异构分布式环境下的约束管理问题*)

Constraint Management in Heterogeneous Distributed Environment

杨晓春 王 丹 王 斌 王国仁 于 戈
(东北大学计算机软件研究所 沈阳110006)

Abstract Distributed integrity constraints arise naturally when information systems interoperate, due to interdependencies between data. Traditional constraint management is reasonable assumptions in centralized or tightly coupled distributed environment, they typically do not hold in loosely coupled heterogeneous environments, and traditional constraint management techniques are therefore inapplicable in such cases. This paper reviews the basic description for constraint management and provides some issues including architecture, knowledge model and execution model that should be considered in heterogeneous distributed environment.

Keywords Constraint management, Heterogeneous distributed environment, Event, Knowledge model, Execution model

1 引言

随着网络技术的发展,蕴藏巨大信息的网络可以被视为存储信息的数据库,但这种分布式环境要求具有更有效、复杂的信息处理能力。当数据存储在网络上松耦合、异构的系统中,彼此又存在联系时,就产生了数据的完整性约束问题。例如,虚拟企业[1]为物理上分布的结点提供高度的交互空间,以便于这些事先存在的企业可以随时决定是否加入到一个信息共享和交换的网络中。而这些企业是自治、异构且彼此独立的,可能使用不同的信息管理系统和控制策略。企业内部的数据源间以及企业之间都存在潜在的制约关系。

在分布式、异构环境下的约束管理问题比集中式环境下更加复杂,究其原因有:

·松耦合、异构的环境典型地不支持多数据库事务。因此,没有现成的机制保证跨越多数据库的一致性查询和更新,而且也没有对于某个执行过程的原子性定义。

·每个数据源支持不同的访问和约束监控能力。例如,一个"不合作"的数据源可能只提供对数据的只读访问,而另一个"合作"的数据源可以提供对数据的写访问,并具有自动公告更改数据的能力。

·在检测和存储涉及到多个场地的约束时会存在一定的通讯延迟,不可能保证数据在系统运行的任意时刻都是一致的。

国外学者对于松耦合系统中的约束管理进行了较深入的研究。为支持松耦合约束管理,提出了维护简单算术约束的分界协议[2];有的对事务概念进行扩展[3],通过适当削弱调度的正确性实现分布式环境下的约束管理。但这两种方法大大限制了涉及约束的数据项,例如约束只能在局部串行的数据之上[4]。一些研究临时数据库的学者尝试对数据库进行扩展,允许基于临时信息的有效存储和索引[5],但是用于分布式环境的、基于消息的通讯和基于事件的方法[6]比临时逻辑编程方法[5]更自然和方便。

基于事件的方法是确保分布式对象间信息的完整性约束,实现约束管理的主要方法,它包括定义知识模型和执行模型。知识模型用于描述基于事件的约束规则,而执行模型定义系统的运行策略。在分布式环境下,知识模型和执行模型都要依赖于系统的体系结构,并且要对集中式环境中的约束管理进行有效的扩展。本文结合约束管理目前的研究现状,总结、归纳了在松耦合、异构、分布环境下约束管理面临的问题及对策。

2 体系结构

体系结构方面,提供信息的数据源(DS)与规则管理器不再是集中式,而是分布式的。图1给出了分布式环境下约束管理的通用体系结构[6],描述了约束管理器(CM)、数据源和应用之间的关系。DS可以是关系数据库系统、对象数据库系统、文件系统、XML文档系统

*)本课题得到863/CIMS主题(863-511-946-003)和第六届霍英东青年基金资助。

等，每个DS有自己特殊的接口，接口仓中记录本场地DS的接口定义，为了使约束管理器不必考虑局部场地的异构性，CM转换器访问局部DS接口，形成统一的接口标准，并从CM接口输出给CM解释器，各场地的CM解释器组合在一起构成分布式约束管理器，约束管理器在系统运行时负责访问候选策略，动态决定相应的执行策略。

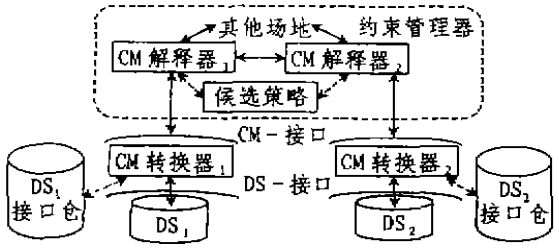


图1　分布式环境下约束管理的体系结构

3　知识模型及扩展

描述基于事件的约束规则通常包括三个部分：事件、条件和动作，称为ECA规则，即当一个事件发生时，检测条件是否满足，如果满足则执行指定的动作。形式如下：

on event if condition then action

约束规则的知识模型涉及到多方面的因素[7]，如表1所示。

表1　知识模型的基本描述

类型	因　　素
事件	源：{结构操作、行为调用、事务、应用程序、例外、时钟、外部事件}
	粒度：{实例、集合、分布式组件}
	类型：{简单事件，复合事件}
	操作：{or、and、seq、closure、times、not}
	消费模式：{最新事件、时序事件、累计事件、连续事件}
条件	角色：{强制，可选、省略}
动作	选项：{结构操作、行为调用、更新规则，废弃通知，客观动作，替代动作}

3.1　基本描述

事件是在某个时间点上瞬间发生的，具有原子性的操作。一个事件的产生与监控很大程度上依赖于产生事件的源，包括：

·结构操作：针对某个结构的操作而导致事件的发生。例如，插入、更新、删除、访问DS中的某个数据项。

·行为调用：执行用户定义的操作而导致事件的发生，例如，点击界面中的按钮而产生了新的界面。

·事务：事务命令而导致事件的发生。例如，事务的开始、废弃与提交等。

·应用程序：针对用户输入的信息，由应用程序显式定义事件的产生。

·例外：系统操作的例外而导致事件的发生。例如，试图访问没有被授权的信息。

·时钟：指定事件在某个时间点上或时间段内产生，如确定虚拟会议召开时间等[3]。

·外部事件：由系统外的其它因素（包括自然因素与人为因素）导致事件的发生。

事件粒度指明产生事件的客体的规模，可以是类的某个具体实例、多个对象组成的集合或分布式组件本身。

事件的类型包括简单事件与复合事件：

·简单事件：系统中预先定义的事件。简单事件包括数据库事件、临时事件、显式事件，数据库事件与数据库操作有关，临时事件是与时钟有关的事件。

·复合事件：由简单事件和复合事件构成的一个单一事件被称为复合事件。复合事件可以由六种事件操作符构成，分别是：或操作（or）、与操作（and）、任选操作（any）、序列操作（seq）、阶段操作（peridiodic）和取非操作（not）[8]。

根据应用的不同特征，决定了复合事件的不同消费模式。

·最新事件：多次发生的同类事件只修改某一个数据项。例如，医院病房的监护、全局定位跟踪等。此时，主要考虑最迟发生的事件。

·时序事件：不同事件实例间存在联系，因此要检测每个起始事件，维护不同事件实例间的联系。例如，在应用程序的bug报告与发布之间，在事务开始与结束之间都存在这种联系。

·连续事件：所有的事件实例都必须被考虑，适用于诸如股票市场的趋势分析和预测等应用。

·累计事件：适用于有截止事件的应用，例如在银行业务中，每天要记录下所有的存款和取款交易，而在晚上12:00进行平帐处理。

条件的角色指明规则是否包含条件，在某些情况下，条件被隐藏，规则变为EA（事件-动作）规则。而规则中的动作可以更新数据源或规则集的结构，在数据源上执行一些行为调用，向用户或系统管理员发布通知，废弃事务或使用do-instead实施动作的选择[10]。

3.2　扩展描述

3.2.1　时间约束　在集中式环境中，事件的执行顺序是首要的；而在分布式环境中，时间则比顺序重要。一个复杂的问题是设置时间印（time-stamp），在集

中式系统中有单一的时钟用于记录事件发生的顺序，即由于两个不同的简单事件不可能同时发生，它们总可以被排出先后顺序。于是，复合事件的产生时间可以由构成它的成员事件来决定。而在分布式系统中，没有全局时间，每个场地有自己的局部时钟，不同场地中的多个事件可以同时产生，由于构成复合事件的成员事件可能来自于不同的场地，要求事件的时间印应具有全局意义。因此，局部时钟必须通过特殊的时间服务器进行同步，向每个场地传输时间信息，并且允许两个同时发生的事件具有一定的时间差。因此，除了要有局部时间印用以构造场地内的复合事件外，还需要全局时间印作为监测场地间复合事件的规范形式[11]。

3.2.2 接口描述 在分布式环境下，对 DS 和 CM 转换器的访问与控制只能通过接口来完成。文[12]给出了一个支持多数据库环境的约束维护方法，它主要面向关系数据库，支持基本的 SQL 操作，并且要求场地间具有持久化的队列机制。类似地，文[13]基于同构的关系接口描述了一个支持数据库间约束的框架。这两种方法都不适用于异构环境下的约束管理，数据源可能为约束管理器提供不同的接口，没有统一的规范。

因此，文[6]采用了基于事件模板的接口描述方法，对于涉及到约束的每个数据项，使用基于规则的符号，描述数据项如何被读、写，及如何被监控。事件模板的描述形式如下：

event-template ε=(Time, Desc, Old, New, Rule, Trigger)

分别代表事件的发生时间、事件的类型描述、事件发生前数据库的状态、事件发生后数据库的状态、事件源对应的规则(作为哪个规则的执行结果)、将触发哪个规则的执行。于是，规则的描述被演化为用事件模板表示的形式：

$$Rule=\varepsilon_1 \wedge C_1 \rightarrow C_2 \wedge \varepsilon_2; T \leqslant \delta$$

如果一个事件能够匹配 ε_1，并满足条件 C_1，则在 δ 秒内将导致所有能够满足条件 C_2，并能够匹配 ε_2 的事件发生。

4 对执行模型的扩展

执行模型是在知识模型的基础上，指定约束规则集在运行时的执行策略。执行模型与提供约束管理的系统的体系结构密切相关，分为五个阶段，如表2所示。

与集中式环境不同，针对不同的处理阶段，异构、分布式环境下的约束管理主要考虑以下几方面问题。

表2 执行模型的五个处理阶段

阶段	内容
标识阶段	确定由事件源产生的事件何时发生。
触发阶段	使用已产生的事件触发相应的规则，与该规则相关联的事件构成事件的实例。
评价阶段	评价触发规则的条件，满足条件的所有规则实例构成一个规则冲突集。
调度阶段	指定规则冲突集是如何被处理的。
执行阶段	执行被选中的规则实例的动作，该动作的执行可能导致一系列规则的相继执行。

4.1 检测复合事件

在标识阶段，事件检测可以在场地间进行，每个场地包含一个局部事件监测器和一个全局事件监测器[11,14]。局部事件监测器与集中式环境中的类似，只监测在局部场地上产生的事件，而全局事件监测器负责监测场地间的复合事件，首先将与事件相关的组件注册到各自所在的局部监测器中，于是，全局事件的监测就可以分布到不同的场地上执行。只要事件被监测到，监测信号被发送到与该事件相关的所有场地的监测器上。类似地，规则管理器也要提供相应的分布式处理策略：1)提供一个中心规则管理器；2)每个场地有自己的规则管理器；3)全局规则管理器维护全局规则集。

考虑一个复合事件 E=seq(E_1 and E_2, E_3, E_2 and E_4)，E 表示 E_1和 E_2被检测后才能检测 E_3，E_3被检测后才能检测 E_2和 E_4，由于构成复合事件的操作是 and 和 seq 操作，所以要求在不同场地的 E_1，E_2，E_3，E_4都必须检测到才能标识该复合事件 E，如果事件的执行序列是 $e_1^1, e_1^2, e_2^1, e_3^1, e_2^2, e_4^1, e_3^2, e_4^2$，则针对第2.1节提出的四种事件消费模式可以检测到不同的复合事件的实例。

·最新事件：只使用最近发生的起始事件进行匹配，并不是所有成员事件的发生都被用于检测一个复合事件。一旦复合事件被检测到，考察所有参加匹配的成员事件，如果不是最近发生的起始事件，则不能用于以后的事件匹配。因此可能的事件实例是$\{e_1^2, e_2^1, e_3^1, e_4^1\}$和$\{e_1^2, e_2^2, e_3^2, e_4^2\}$。

·时序事件：在时间轴上，当复合事件的起始与结束配对时，就认为一个复合事件的实例被检测到，最早起始的成员事件匹配最早结束的成员事件。一旦复合事件被检测到，使用每个成员事件的最早发生的实例构成该复合事件，所有参加匹配的成员事件不能用于构成该复合事件的其它实例。因此可能的事件实例是$\{e_1^1, e_2^1, e_3^1, e_4^1\}$和$\{e_1^2, e_2^2, e_3^2, e_4^2\}$。

·连续事件：复合事件的检测依赖于在各时间点的成员事件，每当有一个起始事件发生，就开始对一个复合事件进行检测，这种检测方法特别适用于对事件序

列上对某些趋势的跟踪分析，一个起始事件至少可被使用一次，因此可能的事件实例是$\{e_1^1, e_2^1, e_4^1, e_5^1\}$、$\{e_1^2, e_3^2, e_4^2, e_5^2\}$和$\{e_1^3, e_3^3, e_4^3, e_5^3\}$、$\{e_1^3, e_3^3, e_4^3, e_5^4\}$。

·累计事件：记录所有构成复合事件的成员事件的发生情况，直到一个复合事件被检测到。一旦复合事件被检测到，所有被记录的成员事件均不能用于匹配其他实例。因此，E的事件实例是$\{e_1^1, e_2^1, e_3^1, e_4^1, e_5^1\}$。

时序事件与连续事件的本质区别在于，时序事件的起始事件只能与唯一的终止事件进行配对，而连续事件的多个起始事件可与同一终止事件进行配对。与连续事件不同，累计事件中任何一个发生的事件都不能参与同一复合事件的两个不同的实例。关于复合事件与场地通讯及错误处理的关系将在第4.3节讨论。

4.2 通讯负载

在约束管理的各个处理阶段，由于体系结构的变化导致通讯负载问题的产生。例如，CORBA提供事件服务用于支持对象间的异步通讯。通讯通过事件通道获得，一个事件通道是连接事件生产者与消费者的队列。通常，有两种事件通讯模型，一是PUSH模型：事件生产者将事件主动传输给事件消费者；二是PULL模型：事件消费者向事件生产者请求事件数据。

在这两种模型中，事件生产者与事件消费者都清楚地知道事件处理过程，即必须要显式地传输事件。这违反了基于事件的不介入原则，且CORBA不显式支持复合事件的处理[15]。

在讨论支持主动功能的不同组件间通讯时，应考虑以上选项。例如，标识阶段包含事件源与事件监测者间的通讯，是遵循PUSH模型还是PULL模型。对于集中式事件监测器，每个事件都与系统相关，PUSH模型很有效。但由于每个事件必须与事件监测器通讯，导致严重的通讯瓶颈问题。同样地，在触发阶段，事件监测器与规则管理器间的通讯也要遵循PUSH或PULL模型。PUSH模型意味着要显式地向规则管理器提供它感兴趣的一组事件集，而PULL模型则要定期地检测系统中可能会产生事件的所有组件[16]。因此，要根据不同的应用情况选择不同的事件通讯模型。

4.3 错误处理

集中式环境中的错误处理通常考虑与系统相关的细节，例如死锁导致系统停止，操作系统或硬件故障等。而在分布式环境中，导致错误处理的因素增多，包括通讯延迟、通讯中断。这两种情况是采用主动机制的约束管理中各组件要时刻面临的问题。

解决手段依赖于组件与错误造成的影响。例如，在标识阶段，由于通讯延迟，可以导致全局事件监测器接收到不同的事件产生顺序，"意外动作可以选择异步评价或同步评价"[11]。异步评价忽略了可能出现的延迟事件，对于一个复合事件监测器，如果有合适的事件到达时就开始进行事件评价。因此，如果考虑复合事件的执行顺序时就会发生错误的评价结果；而同步评价要等待所有与复合事件相关的延迟事件都到达后，才能进行评价。当复合事件$E=E_1$ or E_2时，如果E_1场地发生错误，异步评价只考虑E_2就可以进行评价，而同步评价则不能进行。虽然同步评价能保证事件的执行顺序，但如果有通讯中断或场地失败的情况发生，则评价将被长期地锁住。

正如Schwiderski指出的，选择何种策略主要取决于具体应用。同理，在条件评价与动作执行期间的错误处理也应被考虑。对于动作执行的可选方法是选择一个可以替代的动作进行执行。

4.4 条件评价

在松耦合的分布式环境中，为了降低由于通讯故障导致的错误，提高约束管理的效率与正确性，希望尽可能地减少场地间的信息交互，而条件评价通常与局部DS相关，这意味着CM不仅要调度局部DS完成相应操作，还要经常地访问DS的状态，以判断是否满足约束规则中的条件。单纯基于数据库理论的查询优化不能从本质上改变访问局部DS的次数。采用主动机制的增量维护方法[17]试图通过提供单向的PUSH机制减少通讯代价，但并不是每个增量都对约束管理有意义，当大量的增量数据与约束规则无关时，通讯代价反而增高。Stanford大学提出了一种基于关系代数的优化静态分析方法[18]，该方法预先评价每次对局部DS的操作是否与约束规则中的条件相矛盾，如果矛盾，则不必进行条件评价；但如果不矛盾，则还要访问局部DS进行条件匹配。

4.5 一致性维护

不可能保证在任何时刻整个系统都处于一致性状态。通常，使用不严格的维护策略，即给定时间范围。如第2.2.1节所述，在集中环境中，只有顺序是重要的，而在分布式环境中，时间起着重要的作用。时间约束策略为分布式环境提供了一个较宽松的一致性维护机制[6]，虽然这种方法相对简单，但适合于大部分松耦合环境下的约束管理，例如，银行规定只能在22:00至第二天2:00期间内作平帐处理。这种方法为现实环境中的大部分情况提供了形式化的保证，是传统技术不能给予保证的。但若用户或应用想直接对分布式系统进行一致性维护，则必须要对很多方面进行综合考虑。通常，系统提供的保证越弱，使用起来越难。

结束语 分布式环境中，由于互操作的信息系统内部的数据间存在依赖关系，导致完整性约束管理问题的产生。传统约束管理技术只简单地考虑原子事务、锁和一致性查询。这些假设是在集中式、紧耦合环境

中，它们典型地不支持松耦合，异构环境，因此，传统约束管理技术不再适合，另外，异构环境导致不同的信息源可以为约束管理提供不同的机制与工具，这使得约束管理问题更加复杂，目前，异构环境下的约束管理既不能使用监控策略，又不能通过即席查询被监控，它们都具有错误倾向，能导致不可恢复的数据源间信息的不一致性。本文总结了分布式，异构环境下约束管理面临的问题与挑战，并例举了前人的相关工作，及目前存在的问题。

参 考 文 献

1 Wood A, Milosevic Z. Describing Virtual Enterprises: the Object of Roles and the Role of Objects. OOPSLA'98 Workshop on Objects, Components and the Virtual Enterprise [EB/OL]. Available at: http: // www. cs. ted. ie/ Virtues/ocve98/proceedings/003. html

2 Barbara D, Garcia-Molina H. The demarcation protocol: A technique for maintaining linear arithmetic constraints in distributed database systems [A]. In: Proc. of the Intl Conf. Extending Database Technology [C]. Vienna, Austria. 1992. 373～388

3 Elmagarmid A. Special issue on unconventional transaction management. Data Engineering Bulletin, 1991, 14(1)

4 Breitbart Y, Garcia-Molina H, Silberschatz A. Overview of multidatabase transaction management [J]. The VLDB Journal, 1992, 1(2): 181

5 Abadi M, Manna Z. Temporal logic programming [J]. Journal of Symbolic Computation, 1989, 8(3): 277～295

6 Chawathe S, Garcia-Molina H, Widom J. A toolkit for constraint management in heterogeneous information systems [A]. In: Proc. of the 12th Intl Conf. on Data Engineering [C]. New Orleans, Louisiana. 1996. 56～65

7 Norman W P, Oscar D. Active Database Systems [J]. ACM Computer Surveys, 1999, 31(1)

8 Yang X, Wang G, Yu G, Lee D. Modeling Enterprise Objects in a Virtual Enterprise Integrating System ViaScope [A] Internet Applications In. Proc. of the 5th ICSC. LNCS 1749 [C]. German: Springer-Verlag, 1999. 166～175

9 Chakravarthy S, et al Composite events for active databases: semantics, contexts and detection[A] In Proc of the 20th VLDB Conf. [C]. Santiago, Chile. 1994. 606～617

10 Stonebraker M, Jhingran A, Goh J, Potamianos S. On rules, procedures, caching and views in database systems [A]. In. Proc. of ACM SIGMOD. 1990. 281～290

11 Schwiderski S. Monitoring the behavior of distributed systems [D]: [Ph. D. Thesis]. University of Cambridge, United Kingdom. 1996

12 Cret S, Widom J. Managing semantic heterogeneity with production rules and persistent queues[A] In: Proc. of the VLDB[C]. Dublin, Ireland. 1993. 108～119

13 Rusinkiewicz M, Sheth A, Karabatis G. Specifying interdatabase dependencies in a multidatabase environment[J]. IEEE Computer, 1991, 24(12): 46～51

14 Bacon J, Bates J, Hayton R, Moody K. Using events to build distributed applications[A]. In: Proc. of the International Workshop on Services in Distributed and Networked Environments(SDNE)[C]. 1995. 148～155

15 Koschel A, et al. Configurable active functionality for CORBA[A]. In: Proc. of the 8th ECOOP Workshop 7 on CORBA[C]. 1997

16 Bueltzingsloewen V, et al. ECA functionality in a distributed environment [A]. In Active Rules in Database Systems, Springer-Verlag, 1999

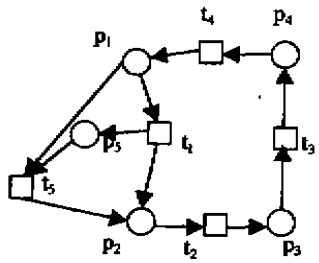
17 Grefen P, Widom J. Protocols for Integrity Constraint Checking in Federated Databases[J]. International Journal of Distributed and Parallel Databases, 1997, 5(4): 327～355

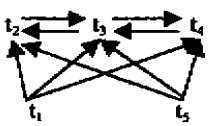
18 Baralis E, Widom J. Better Static Rule Analysis for Active Database Systems [J]. ACM Transactions on Database Systems, 2000

(上接第24页)

节点，公平依赖关系为弧的图，若变迁 t_1 公平依赖于 t_2，则从 t_1 和 t_2 有一条有向弧。

3 举例





公平依赖图

Bk-RVS={(1,1,1,1,0),(0,1,1,1,1)}

参 考 文 献

1 Murata T, Wu Z H. Fair Relation and Modified Synchronic Distance in a Petri Net. J. of the Frandlin Institute, 1985, 320(2)

2 Murata T, Wu Z H. Use of petri Nets for Distributed Control of Fairness in Concurrent Systems. In: Proc. of the First Intl. Conf. on Computer and Applications. 1984. 6

3 吴哲辉. 有界 Petri 网的活性和公平性的分析与实现. 计算机学报, 1989. 4

4 吴哲辉, 王培良. 公平网的一个充分必要条件. 科学通报, 1990. 35(16)

5 王培良, 吴哲辉. 公平网的一组直接判断条件. 计算机学报, 1993. 1