、I/O 管理等功能。我们在实验中使用的事务调度算法是 EDF,使用的并发控制协议是 2PL-HP, I/O 调度是基于优先级的非抢占式调度,这里优先级和页面的截止期成反比。其他重要的实验参数在表 2 中详细的列出。

3.1　实验环境

我们进行算法测试的实验环境如表 2 所示。

表 2　实验参数

数据库规模	1 000 Blocks
缓冲区长度	0～1 000 Blocks
测试指标	Success RatioTotal Hit Ratio (事务在内存中得到的页的和事务访问的总页面数的比率)
事务调度算法	EDF
并发控制策略	2PL-HP
数据访问模式	倾斜访问(80～20 Rule)
事务到达模式	均值为 m 的独立同分布的指数分布
I/O 操作时间	均值为 k 的独立同分布的指数分布, k = 30 个时间单位
I/O 调度策略	优先级无抢占
每个事务访问的页面	3～15, Mean Operation = 9
每个页面的处理时间(Calculate Time)	34 个时间单位
更新时间	6 个时间单位
加锁/解锁时间(Lock Time/Unlock Time)	1 个时间单位
P_W(写操作概率)	0.5
松弛率(Slack_Ratio)	3～4
截止期计算公式	Mean Operation ×(Lock Time + Unlock Time + Calculate Time) ×(Slack_Ratio + 1)
低负载	2 事务/s, CPU 利用率 = 0.7
中负载	4 事务/s, CPU 利用率 = 1.4
高负载	6 事务/s, CPU 利用率 = 2.1

实验的参数主要参照文献[10, 11]中的实验参数。其中缓冲区的长度是缓冲区管理而言的一个重要参数。我们研究了不同缓冲区规模下系统的性能。

3.2　实验曲线

我们测量了在不同的缓冲区长度和不同负载的条件下,基于反馈控制的 LRU2-dl 的性能指标,包括 Hit Ratio 和 Success Ratio,形成了本节的各实验曲线。图 3、图 4 是三种负载情况下的命中率曲线。

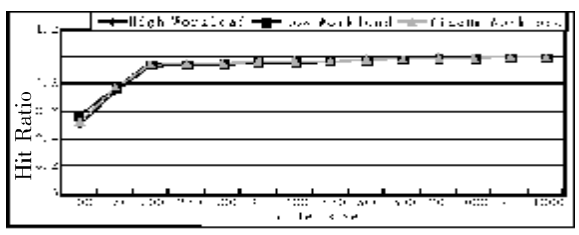

图 3　不同负载下不同缓冲区长度的 Hit Ratio

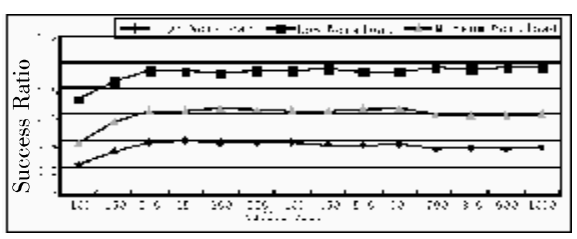

图 4　不同负载下的 Success Ratio

3.2.1　低负载时

如图 3、图 4 所示,当缓冲区长度小于 200 时, Hit Ratio(H 曲线)和 Success Ratio(S 曲线)随着缓冲区长度的增加有比较快的增长,这是由于缓冲区长度较小时 Hit Ratio 小于缓冲区长度较大时的 Hit Ratio,因此小缓冲区会引起比较多的 I/O 操作,事务也因此花费更多的时间获取数据,增加了事务错失截止期的概率。当逐渐增加缓冲区规模时, I/O 操作将减少,使事务可以在更短的时间内获取数据,因此系统的 Success Ratio 将随着缓冲区的增加而增加。

当缓冲区长度大于 200 时, H 曲线和 S 曲线随着缓冲区的增加只有边际的提高。这是因为数据的访问模式是倾斜访问,

遵循 80-20 规则,即 80% 的事务只集中访问数据库中特定区域(一般只占数据库规模的 20%)的数据。

当缓冲区大于 700 时,由于 Hit Ratio 已经足够大,几乎消除了 I/O 操作对于事务的影响,因此 H 曲线和 S 曲线都不再随缓冲区的增加而上升。

3.2.2 中负载时

当缓冲区长度小于 600 时,H 曲线的趋势和低负载情况下的一样,其原因也相同。

当缓冲区大于 600 时,虽然 H 曲线随着缓冲区的增加而增加,但是 S 曲线却开始下降。这个反常的现象可以用并发事务对于数据的争夺来解释。在中负载条件下,当缓冲区长度大于 700 时,几乎没有事务是因为 I/O 操作而超时或夭折的,由于负载比较高,因此并发事务之间的数据争夺就比较严重,由于一些被低优先级事务持有的锁又被高优先级事务请求,引起了持有锁的低优先级事务的夭折。但是在缓冲区较小时,例如 200 ~600,这些事务可能是被 I/O 操作阻塞的,它们就不会被高优先级的事务锁夭折,因此有机会执行完毕,这种情况下系统的 Success Ratio 也比较高;同时,由于这些事务的阻塞,系统中的数据争夺现象得以缓解,其他的事务有机会得到它们的数据并且在截止期内完成。因此在中负载情况下,缓冲区较小时的 Success Ratio 可能高于缓冲区较大时的 Success Ratio。

3.2.3 高负载时

高负载时的 H 曲线趋势和中、低负载条件下的曲线是一样的,其原因也相同。而高负载下的 S 曲线的趋势与中负载条件下的曲线相似,只是在不同缓冲区规模下,Success Ratio 的变化更加明显一些。

图 3 显示出:不论系统的负载情况如何,H 曲线基本是一样的,这也源于我们使用 80-20 的规则模拟数据访问。而图 4 显示了不同负载条件下,低负载时系统具有最高的 Success Ratio,而高负载时的 Success Ratio 最低。同时在中、高负载条件下,系统的 Success Ratio 以缓冲区规模在 200 ~600 时为最高。

根据上述的实验结果,我们设定了 FCLRU2-dl 算法的各项参数,包括 B_{ll}, B_{lm}, B_{lh} 和 S_{rl}, S_{rm}, S_{rh},并且对该算法进行了测试。此处 B_{ll} 为 700, B_{lm} 为 600, B_{lh} 为 200。

我们使用了无反馈的 LRU2-dl 算法与 FCLRU2-dl 算法进行比较,基于反馈控制的 LRU2-dl 使用变化的缓冲区长度,低负载下缓冲区长度为 700,中负载下为 600,高负载下为 200;而无反馈的 LRU2-dl 算法则使用固定的缓冲区长度即 1 000,即数据库全部位于内存中。两个算法的运行结果曲线如图 5 所示。

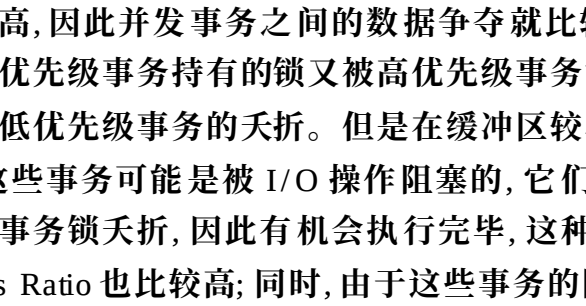

图 5 基于反馈控制的 LRU2-dl 和无反馈控制的 LRU2-dl 的 Success Ratio 曲线比较

从图 5 中可以看到,FCLRU2-dl 算法在满足系统需求和缓冲区使用效率上都优于 LRU2-dl:使用了更少的缓冲区,获得了更优的 Success Ratio。

3.3 结果讨论

根据实验结果,可得出以下结论:

(1) 由于 I/O 调度对于事务调度和并发控制的影响,在使用一定的事务调度算法和并发控制算法时,实时数据库不需要是全部位于内存的,实时数据库可以更经济的方式运行。当然该结论与系统使用的并发控制算法和事务调度算法有关。

(2) 在低负载下,Hit Ratio 的提高总是能够带来 Success Ratio 的提高,但是当 Hit Ratio 达到一定水平时,Hit Ratio 带来的效益就很小了。在中负载和高负载下,提高 Hit Ratio 不一定能提高 Success Ratio,因此实时数据库的缓冲区管理必须直接引入 Success Ratio 作为指标。

4 结论

本文提出了一种基于反馈控制的 LRU2-dl 算法,用于实时数据库的缓冲区管理。该算法包括识别截止期的缓冲区管理策略和反馈控制策略。我们将算法与 EDF 算法和 2PL-HP 算法配合进行了模拟实验,结果表明具备了反馈控制的 LRU2-dl 算法的性能优于无反馈 LRU2-dl 算法。

参考文献:

[1] Stankovic, S H Son, J Hansson. Misconceptions about Real-time Databases[J]. IEEE Computer, 1999, 32(6):29-36.

[2] K Ramamritham. Real-time Databases[J]. International Journal of Distributed and Parallel Databases, 1993, 1(2):199-226.

[3] J Huang, J Stankovic Buffer Management in Real-time Databases [R]. COINS TR 90-65, Dept. of Computer and Information Science, Univ. of Massachusetts, 1990.

[4] M J Carey, R Jauhari, M Livny. Priority in DBMS Resource Scheduling[C]. Proceedings of the 15th VLDB Conference, Amsterdam, the Netherlands, 1989. 397-410.

[5] Abraham Silberschatz. 数据库系统概念(第 4 版 影印版)[M]. 北京:高等教育出版社, 2002. 288-290.

[6] W Effelsberg. T Haerder. Principles of Database Buffer Management [J]. ACM Transactions on Database Systems, 1984, 9(4):560-595.

[7] A Dan, D M Dias, P S Yu. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes[C]. ACM SIGMETRICS, Denver, 1990. 143-152.

[8] A Datta, S Mukherjee, I Viguier. PAPER: Prefetching and Priority Cognizant Buffer Replacement Policies in Real-time Active Database Systems[J]. Journal of Systems and Software, 1998, 42: 227-246.

[9] Anthony K, H Tung, Y C Tay, *et al*. BROOM: Buffer Replacement Using On-line Optimization by Mining[C]. Proceedings of the 17th International Conference on Information and Knowledge Management, 1998. 185-192.

[10] J R Haritsa, M J Carey, M Livny. On Being Optimistic about Real-time Constraints[C]. Proceedings of the 9th ACM Symposium on Principles of Database Systems, Nashville, Tennessee, 1990. 331-343.

[11] Robert Abbott, Hector Garcia-Molina. Scheduling Real-time Transactions: A Performance Evaluation[J]. ACM Transactions on Database Systems, 1992, 17(3):513-560.

[12] 舒良才,刘云生,李国徽. 实时内存数据库的数据管理[N]. 计算机世界, 1999, (40).

[13] E J O Neil, P E O Neil, G Weikum. The LRU-K Page Replacement Algorithm for Database Disk Buffering[C]. Washington D. C., USA: ACM SIGMOD Conf., 1993. 297-306.

作者简介:

国志宏(1976-),男,山东泰安人,硕士,主要研究方向为实时数据库语言;王宏安(1963-),男,江西南昌人,研究员,博士生导师,博士,研究方向为实时数据库技术;王强,博士,研究方向为实时数据库技术。