计算机科学 1996 Vol. 23 №. 6

关系型主动数据库的事件模型与监测*)

姜跃平 孙 明 李庆忠
(山东大学计算机科学系 济南 250100)

摘 要 This paper introduces a model for events in ECA rule based on relational active databases, in which the concept of event attributes and attribute function are extended. It brings in a detection framework based on event log and event interface, allowing the relative independence of event detection from rule processing. It also investigates the corresponding optimization strategies.

关键词 Active database, Event, Conditon, Model, Detection.

ECA(事件-条件-动作)规则是主动数据库系统中广泛采用的主动服务机制,其基本语义是:当事件发生时,即时地或在规定时刻检查条件是否满足,若是则即时地或在规定时刻执行规则的动作。与以前的SA(情形-动作)规则相比,ECA规则最显著的改进就是:将"情形"(situation)划分为事件和条件两部分,后者面向数据,前者则面向操作。由于事件成为一个相对独立的规则组成部分,并且相应地拥有相对独立的表示和监测机制,因而使ECA规则系统的反应能力具有良好的弹性一只要扩充或调整事件管理部分使之容纳新的事件类型,规则系统就能描述并监控更多样的数据库内外的状态变化。

对事件机制的研究只是近几年来才得到一定的重视[1-4]。原子事件类型的扩充和复合事件的出现使事件不仅能描述传统的数据库更新操作,而且能表达一些很复杂的时序逻辑[1,5]。这种能力对于证券交易、市场跟踪等应用领域是十分重要和有益的。然而,由于事件与方法和消息的某种"天然"的相近,这些研究[1,2,3,4]都是以面向对象数据库模型为基础的。我们认为,对关系型主动数据库的事件机制的针对性研究应该得到重视,理由有二:

(1)虽然面向对象数据库是未来的数据库系统一个有代表性的发展方向,目前和近期内市场导向却仍是关系数据库。

(2)虽然对象数据库上的事件研究的一些成果在关系数据库上也是有效的(如复合事件操作符的语义等),但在事件的内部存储和表示形式、事件可重用性、监测机制、实现效率等方面,由于对象数据库和关系数据库在对象标识、继承、优化技术基础等方面存在巨大差异,前期的有关成果不能适用于关系数据库。

本文给出了一个面向关系数据库的事件模型,给出了其内部表示方式、总体监测框架和复合事件表达式的计算方法,并对关系模型下相关的优化策略做了讨论。

一、事件模型

事件是在数据库系统运行中的某一特定时刻的一个对系统有某种意义的"发生"[1,2]。事件(包括复合事件)都具有"原子性"——在某一时刻,或者完全发生,或者根本不发生,没有第三种状态。

1.1 原子事件

原子事件(primitive event)是规则系统预先定义的,因而只有有限种。原子事件可以按照性质近似于聚类分成若干不同类型。在面向对象的主动数据库中,与对象操纵有关的方法的执行都是原子事件[1,2,3]。在关系库中,原子事件可以分为三类:(1)数据操纵类(2)事务类(3)时序类。其中,(1)对应于插入、更新、删除等改变数据状态的动作。(2)类事件在事务的某一特定时刻发生,如事务的开始、提交和撤消等。(3)是一个绝对的或相对的(相对于某原子事件)时间点。前者如"(1996.04.18/13:00:00)",后者如"insert(emp)+(00:03:00)"。

对前两类原子事件,可以更进一步地指定事件在动作(事务点)之前或之后发生,如"before insert (emp)"和"after commit"。其中"before"和"after"称

*)国家自然科学基金资助项目。姜跃平 博士生,研究方向为主动数据库和面向对象数据库。孙明 硕士生,研究方向为主动数据库。李庆忠 副教授,研究方向为面向对象数据库。

为事件修饰符。

1.2 复合事件

用系统规定的事件操作符把若干成分事件(原子的或复合的)联结起来,作为单个事件处理,称为复合事件。复合事件的发生也具有原子性,并且同样可以用事件修饰符界定具体发生时刻。导致某复合事件 Ec 发生的原子事件 Ep 称为其结束事件;"before(after)Ec"的发生时刻即为"before(after)Ep"的发生时刻。

复合事件的组成用事件表达式表示。

定义 1.1 事件表达式。(1)任意原子事件 E 是事件表达式;(2)如果 E1,E2,...,En 是事件表达式,则事件操作符作用于 E1,E2,...,En 上的结果为事件表达式;(3)任意事件表达式 E,(E)是事件表达式。

事件表达式的语义表达能力取决于系统支持的事件操作符,限于篇幅,我们仅列出最基本的事件操作符及其语义:

(1)AND(∧):事件 E1 和 E2 的合取,记作 E1∧E2。E1∧E2 发生当且仅当 E1 发生且 E2 已发生,或 E2 发生且 E1 已发生。

(2)NOT(¬):事件 E 的否定,记作¬E。¬E 发生当且仅当任意原子事件发生且这一事件发生不导致 E 发生。

(3)OR(∨):事件 E 的析取,记作 E1∨E2。E1∨E2 发生当且仅当 E1 发生或 E2 发生。

(4)*SEQUENCE*(→):*事件 E1 和 E2 的顺接*,记作 E1→E2。E1→E2 发生当且仅当 E2 发生且 E1 已发生过。

(5)ANY(∨'m):n 个事件 E1、E2,...,En 的 m 元析取,记作 ∨'m(E1,E2,...,En)。它发生当且仅当这 n 个事件之一发生且另有 m−1 个事件已发生。

复合事件能够有效地减少类似规则关于不同原子事件的重复定义。一个复合事件可以定义在多个表上,从而同时反应多个表的数据状态的变化。

1.3 事件属性与属性函数

原子事件和复合事件上都可定义一个或多个属性(类似于文[1]的 parameter)。特别地,可以为数据操纵和事务类原子事件定义标识属性。前者是所操纵的表名,如"insert(emp)"表示向 emp 表中插入元组;后者是所在事务的唯一标识(tid),如"before commit(t1)"表示事务 t1 的提交前时刻。标识属性不同的同种原子事件应视为不同事件。

其它属性一般用来记载事件发生时的数据库状态或其它信息(如用户标识)。例如对于复合事件 E=E1→E2,E1 为"insert(emp,max_salary)",E2 为"before commit(this)",以 E 为触发事件的规则的条件和动作为"如被插入元组中 salary 超过 1000 的,则给出警告"。这里 E1 的属性 max-salary 用来记录被插入元组的 salary 列的最大值。

注意到上例中,在 E1 与 E2 之间可能有其它动作改变被该 insert 插入的元组的 salary 值,所以 max-salary 必须在 E1 被检测到时立即算出并加以保存,而不能等到条件求值时再查取。故而我们允许为事件的属性定义相应的属性函数,其返回值即是属性值。这样的机制要求事件最好作为独立的实体存在于系统中。

二、内部表示与监测框架

2.1 内部表示

在关系模式下,事件的内部表示有三种方式:

(1)作为表的定义的一部分存放在系统表中。用户在定义表时,同时说明该表上能够产生哪些事件,这种方式的弊端显而易见。首先,事件模式修改极为不便;其次,复合事件难以表示,而且事件寄身于表的其它信息之中,难以扩展支持新的事件类型。

(2)作为规则定义的一部分。由于规则一般要作为独立的实体,由专门的系统表存储,使得事件可以动态地定义和修改,而不必依赖于产生它的表。此外,事件与规则的对应(联系)得到了自然的支持,但是事件的存储和管理仍然要依赖于规则,其可扩展性仍然受很大限制。尤其是事件本身不能再有自己的结构,使得事件的属性、属性函数和状态、标志等都无法表示。

(3)作为独立的实体,由专门的系统表存储,一个事件对应该表的一个元组。这种方式使得事件的模式改变更加灵活,既不依赖于用户表,也不依赖于规则,事件的存储格式可以独立修改,容易扩展新的实践类型。使事件拥有自己的结构,并且这种结构可以方便地扩充。此外,高级用户可以象操作其它表一样,利用系统原有的数据操作界面和工具进行事件的定义、删除和修改。但这种方式要求另立机制来建立和维护规则与时间的对应关系。

综上所述,事件作为独立实体显然是最好方式,值得注意的是,应该为每个事件建立一个唯一标识,它既起到 OO 模式下的对象标识的作用,又要是用户可见的,以便于激发事件。

2.2 事件日志(Event Log)

事件日志记录自规则系统启动依赖原子事件的每个发生。一个事件的发生由以下数据项组成:1)事件类型。原子事件的示例,如"insert";复合事件的类型即事件的唯一标识。2)标识属性。3)时戳。它标记

出事件发生的先后次序，计算并记录其属性的值以备条件、动作或其它复合事件使用。

复合事件的发生可以由事件日志中其成员事件的发生来表示。一个原子事件的发生可作为多个复合事件发生的"成员"。

2.3 监测框架

由于我们把事件视为独立的实体，规则系统需要在表(数据)、规则和事件之间维护正确的关联。在这样的系统中，数据或应用产生相应的事件，时间触发相应的规则。而事件的监测应该既独立于数据操纵，也独立于规则处理。

表与时间的关联可以在定义表时说明，也可以在运行时动态说明。一经说明，即由系统与表的其它模式信息一起存储和管理，我们称之为事件接口。时间接口的作用表现在：(1)用户只需指定事件，对事件的监测即由系统自动完成，不再需要用户用程序监控；(2)由于不在接口中说明的原子事件就不予产生，可以减少系统的原子事件发生数量，使系统性能提高。

规则与时间的关联也视为事件接口，有两种建立途径：(1)在定义规则时说明，称默认接口；(2)在运行时动态指定，称动态接口。建立动态接口并不意味着默认接口的改变，但系统只有在没有发现动态接口时，才根据默认接口控制规则的触发。动态接口是挥发性的。

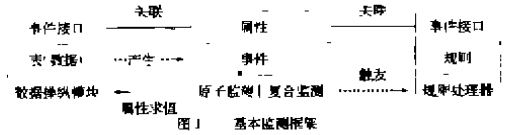

图1 基本监测框架

图1给出一个基本监测框架。原子事件监测器根据事件接口检测到表(或应用)上响应的原子事件的发生，并做必要的属性求值；同时通知复合事件监测器计算是否有新的复合事件的发生。最后所有发生交给规则处理器，触发相应的规则。

三、优化策略

对事件机制的优化主要有五种策略：(1)减少原子事件的产生数目，如2.3节中的事件接口机制；(2)定期清除已失去存在意义的"垃圾"事件；(3)根据具体的上下文定义，减少事件日志上的活动发生，从而减少复合事件的发生，如文[1]提出的"参数化上下文"；(4)对情形进行合理的E-C划分，把由事件监测器承担的计算性工作尽量交给计算器去做。因为条件求值可以看作是处理一组复杂的查询，在关系模型下，除了可以避免对表的重复性查询外，还可以利用许多传统的和新兴的查询优化技术，如多查询优化、昂贵谓词优化等等；(5)避免不必要的复合事件计算。下面仅就策略(5)展开讨论。

在实际系统中，当某一原子事件发生时，如果从头计算系统中定义的每一个复合事件是否发生，显然是不合理的。我们可以采取以下优化措施：

(1)记录复合事件的依赖事件集(即直接或间接地促使复合事件发生的原子事件的集合)，当不在依赖集中的原子事件发生时，不再对复合事件进行计算。事件监测器要为每个复合事件建立一个专用的结构，表示事件的构成和进展状况，如文[1]的事件树、事件图和文[2]的扩展优先自动机。依赖集则作为这种结构的附属信息存储，依赖集的内容随着事件模式或时间的进展状况的改变而改变。例如：设$E=P1 \wedge P2$，在P1和P2都未发生时，E的依赖集为{P1,P2}；若P1已发生，则依赖集变为{P2}。

(2)在相关的原子事件发生时，根据复合事件的当前状况(而不是从零开始)进行渐增模式计算。文[2]的自动机的状态变化可以很直观地表达复合事件的进展状况，但随着事件表达式的增长，这一机制所用的空间将呈指数增长。文[1]的事件图有效地避免了共用子事件的重复存储和监测，但却使得每个复合事件的进展状况难以表示，无法进行渐增式计算，还有待进一步的研究。

结束语 本文以关系模型为背景，在事件模型中扩展了事件属性和属性函数的概念；比较分析了事件的不同的内部表示方法，给出了基于事件日志和事件接口的监测框架；并对优化策略做了初步讨论。由于事件是ECA规则带来的新机制，因而还有大量的工作需要深入开展。其中包括：(1)怎样用更高级的语言描述复杂的实践；(2)怎样为具体的规则系统找到合理的EC划分；(3)怎样进一步提高复合事件的监测效率。

参考文献

[1] S. Charkravarthy et al., Composite Events for Active Databases: Semantice, Contexts and Detection, Proc. of 2nd VLDB, 1994

[2] N. H. Gehani et al., Event Specification in an Object-Oriented Database, Proc. of ACM SIGMOD, 1992

[3] E. Anwat et al., A New Perspective on Rule Support for Object-Oriented Databases. Proc. of ACM SIGMOD, 1993

[4] S. Charkravarthy et al., An Expressive Event Specification Language for Active Databases, Data And Knowledge Eng. 13(3), Oct. 1994

[5] A. P. Sistla et al., Temporal Triggers in Active Databases, IEEE TKDE 7(3) Jun. 1995