

一种面向实时数据库存储引擎 的设计与实现

刘 波, 范士明, 刘 华

(北京卫星信息工程研究所, 北京 100086)

摘 要: 卫星地面设备监控中, 需研究实时数据库技术来管理大量的分布式环境下的实时数据。为此文章设计了一种实时数据存储引擎。该引擎借助内存文件映射机制实现内存表, 以 LRU 算法和页面缓冲池实现了磁盘表, 通过位图方法管理数据库表的文件空间, 以影子页面法实现了事务的提交与回滚, 并设计了数据库表的逻辑结构。通过使内存表和磁盘表结构保持一致, 确保了数据库运行时内存表和磁盘表的灵活转换, 影子页面法确保了数据库具有实时的故障恢复速度。结果表明, 该引擎适用于存储卫星地面设备监控中的大量实时数据。

关键词: 实时数据库; 存储引擎; 内存文件映射; LRU 算法; 页面缓冲池; 影子页面法

中图分类号: TP31

文献标识码: A

文章编号: 1673-629X(2011)08-0034-05

Design and Implementation of Storage Engine for Real-Time Database

LIU Bo, FAN Shi-ming, LIU Hua

(Beijing Institute of Satellite Information Engineering, Beijing 100086, China)

Abstract: It is necessary to research real-time database to manage massive distributed real-time data in satellite ground device monitoring. In order to resolve the problem of storage of massive real-time data effectively, proposed one storage engine for real-time database. This engine accesses memory-resident table through memory map file mechanism. The converting of file address and linear address mutually is realized by page pool and LRU algorithm while accessing disk-resident table. The table's file space is managed by bitmap method and shadow page method is also introduced in designing of database table structure, based on this foundation, transaction committing and rollback is achieved. This engine can convert one table from disk-resident to memory-resident and vice versa when database is running. This storage engines also ensures the real-time database has the potential of real-time recovery when confronting table failure. The result shows that this engine is suitable to storage massive real time data in satellite ground device monitoring.

Key words: real-time database; storage engine; memory map file; LRU algorithm; page buffer pool; shadow page method

0 引 言

卫星地面设备监控系统需要对大量数据信息进行采集、传输、综合分析、计算等处理。从监控系统组成可以看出, 数据是联系各功能模块的纽带。随着卫星地面应用系统的发展, 地面设备监控系统的功能需求也不断增多、增强, 数据量也不断扩大, 数据之间的关系也越来越复杂。因此需要将数据库技术引入卫星地面设备监控, 用数据库技术来管理、处理监控过程中的数据。但卫星地面设备监控中数据的一个显著特点是

具有时间特性, 且有效时间是短暂的, 过时则失效。而以关系数据库为代表的传统数据库的设计目标是维护数据的正确性、保证系统的低代价和提供友好的用户接口。这种数据库系统对传统的商务和事务型应用是有效、成功的, 但对于新领域的实时数据和实时事务的应用要求难以胜任。所以, 需要结合数据库技术和实时技术, 研究具有显式定时限制的实时数据库系统^[1]。

数据库存储引擎直接影响到实时数据库的稳定性、可靠性、实时性。卫星地面设备监控中, 实时数据库面临大量实时数据, 最新的实时数据必须能在一定时间内存入数据库, 相对陈旧的历史数据在给定关键字时必须能快速检索到相应的值^[2]。传统的数据库存储引擎一般针对离散的业务型数据, 没考虑到卫星地面设备监控中实时数据特点和检索要求, 导致采用传统数据库存储引擎的实时数据库性能低下, 不能满足

收稿日期: 2010-11-24; 修回日期: 2011-03-04

基金项目: 中国空间技术研究院 cast 创新基金(201008)

作者简介: 刘 波(1983-), 男, 河南信阳人, 博士生, 研究方向为分布式实时数据库技术; 范士明, 研究员, 研究方向为卫星遥感信息处理技术。

要求。对此文中根据地面设备监控中实时数据的特点和检索要求,采用内存映射文件实现内存表,在访问磁盘表时以页面缓冲池和 LRU 算法实现进程线性地址与文件地址的转换,并能实现表存储类型的动态互换。在数据库表结构设计中引入了影子页面法,相比基于日志的故障恢复方法,该方法能快速恢复故障表。

1 文件地址与线性地址的转换

在实时数据库中必须实现磁盘表和内存表,磁盘表中大部分驻留在磁盘,通过开辟一片内存缓冲区来访问。内存表将整个表加载进内存,避免了大量的磁盘 IO 操作,支持数据库的实时事务处理。

1.1 内存表的文件地址与线性地址转换

内存表的文件地址与线性地址转换采用图 1 左边所示模型,即整个表驻留在进程虚拟地址空间里,由操作系统的内存文件映射机制实现地址转换。实时数据库中的大量实时事务要求系统能快速处理,但如果对文件的读写访问采用普通的文件读写操作,那么由此带来的磁盘存取、内外存的数据传递、缓冲区管理、排队等待以及锁的延迟会使得事务执行时间慢且无法预料。而内存映射文件在进程的虚拟地址空间中保留一段区域,将来自磁盘上的数据库文件提交给该区域,将文件映射到这段虚拟内存之中。这时内存文件映射的物理存储器来自一个已经存在于磁盘上的文件,而非系统的页文件,可以用存取内存数据的方式直接操作文件中的数据,就像整个文件已经被加载到内存中一样。因此使用内存映射文件处理磁盘文件时,不必再为文件申请并分配缓存,所有的文件缓存操作均由操作系统直接管理。

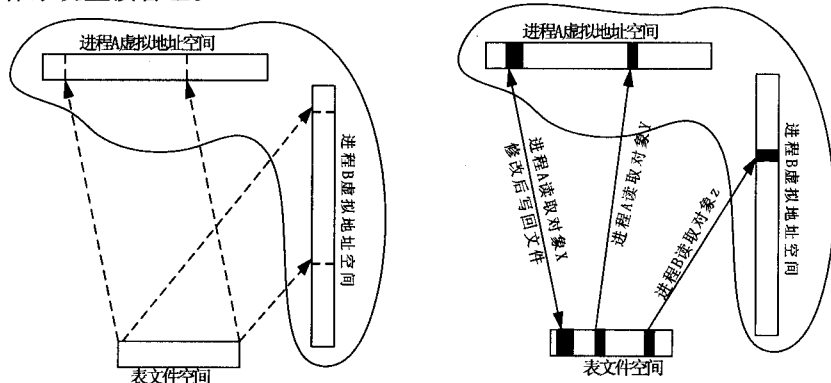


图1 内存表和磁盘表的文件地址与线性地址转换

1.2 磁盘表的文件地址与线性地址转换

当进程访问磁盘表时,实时数据库通过缓冲区操作文件中的数据,缓冲区是进程虚拟地址空间的一部分,对磁盘表的操作必须实现文件地址和进程线性地址的转换,转换方式采用图 1 右边所示模型。缓冲区由 n 个页面组成,页面大小可配置为 4kB 的整数倍。

考虑磁盘数据传输特点,文中磁盘表文件空间和进程虚拟地址空间数据交换单位是一页面。访问文件对象时首先将该文件对象所在磁盘页面复制到缓冲区中,即建立磁盘页面的内存映像。管理缓冲区的数据结构由单链表、一双链表和一哈希表组成。初始时单链表由 n 个 PageHeader 组成,双链表由一个 PageHeader 组成,引入哈希表的目的是能快速查询磁盘页面是否驻留在缓冲区中。当建立一磁盘页面的内存映像时,哈希表中必须存储磁盘页面地址与分配给该磁盘页面的缓冲区页面编号间的映射关系。当将一磁盘页面从缓冲区中置换出去时,必须删除哈希表中该磁盘页面文件地址与分配给该磁盘页面的缓冲区页面编号间的映射关系^[3,4]。

如图 2 左边所示,PageHeader 中的数据库文件地址指使用缓冲区页面的磁盘页面文件地址;访问量指当前使用该缓冲区页面的线程个数;状态有 raw、wait 和 dirty 三种或它们的可能组合,其中 raw 状态表示页面正被一个读取磁盘页面操作锁住,wait 状态表示有线程正等待页面的磁盘读取操作完成,dirty 表示该页面内容被修改,即磁盘页面和其内存映像内容不一致,如果磁盘页面被换出,其修改必须写回磁盘;next 指针指向链表中下一个 PageHeader;prev 指针指向链表中上一个 PageHeader,由 FreePage 指向的双链表中没有使用 prev 指针,它事实上是单链表。初始时 FreePage 为 1,单链表中有 n 个空闲页面可供分配。 P_0 代表一空闲内存页面,初始时 P_0 为首的双链表只包含一个 PageHeader,表示已装载了磁盘映像但访问量为 0 的缓冲区页面个数为 0。

1) 磁盘页面已建立内存映像时的地址转换。

判断缓冲区页面此时访问量是否为 0。

(1) 如果访问量为 0,表明该页面目前没被任何线程使用,将其从 P_0 为首的双链表中移除。

(2) 如果此时访问量大于 0,且页面处于 raw 状态,表明当前有线程正装载磁盘映像到该页面中,线程必须等待,此时添加 wait 到该页面状态位中,并休眠直到其它线程唤醒它。

(3) 如果对磁盘表的文件地址是写访问,需添加 dirty 状态位。

(4) 上述操作完成后缓冲区页面的访问量加一,返回缓冲区页面的首地址。

2) 磁盘页面无内存映像时的地址转换。

通过哈希表没有找到磁盘页面的内存映像时,需

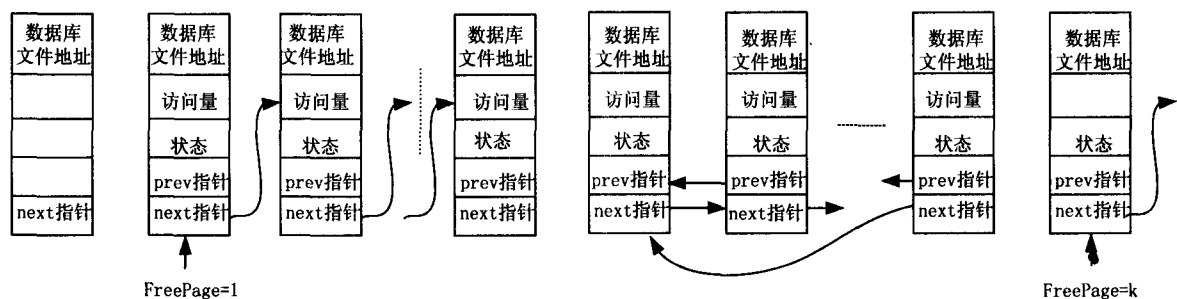


图 2 缓冲区的链表示意图

在 FreePage 单链表和 P_0 为首的双链表中寻找可用的缓冲区页面。

(1) 如果 FreePage 链表不为空, 可将 FreePage 所指缓冲区页面分配给磁盘页面。FreePage 单链表中的页面都是空白页面, 没被任何线程占用过, 如图 2 右边所示, 可将 P_k 作为工作页面。

(2) 如果 FreePage 单链表为空, 将最先加入 P_0 双链表的页面作为工作页面。设该页面为 P_j , 页面状态为 dirty 表明 P_j 与其磁盘映像不一致, 需将 P_j 写回磁盘以保持一致性。写回磁盘前, 检查 P_j 磁盘文件地址是否大于表文件长度。如磁盘文件地址大于表文件长度, 则设置表文件长度 = 磁盘文件地址 + 缓冲区页面大小。然后将 P_j 从 P_0 为首的双链表中移除, 同时移除哈希表中 P_j 磁盘文件地址与缓冲区页面的映射关系。

(3) 当获得缓冲区中一可用页面时:

(a) 首先将其访问量置为 1, 在哈希表中建立磁盘页面地址与缓冲区页面的映射, 并根据线程是否写访问而设置页面的 dirty 状态位;

(b) 然后判断磁盘文件地址 addr 是否小于表文件长度, 如果小于, 则需读取磁盘页面内容到缓冲区页面中, 此时设置页面状态为 raw;

(c) 读取磁盘操作完成后, 判断是否有 wait 状态, 如果有则唤醒等待线程, 并清除页面的 raw 状态和 wait 状态。

3) 页面的释放操作。

释放磁盘页面占用的缓冲区页面只需将缓冲区页面的访问量减一, 减一后如果缓冲区页面访问量为 0, 将该页面作为 P_0 的下一个 PageHeader 加入到双链表中, 因此文中的磁盘页面置换算法为最近最少使用页面置换 (LRU) 算法。

2 数据库表结构的设计

2.1 文件空间的分配和影子页面法

数据库系统需对文件进行大量写和删除操作, 因此必须以一种合理的方式控制写操作发生的位置, 否

则将产生大量不可利用的文件碎片, 从而降低数据库文件的存取速度和存储密度。文中使用位图方法实现文件存储空间的分配, 根据配置大小设定位图页面个数, 每个位图页面为 4kB, 因此共有 32k 比特位, 每个比特位可分配 8 字节 (也可以不是 8 字节, 可根据系统配置而定)。如果该位为 0, 表示该比特位代表的相应位置的 8 字节尚未分配, 否则相应位置的 8 字节已分配出去, 所以一个位图页面可分配的空间为 256k 字节。当一个新的空间分配请求到达时, 必须找到起始位图页面, 并将相应的比特位置 1。

文中采用影子页面法^[5,6]用于数据库的实时故障恢复, 为此每个记录, 无论是索引项、常规记录还是位图页面记录, 系统都必须为其分配一个对象标识符 (oid)。每个记录维持两个版本, 一个是为数据库故障恢复而保留的备份版本, 另一个是为事务处理而保留的工作版本。如图 3 所示, 初始时表文件的首部是表的头文件信息, 描述表格每一列的属性、表格的索引信息以及表格名称等。头文件信息后存储有表格的每个记录的地址, 地址有两种, 一种是备份地址, 指向记录的备份版本, 一种是工作地址, 指向记录的工作版本。

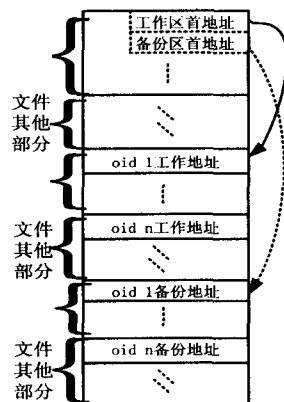


图 3 初始时数据库表结构图

2.2 数据库表逻辑结构的建立

当前数据库中典型的物理存储模型有两种: 行存储和列存储。更新密集型的在线事务处理应用适合采用行存储模型, 而查询密集型的在线分析处理应用适合采用列存储模型, 文中的实时数据库负责大量实时数据的存储和实时计算, 因此采用行存储物理模型, 即

Size	Prev	Next	属性值 1	...	属性值 n	非固定属性列数据 1	...	非固定属性列数据 m
------	------	------	-------	-----	-------	------------	-----	------------

图4 记录存储

一条记录的所有属性列在文件地址上是线性连续的,记录存储如图4所示。

其中 Size 表示该记录的大小,Prev 是表中上一条记录的 oid,Next 是表中下一条记录的 oid,接下来是每个属性列的值:(1)如果属性列的大小是固定的(整型、浮点型变量等),属性列中存放的是实际值。(2)如果属性列的大小不是固定的(字符串、数组类型等),属性列中存放由 offs 和 size 组成的项,实际值存放在[记录首地址+offs,记录首地址+offs+size)这片空间内。采用这种模型是因为:(1)在表中插入或删除某条记录等价于在双链表中插入或删除记录,这样就可以双链表的方式建立了数据库表的逻辑结构。(2)每条记录都被分配一个唯一的对象标识符(oid),根据 oid 获知记录的工作地址或备份地址,结合记录头部的 size,可以读取完整的一条记录。

3 事务的提交和表的故障恢复

3.1 事务处理与影子页面法

事务开始时数据库表处于一致性状态,任何记录都只有一个版本,即备份地址和工作地址相同,因此对象标识符备份区和工作区的内容以及对象标识符个数均相同。当事务修改非位图页面记录时,判断记录工作地址是否置修改位。如果有修改,删除工作地址所指的工作版本,如果没有则说明没有额外的工作版本,不必删除。处理完后为记录分配新的存储空间,将新分配空间起始地址作为记录工作地址,并设置修改位,记录备份地址保持不变且所指记录内容不变。

任何对象都通过在位图页面中的写1或清0来分配或释放空间,这必然修改位图页面。当修改位图页面记录时,判断记录工作地址是否置修改位。如果置修改位,表明位图页面已经有影子页面,不必再为其复制一个影子页面。否则位图页面需要一个影子页面,为该位图页面记录分配一个影子页面。分配影子页面可能会引起位图页面的一连串复制操作,但由于分配的局部性原理,时刻相近的分配操作总发生在连续的位图页面上,位图页面复制操作不会生成过多影子页面。

当事务操作中插入一条记录时,必须为该记录分配一对象标识符。如果 freeoid 的单链表为空,需扩充工作区以存储更多的自由 oid,第一次扩充工作区后数据库表的结构如图5所示。当事务删除记录时,记录备份地址和备份版本均

保持不变。因此事务删除记录时并没有真正删除记录备份版本,它只将记录工作地址置无效。但记录删除后需回收被删除记录在工作区中的对象标识符,freeoid 单链表存储刚回收的标识符。当 oid 回收后该 oid 在某时刻又分配给了另一条记录,这时同一个 oid 记录的备份版本和工作版本可能不相关,但这不影响程序的正确性,因为:(1)事务成功提交时,该 oid 只保留一个工作版本,其备份版本被删除,因为事务的本来目的就是要删除原先记录。(2)当事务提交失败,而回退到备份版本时,该记录也只保留一个备份版本,等于事务删除记录的操作被撤销了。

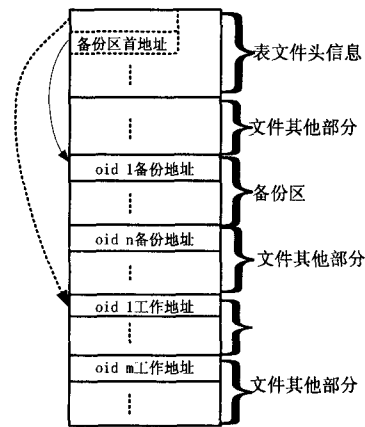


图5 扩充工作区后数据库表结构图

3.2 事务提交以及表的故障恢复

事务处理结束后,需提交它对表做出的修改。由于采用影子页面法^[7],删除和修改记录操作都没有删除记录的备份版本,因此事务提交前必须肃清备份版本。工作区的对象标识符个数只有大于或等于备份区对象标识符个数两种情况,因此备份区的每个对象标识符必定有工作地址。如图6所示,备份地址和工作地址相等表明事务处理过程中没有分配额外空间,不需删除备份版本。备份地址无效表明事务开始前,记录没有备份版本,也就不必删除。备份地址有效表明备份版本必须被删除。经过上述过程处理后,备份版

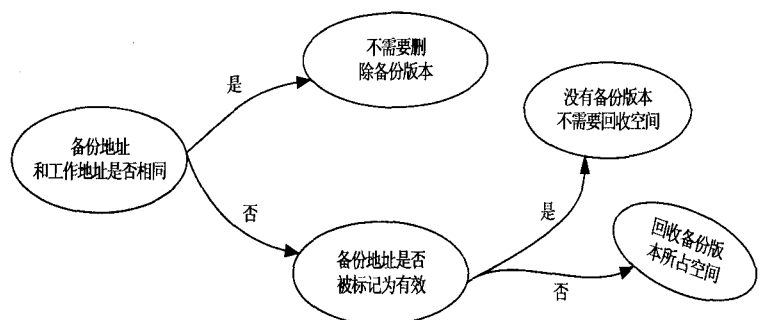


图6 删除记录备份版本示意图

本占用的空间被回收。当空间回收后,确保备份区和工作区大小相同且每个对象标识符的备份地址和工作地址相同,就完成了事务的提交。这时数据库表又恢复到一个新的一致性状态,相互交换备份区和工作区的角色后,可以开始新一轮的事务处理。

当事务失败或数据库意外重启时,系统需恢复数据库表。故障恢复等于撤销事务对数据库做出的改变,影子页面法思想是备份版本保持不变,修改总在工作版本上。因此恢复工作只需将备份区的内容拷贝回工作区起始地址即可,这时工作区与备份区内容完全相同。相比基于日志^[8,9]的故障恢复技术,影子页面法消除了日志记录和处理的开销,故障恢复速度快,适于卫星地面设备监控这种实时性要求较高的场合。

4 结束语

对已创建的表,实时数据库系统必须提供将表的存储类型灵活转换的功能,即从内存表到磁盘表或从磁盘表到内存表都能在数据库运行时转换。这是因为在实时应用里,对某些占用大量内存,但近期不太可能访问的内存表,将其转换为磁盘表能为应用程序保留更多内存。而对驻留在磁盘上,但近期需要频繁访问的磁盘表,将其转为内存表,对避免数据库的磁盘 IO 操作,保证实时的并发读写性能也至关重要。为确保数据库运行时,能动态地转换表的存储类型,文中的内存表和磁盘表采用一样的文件结构,创建表格、插入、删除、修改记录和索引的创建查询等操作方式也保持一致。

磁盘表的线性地址与文件地址的转换过程中,线程使用页面完毕后必须立即释放对页面的控制权,对数据库这种典型的高并发系统,缓冲区作为共享资源,任何线程都不能长久占有某页面,否则将导致大量的不可换出的页面驻留在缓冲区内,有时甚至会出现无法为一个磁盘页面分配缓冲区页面的现象,这将为实时数据库的稳定性带来隐患^[10]。

文中的存储引擎只能解决单机环境下的数据存储,尚不能构建一个可靠的分布式实时数据库。Er-

lang 是爱立信为开发电信交换系统而设计的语言,具有分布式、软实时、高并发、高可靠、代码热插拔等特性,适合构建大规模的“永远开启、永不停机”的分布式系统^[11,12],这使得 Erlang 成为一个理想的分布式实时系统的开发语言,下阶段将在存储引擎的基础上用 Erlang 语言实现分布式实时数据库的事务处理机制,以方便管理卫星地面设备监控中的分布式实时数据。

参考文献:

- [1] 刘云生. 现代数据库技术[M]. 北京:国防工业出版社, 2001.
- [2] 吴建强. 流程工业实时数据库研究和开发[D]. 杭州:浙江大学, 2003.
- [3] 钟宝荣, 袁文亮. 内存数据库中空闲页面管理的方法研究[J]. 计算机工程与设计, 2007(7): 37-40.
- [4] 余翔湛, 殷丽华. 动态共享内存缓冲池技术[J]. 哈尔滨工业大学学报, 2004, 36(3): 128-132.
- [5] 刘云生, 何君辉. 一种主动实时数据库的系统内存管理方法[J]. 计算机应用, 2004, 24(4): 24-26.
- [6] 卢春鹏. 一种嵌入式系统的内存分配方案[J]. 单片机与嵌入式系统应用, 2002(12): 12-16.
- [7] Shu L C, Sun H M, Kuo T W. Shadowing-Based crash recovery schemes for real-time database systems[C]//In: Proc. of the 11th Euromicro Conf. on Real-Time Systems. York, England, UK; [s. n.], 1999: 260-267.
- [8] 肖迎元. 分布式实时数据库技术[M]. 北京:科学出版社, 2009.
- [9] Panda B, Tripathy S. Data dependency based logging for defensive information warfare[C]//In: Proc. of the ACM Symp. on Applied Computing. Villa Olmo, Como; [s. n.], 2000: 361-365.
- [10] Gray J, Reuter A. 事务处理:概念与技术(影印版)[M]. 北京:人民邮电出版社, 2009.
- [11] Armstrong J. Making reliable distributed systems in the presence of software errors[D]. Sweden: Royal Institute of Technology, 2003.
- [12] Armstrong J. Programming Erlang Software for a Concurrent World[M]. USA: Pragmatic, 2007.

(上接第 33 页)

- 计算机技术与发展, 2008, 18(5): 51-53.
- [7] 赵伟, 何丕廉, 陈霞. Web 日志挖掘中的数据预处理技术研究[J]. 计算机应用, 2003(5): 62-64.
- [8] Tsay Y, Chiang J. An efficient method for mining association rules[J]. Knowledge-Based Systems, 2005, 18(3): 99-105.
- [9] 郭有强. 一种高效的关联规则维护算法研究与实现[J]. 计算机技术与发展, 2007, 17(10): 123-126.
- [10] Zhao K, Liu B, Tirpak T M, et al. A visual data mining framework for convenient identification of useful knowledge[C]//ICDM '05. [s. l.]: [s. n.], 2005: 530-537.
- [11] Oosthuizen C, Wesson J, Cilliers C. Visual web mining of organizational web sites[C]// Proceedings of the Conference on Information Visualization. [s. l.]: [s. n.], 2006: 395-401.
- [12] Chen Jiyang, Sun Lisheng, Zaia O R, et al. Visualizing and discovering web navigational patterns[C]//Seventh ACM SIGMOD International Workshop on the Web and Databases (WebDB 2004). [s. l.]: [s. n.], 2004: 13-18.