

文章编号:1001-9081(2005)03-0631-03

基于模式更新的实视图一致性维护策略

刘 群,张春海,李 华

(中国海洋大学 计算机科学系,山东 青岛 266071)

(lq2002@mail.ouc.edu.cn)

摘 要:通过定义不同数据源更新之间的并发依赖关系和同源依赖关系,利用概念化事务模型可实现松散耦合环境中视图的并发维护。在此基础上,提出 VM_SCNF 算法,解决不稳定网络环境中基于模式更新的实视图一致性维护问题,并通过实验原型验证了算法有效性。

关键词:实视图维护;模式更新;概念化事务模型;复合更新

中图分类号:TP311.13 **文献标识码:**A

Materialized view maintenance strategy based on schema changes

LIU Qun, ZHANG Chun-hai, LI Hua

(Department of Computer Science, Ocean University of China, Qingdao Shandong 266071, China)

Abstract: The view maintenance process in the loosely coupled environment was implemented as a conceptual transaction, by the definitions of concurrency dependency and same-source dependency. Based on schema changes, the VM_SCNF algorithm was proposed to solve the consistency of view maintenance processes in the non-FIFO network environments. Finally, a preliminary experiment was carried out to verify the efficiency of the proposed algorithm.

Key words: materialized view maintenance; schema changes; conceptual transaction model; complex updates

0 引言

随着基于 Web 的应用的高速发展,异构数据集成在现代信息系统中起到越来越重要的作用。集成后的数据通常保存在数据仓库的实视图中,用于查询和分析。对于丰富结构的数据,通过模式集成和模式映射实现异构数据交换。

在传统环境中,实视图维护问题已得到了充分的研究,补偿算法可以在数据源模式保持不变前提下,利用补偿查询来消除视图维护的不一致^[1]。然而,在松散耦合环境中,数据源可能随时更新数据模式,若模式更新影响了视图定义,补偿算法中的查询必然失败,导致视图维护终止。

本文首先讨论了两类实视图维护异常:数据更新异常和模式更新异常。通过定义不同数据源更新之间的并发依赖关系和同源依赖关系,利用概念化事务模型实现松散耦合环境中实视图的并发维护。在此基础上,提出 VM_SCNF 算法,解决不稳定网络环境中基于模式更新的实视图一致性维护问题。最后,通过已经实现的原型系统验证了该算法的有效性。

1 视图维护

1.1 传统实视图维护过程

传统实视图维护过程包括三部分:VM (View Maintenance), VS (View Synchronization) 和 VA (View Adaption)。VM 解决并发数据更新后视图的维护^[1]。当数据源的模式发生变化时,VS 着眼于如何重写视图定义^[2]。VA 增量地修正视图内容,使其满足重写后的视图定义^[3]。

视图一致性的级别包括:收敛一致性、弱一致性、强一致性和完全一致性^[1]。这四种一致性的程度依次加强。多数情况下,用户要求实视图维护满足强一致性。

1.2 两类实视图维护异常

考虑例 1 中的视图 Paper_infor,它的视图定义由数据源 1

和数据源 2 的模式集成而得(图 1):

```
CREATE VIEW Paper_Info AS
SELECT a.title, a.author, abs, pub, full, citat
FROM Paper_abs a, Paper_full f
WHERE a.title = f.title and a.author = f.author
```

首先讨论两类视图维护异常:数据更新异常和模式更新异常。所谓数据更新异常,即数据源并发提交的数据更新(仅讨论插入和删除)影响先前发生的更新的视图维护过程,导致维护结果错误。对于此种情况,通常采取补偿算法进行结果修正^[1]。本文重点讨论模式更新异常,即导致视图维护终止的数据源并发模式更新。例 1 中,若数据源 1 修改 XML 到关系的映射,去掉 author 属性或把 author 属性重命名为 writer 属性,视图定义必须重写。同时,维护过程中按照原视图定义进行的查询均会宣告失败,导致视图维护终止。

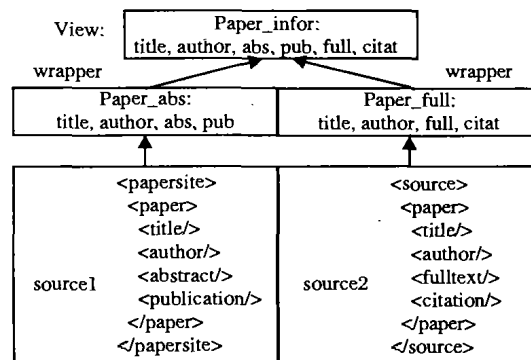


图 1 例 1 的数据源模式以及视图定义描述

1.3 更新的依赖关系

数据源更新的不同类型对应不同的实视图维护过程^[4]。用 DU 表示数据更新,它的维护过程表示为:

$$M(DU) = R(VD) R(DS_1) \cdots R(DS_n) W(V) C(V)$$

收稿日期:2004-08-16;修订日期:2004-10-26

作者简介:刘群(1980-),女,山东青岛人,硕士研究生,主要研究方向:异构数据集成、数据库;张春海(1963-),男,山东青岛人,副教授,主要研究方向:数据库、数据挖掘;李华(1965-),女,山东青岛人,副教授,主要研究方向:数据库。

用 SC 表示模式更新,它的维护过程表示为:

$M(SC) = R(VD) W(VD) R(DS_1) \cdots R(DS_n) W(V) C(V)$, 其中 $R(VD)$ 表示读视图定义; $W(VD)$ 表示根据模式更新重写的视图定义; $R(DS_i)$ 表示对第 i 个数据源进行查询; $W(V)$ 表示增量的修正视图内容; $C(V)$ 表示提交整个视图维护过程。

根据 $M(SC)$ 的表示可知,不同更新的维护过程,对于作为临界资源的视图定义的读写冲突($R(VD)$ 与 $W(VD)$ 冲突)造成模式更新异常。

考虑模式更新的三种情况:属性重命名,丢弃属性和添加属性。对于属性重命名和添加属性,不影响视图内容,只需要修改相应的视图定义。本文讨论的模式更新主要为丢弃属性。

定义 1 并发依赖和同源依赖

$U(DS_i)$, $U(DS_j)$ 分别表示提交给不同数据源 DS_i 和 DS_j 的两个本地更新,相应的视图维护过程记作 $M(U(DS_i))$ 和 $M(U(DS_j))$ 。

$M(U(DS_i))$ 并发依赖于 $M(U(DS_j))$, 记作 $M(U(DS_i)) \xleftarrow{cd} M(U(DS_j))$, 当且仅当 $M(U(DS_i))$ 包括读视图定义操作 $R(VD)$, 而 $M(U(DS_j))$ 包括重写视图定义操作 $W(VD)$ 。

$U_1(DS_i)$, $U_2(DS_i)$ 表示依次提交给同一个数据源 DS_i 的两个本地更新。 $M(U_2(DS_i))$ 同源依赖于 $M(U_1(DS_i))$, 记作 $M(U_2(DS_i)) \xleftarrow{sd} M(U_1(DS_i))$, 当且仅当更新 $U_1(DS_i)$ 在更新 $U_2(DS_i)$ 之前提交。

两个更新 U_1 和 U_2 , 定义 $M(U_2)$ 依赖于 $M(U_1)$, 记作 $M(U_2) \leftarrow M(U_1)$, 当且仅当 $M(U_2)$ 并发依赖或者同源依赖于 $M(U_1)$ 。

若 $M(U_2) \leftarrow M(U_1)$, 同时 $M(U_1) \leftarrow M(U_2)$, 则 $M(U_2)$ 和 $M(U_1)$ 构成循环依赖。

若 $M(U_2)$ 依赖于 $M(U_1)$, 且 $M(U_1)$ 在 $M(U_2)$ 之前完成, 则此依赖关系为安全依赖, 否则为非安全依赖。非安全依赖关系造成视图维护异常。

2 实视图维护概念化事务模型

2.1 锁机制的并发控制策略不适合视图维护

传统的基于锁机制的并发控制策略利用两段锁协议实现:在对任何数据进行读、写操作之前,要申请并获得对该数据的封锁;在释放一个封锁之后,事务不再申请和获得任何其他封锁。

诸如 Web 或 DataGrid 等松散耦合环境中,数据源不允许来自外部的封锁,本地更新必须自治提交。故对于实视图维护,基于锁机制的并发控制策略无法实现。

2.2 实视图维护概念化事务模型

视图维护概念化事务模型 (VM_Trans Model) 由数据源的本地更新和相应的视图维护过程两部分组成。定义概念化全局事务 VM_Trans 为:

$$VM_Trans = T_s T_v =$$

$$U(DS_i) C(DS_i) R(VD) W(VD) W(V) C(V)$$

其中 $U(DS_i)$ 表示数据源 i 的本地更新, $C(DS_i)$ 表示提交本地更新, $W(V)$ 表示增量的修正视图内容, $C(V)$ 表示提交整个概念化全局事务。

全局事务从数据源的本地更新开始启动,当视图维护结束后提交整个事务。若全局事务在执行过程中遭遇异常,因为无法撤销数据源已经提交的本地更新,故只有部分滚回,即重新进行视图维护工作,直至正确提交全局事务。

本文提出的全局事务模型只是封装两种操作的概念化事务模型,并非传统意义上的事务。数据源的本地更新允许自

治提交,每当全局事务遭遇异常,进行滚回操作时,并非全部滚回,而仅重新进行相应的视图维护工作。

3 VM_SCNF 算法

3.1 不稳定网络环境中的视图维护

目前已有的若干实视图维护算法,都是基于 FIFO(先进先出)的假设,即数据源本地更新的顺序与视图管理器接收的顺序一致^[1-5]。若处于不稳定的网络环境中,数据源本地更新顺序与视图管理器接收顺序难以保持一致。根据接收的乱序更新进行视图维护,必将产生视图维护过程之间的非安全依赖关系,导致视图维护异常。

为解决不稳定网络环境中的视图一致性维护问题,我们改进视图管理器中更新消息队列 UMQ 的结构,在队列中同时记录更新的全局标识 (G_id)、数据源标识 (DS_id) 和数据源中本地更新标识 (L_id)。视图管理器同时保存一个缓冲区域 (buffer 区域) 和每个数据源最近更新的局部标识 L_num。数据源提交的更新信息首先存入 buffer 区域,利用 DS_id、L_id 和 L_num 可以确定更新丢失情况以及更新的正确提交顺序。一旦出现视图管理器接收的更新顺序与实际顺序不符,或更新丢失情况,就终止当前视图维护事务,调整更新处理顺序或向数据源发送消息,要求重新提交丢失的更新信息。

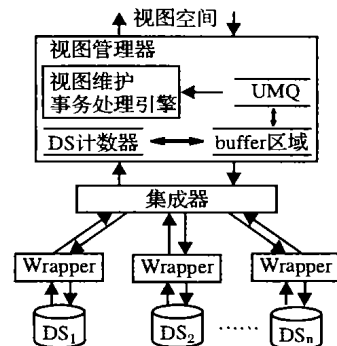


图2 不稳定网络环境中实视图维护事务模型基本框架

3.2 依赖有向图和复合更新

视图一致性维护要求识别不同数据源更新之间的依赖关系。我们首先建立依赖有向图,由此决定视图维护策略。

定义 2 依赖有向图定义为 $G = (V, E)$, 节点集 V 表示 UMQ 中等待维护的正确顺序的更新队列 U_1, \dots, U_n , 有向边 E 表示依赖关系。 $U_i \rightarrow U_j$ 表示视图维护过程 $M(U_j)$ 并发依赖或同源依赖于视图维护过程 $M(U_i)$ 。

在依赖图中,可能出现循环依赖,形成“环”。考虑例 1 中的视图定义:假设数据源 1 发生数据更新 U_1 , 随后数据源 2 发生模式更新 U_2 , 接着数据源 1 又发生模式更新 U_3 。则 $U_2 \rightarrow U_1$, $U_3 \rightarrow U_2$ 和 $U_2 \rightarrow U_3$ 构成并发依赖关系, $U_1 \rightarrow U_3$ 构成同源依赖关系。

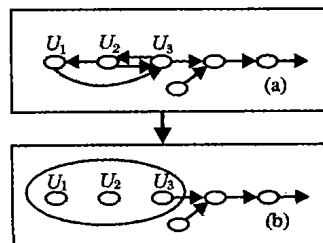


图3 例1的依赖有向图表示

如图 3(a) 所示, $U_1 U_2 U_3$ 形成“环”状循环依赖, 即非安全的依赖关系, 导致死锁的发生。

传统事务处理死锁问题的方法是选择一个进程, 将其撤

销,由此打破“环”。在视图维护过程中,由于无法撤销数据源已经提交的本地更新,故传统方法不可行。在此提出把形成循环依赖的多个更新看作一个复合更新,整体进行处理^[4],如图3(b)所示。这样就消除了依赖关系中的环路,避免了死锁的发生。

3.3 VM_SCNF 算法

算法 VM_SCNF 利用概念化事务模型,以及 buffer 区域、UMQ 和数据源最近更新的局部标识 L_num,解决不稳定网络环境中基于模式更新的实视图一致性维护问题。主要执行步骤如下:

Step 1 检测 buffer 区域中的更新队列:

若出现由于不稳定网络环境造成的本地更新乱序提交的情况,则调整更新维护顺序;若出现本地更新丢失的情况,则向相应数据源发送消息,要求重新提交丢失的更新信息,继续进行 Step1 的检测。通过检测,则进行 Step2。

Step 2 由 buffer 区域向 UMQ 传送正确顺序的更新队列。

Step 3 对 UMQ 中更新队列建立依赖有向图:

1) 若图中无环,则进行拓扑排序,建立偏序关系。所需的时间复杂度为 $O(n+e)$,其中 n 为更新数目, e 为依赖关系数目。

2) 否则,图中有环,首先识别所有的环,所需的时间复杂度为 $O(n+e)$ 。把环内包含的更新看作一个复合更新,具体维护过程见 Step 4。

Step 4 视图维护处理过程:

1) 取依赖有向图的最顶级节点作为当前处理节点。若当前处理节点为复合更新节点,则进行 2),否则跳转到 5)。

2) 对于复合更新中包含的多个更新,按照数据源进行分组,记作 $\sum_{i=1}^m \langle \{DU_i\}, \{SC_i\} \rangle$,其中 i 为数据源标识。

3) 针对复合更新,计算每个数据源的模式更新的最终结果。例如,若对于数据源 1,通过 2),有 $\{SC_1\} = \{U_1, U_2\}$,其中模式更新 U_1 为重命名属性 author 为 writer,模式更新 U_2 为删除属性 writer。则数据源 1 的模式更新的最终结果为:删除属性 author。

4) 基于模式更新最终结果,重新考虑数据源的数据更新。例如,若数据更新 $U_3 = \text{insert}(a, b)$, $U_4 = \text{insert}(c)$,模式更新的最终结果为删除第二个属性。则最终的数据更新为 $U_3 = \text{insert}(a)$, $U_4 = \text{insert}(c)$ 。

5) 若有模式更新,则重写视图定义;否则继续。

6) 通过补偿算法增量地修正视图内容。

7) 若视图维护成功,则从 UMQ 中去除当前更新,提交当前视图维护事务,转向 Step1,启动下一个视图维护事务。若以上任何步骤遭遇异常,视图维护事务终止,转向 Step1,部分滚回,重新进行更新的视图维护。

VM_SCNF 算法能够达到视图维护的强一致性。但是,由于算法采取复合更新的方法解决循环依赖问题,数据源的每一个状态不可能在实视图中都有一个与之对应的状态,故不能达到完全一致性。

3.4 算法有效性验证

我们实现了原型系统来验证 VM_SCNF 算法的有效性。试验数据包括:两个数据源,每个数据源包含一个具有 10000 条记录的关系。试验环境为:256M 主存,1.7GHz CPU, Windows 2000 + MySQL + Java 平台。用随机出现的乱序更新及更新丢失情况模拟不稳定网络环境。视图定义由两个数据源的关系进行 JOIN 连接而成。

由于视图维护一旦失败,必须放弃先前已进行的工作,重新开始维护过程,终止代价非常昂贵,故 VM_SCNF 算法采取

悲观策略,尽可能事先发现异常,矫正异常。当然,由于随后发生的模式更新仍然可能导致正在进行的视图维护出现异常,故悲观策略也不可能完全避免异常的发生。随着模式更新之间的时间间隔的增加,图 4 比较 VM_SCNF 算法和未考虑不稳定网络环境的 VM 算法的视图维护终止代价。

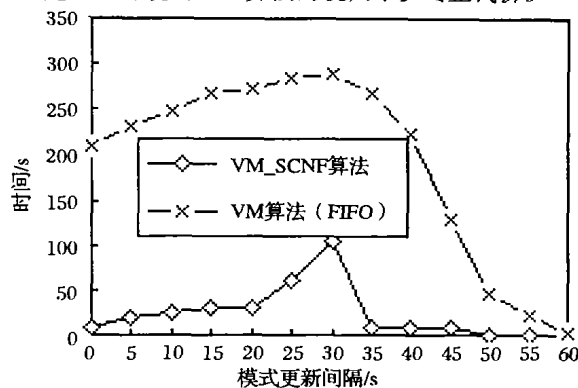


图4 不同模式更新间隔下算法性能分析

对于 VM_SCNF 算法,当模式更新之间的间隔很短时,所有的模式更新几乎同时到达,算法将 buffer 区域的所有模式更新进行排序,利用复合更新的方法处理依赖,矫正异常,终止代价很小。随着模式更新之间的间隔的增加,一旦新的模式更新异常终止正在进行的视图维护,终止代价随之相应增加。当新发生的模式更新终止即将结束的视图维护时,视图维护的终止代价达到最高。随后,当模式更新之间的时间间隔进一步加大,更新维护过程相当于串行化执行,消除了更新依赖,避免了视图维护异常,终止代价显著降低。作为比较,在不稳定网络环境中,VM 算法未正确处理更新的视图维护顺序。即使模式更新之间的间隔较短,由于模式更新之间乱序的频繁发生,视图维护过程将频繁遭遇维护异常,终止代价一直相对较高。当模式更新之间的间隔达到一定数值之后,在不稳定网络中,乱序现象发生频率迅速降低,终止代价相应迅速降低。最终,两种算法的终止代价趋于相同。

4 结语

传统环境中,实视图维护策略没有考虑数据源的模式更新。文献[5]中提出的 SDCC 是第一个同时考虑数据源的数据更新和模式更新条件下的视图维护问题。但是它限制每次模式更新之前,数据源要向视图管理器提出申请,等待批准。这种并发以牺牲数据源的自治为代价,不适用于松散耦合环境中的视图维护。

本文利用概念化事务模型解决视图一致性维护问题,把数据源本地更新和相应的视图维护过程看作一个全局事务。提出的 VM_SCNF 算法很好地解决了不稳定网络环境中,基于模式更新的实视图一致性维护问题。

参考文献:

- [1] ZHUGE Y, GARCIA-MOLINA H. View Maintenance in a Warehousing Environment[A]. SIGMOD[C], 1995.
- [2] LEE A, NICA A. The EVE Approach: View Synchronization in Dynamic Distributed Environments[A]. IEEE TKDE[C], 2002.
- [3] NICA A, RUNDEN E. View Maintenance after View Synchronization[A]. Proceedings of IDEAS[C], 1999.
- [4] CHEN S, CHEN J, ZHANG X. Detection and Correction of Conflicting Source Updates for View Maintenance[A]. Proceedings of IEEE ICDE[C], 2004.
- [5] ZHANG X, RUNDEN EA. Integrating the Maintenance and Synchronization of Data Warehouses Using a Cooperative Framework[J]. Information Systems, 2002, 27(4): 219-243.