文章编号:1001 - 9081(2006)10 - 2417 - 04

基于触发器机制的主动数据库模型研究

张文超,张 璟,李军怀
(西安理工大学 计算机科学与工程学院,陕西 西安 710048)
(superjung@163.com)

摘 要:触发器是当前商用 DBMS 用于完整性实施和主动控制的一个主要机制。为了满足许多应用系统对数据库系统的主动性需求,提出了一种基于触发器机制的主动数据库的新模型,并且在此模型基础之上,运用了一种简单的、易于实现的方法来建立基于触发器机制的主动数据库。

关键词:主动数据库;事件—条件—动作规则;触发器
中图分类号: TP311 **文献标识码**:A

New model of an active database based on triggers

ZHANG Wen-chao, ZHANG Jing , LI Jun-huai
(*School of Computer Science and Engineering, Xi'an University of Technology, Xi'an Shaanxi* 710048, *China*)

Abstract: Trigger is an important mechanism of integrity enforcement and active control for commercial DBMS. In order to satisfy the active needs of many application systems for the database systems, a new model of an active database based on triggers was presented. And on the basis of this new model, a simple method that can be realized easily was applied to set up an active database based on triggers.

Key words: active database; Event Condition Action(ECA) rule; trigger

0 引言

传统的数据库是被动的,被动的 DBMS 由用户或程序驱动,它们仅当用户或应用程序提出明确的要求时才执行事务操作。但许多应用如计算机集成制造(CIM)、办公室流程控制等都要求对当前数据库状态条件进行监视,当条件满足时,能自动、实时地作出相应的反应(执行特定的活动)。实现数据库的主动性已经成为数据库技术发展的必然趋势。本文针对数据库的主动性需求,提出了一种基于触发器机制的主动数据库的新模型。

1 主动数据库技术的发展

数据库的数据模型一方面规定了库中数据的结构,即划定了其中数据的范围或类型,另一方面数据模型还规定了可供用户对其数据进行的各种操作或运算。数据库管理系统是其数据模型的一种具体实现,用户有请求,它就提供服务。可见,传统的数据库管理系统是一种典型的"服务程序",其提供的服务完全是被动的,丝毫不会主动为用户做点什么。主动数据库系统是一种能根据各种事件的发生或环境的变化主动给用户提供相应信息服务的数据库系统,它在许多领域中有广泛的应用。

主动数据库的一个很突出的思想是要让数据库系统具有各种主动进行服务的功能,并以一种统一而方便的机制来实现各种主动服务需求,除了"主动服务"之外,特别强调了用"统一的机制"来实现。到目前为止,这种机制主要是通过将一些规则预先嵌入数据库系统的办法来实现的。在主动数据库的研究中,其代表性的系统有主动面向对象数据库系统 HiPAC 系统、POSTGRES 系统、Alert 系统等。它们大多是在关系数据库系统(RDBMS)或面向对象数据库系统(OODBMS)的基础上进行扩充而增加主动性功能的。目前人们研究工作主要集中在主动数据库的实现模式和方法上。

2 模型原理与实现

2.1 相关知识

2.1.1 ECA 规则[1]

主动数据库系统实时检测被外部产生的数据库修改事件或其他外部事件,如时间性事件、硬件故障等触发的条件,如果条件为真,则执行相应的动作,而不用用户程序干预。这种触发事件、条件、动作的描述常被形式化表示成为 ECA(Event Condition Action)规则。

所谓 ECA 规则就是事件驱动的"事件—条件—动作"规则,即每条规则指明"当什么事件发生时,在什么条件下执行什么动作"。ECA 规则预先嵌入数据库系统,由事件探测器负责检查这些规则是否已经发生。一般形式可以表示为:

```
RULE <规则名> [(<参数>,...)]
   WHEN <事件表达式>
   IF <条件1> THEN <动作1>;
   ...
   IF <条件n> THEN <动作n>; (n>=1)
END-RULE [<规则名>]
```

上述事件驱动的"事件—条件—动作"规则的语义是:"一旦<事件表达式>所表示的事件发生,计算机就主动触发执行其后的 IF-THEN 规则。即如果<条件1>为真,则执行其后的<动作1>,并且接着逐个检查下一个 IF-THEN 规则,直至执行完为止。"

2.1.2 触发器[2]

标志一个 DBMS 具有主动性的关键特性是触发器

收稿日期:2006-04-07;**修订日期**:2006-06-15

作者简介:张文超(1981-),男,辽宁北票人,硕士研究生,主要研究方向:数据库技术、Web 应用; 张璟(1952-),男,陕西宝鸡人,教授,博士生导师,主要研究方向:Internet 技术及应用; 李军怀(1969-),男,陕西宝鸡人,副教授,博士,主要研究方向:分布式计算、CSCW.

(trigger)。虽然传统数据库不能提供完善的主动功能,但是目前的关系数据库 SQL Server, Oracle 等产品提供的"触发器"以及约束如:主键、外键、默认值等,都是主动数据库主动技术的体现。触发器是一种特殊的存储过程,在 SQL200n 中,触发器是与某一基表相关联的对象,其语法定义如下所示:

```
<触发器定义> ::=CREATE TRIGGER <触发器名称>
<触发动作时间> <触发事件>ON <基表名称>
[REFERENCING <旧值或新值别名列表>]
<触发动作>
<触发动作时间> ::=BEFORE | AFTER
<触发事件>::=DELETE | INSERT | UPDATE
[OF <触发器列的列表>]
<触发动作>::=[FOR EACH {ROW STATEMENT}]
[WHEN(<搜索条件>)] <触发 SQL 语句>
<触发 SQL 语句>::= <SQL 过程语句>
|BAGIN ATOMIC{<SQL 过程语句>;} END
<旧值或新值别名>::= OLD [ROW] [AS] <旧值相关名称>
|NEW [ROW][AS] <新值相关名称>
|OLD TABLE[AS] <旧值表别名>
|NEW TABLE[AS] <新值表别名>|
```

其中,DELETE 或 INSERT 触发器的 <触发器列的列表> 为空,而 UPDATE 触发器在未具体指定该列表时,<触发器列的列表> 也为空。<搜索条件> 是任意有效的 SQL 条件,可以包含布尔值表达式和复杂查询。通过使用 REFERENCING 子句,触发条件和动作可以访问触发事件所影响的行的旧转换变量和新转换变量。

2.2 主动数据库系统(ADBS)的新模型[3,4]

在功能上,本文提出的 ADBS 是由传统数据库系统(DBS)、事件探测器(EM)、事件知识库(EB)、触发器评价器(TJ)和事件处理器(ED)五部分组成,用公式表示是:

ADBS = DBS + EM + EB + TJ + ED

其中:

DBS 用来存储数据和对数据进行维护、管理与运用。

EM 是一个随时监视 EB 中定义的事件是否已经发生的监视模块,一旦监视到某事件已经发生时,就立即将该事件挂起,并启动事件知识库。在这里事件分为时间事件和非时间事件。

EB 是一个由事件驱动的一组知识组成的集合,其中每一项知识表示在相应的事件发生时,如何来主动地执行其中包含的由用户预先设定的动作。目前的实现中往往采用 ECA 规则来表示这种知识。

TJ 负责判断触发条件,并且当多个触发器被同一事件触发时,可以按一定的优先级进行条件评价,也可按一定的优先级选择被触发的活动。

ED 按照规定模式触发活动。

模型如图 1 所示。事件的某些操作原语、事务原语或者外部请求原语及时钟,向事件探测器发消息。事件探测器收到信息之后,随之将事务挂起,接着遍历事件知识库,并根据消息的有关参数确定该事件是否为事件库中已定义的事件。如果该事件已经在 EB 中被定义,那么立即调用触发器评价器,否则,将立即将之前挂起的事务重新启动。触发器评价器首先必须确定要进行时间事件评价还是进行非时间事件评价,然后创建条件评价器线程进行评价。情形评价器按优先级顺序评价所有被激活触发器(可能有多个触发器)中的情形。在有的情况下,情形可能没有限制,这时条件谓词就是恒真,此时触发器的语义为:当特定事件发生时就启动被触发的活动的执行。如果评价结果为真,即相应情形出现,那么评价结果将被传入事件处理器,由事件处理器负责动作的执行。最后解挂触发事务,恢复正常运行。

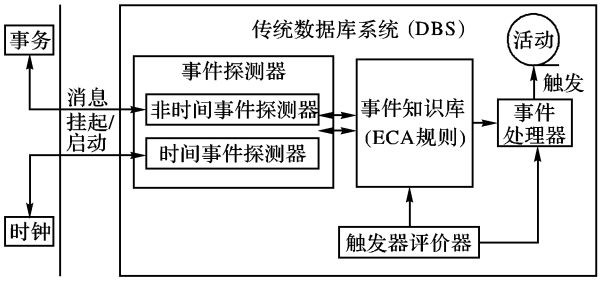

图 1 主动数据库模型

2.3 ADBS 模型的数据结构

在 ADBS 系统中,每一个事件可能触发多个触发器,可见事件和触发器之间存在着一对多的关系,为此可采用二级索引链表的形式来组织事件—触发器的数据结构(如图 2)。根据事件类型建立索引(Event-kind-index),该索引最后的属性是一个指向事件链表(Event-list)的指针,事件链表以单循环链表的方式构成,以便对事件链表进行遍历。对事件链表中的每一个事件节点,都由一个指针指向其对应的触发器链表(Trigger-list),这里触发器链表的构成方式与事件链表(Event-list)完全相同,以便对触发器进行动态维护。触发器链表中按照条件评价的优先级顺序存放着对应于该事件的多个触发器节点,每个触发器节点中都有触发条件和被触发活动等属性。被触发活动是以 SQL 语句的形式存在,可以放在触发器中。触发条件可能是一个布尔变量,也可能是一个复杂表达式,因此用指针 condition 将它连到触发器链表 Trigger-list 中的各个触发器节点。

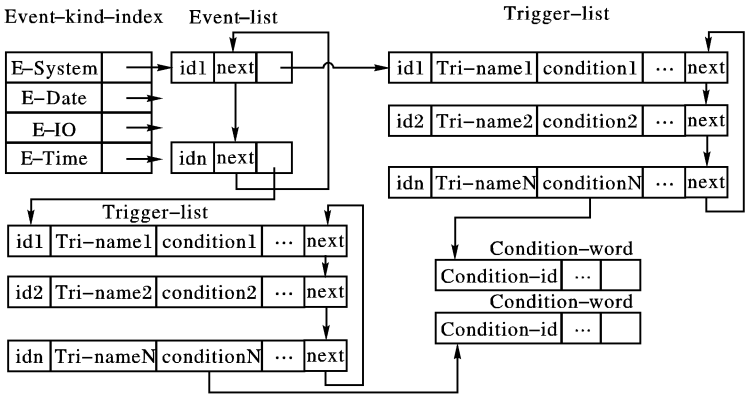

图 2 事件—触发器的数据结构

由于事件—触发器的数据结构是采用二级索引链表的形式来组织的,所以查找触发器的 Search 函数可以用分块查找来实现。由于索引项组成的索引表按关键字有序,则确定块的查找可以用顺序查找,亦可用折半查找;而块中的记录是任意排列的,则在块中的查找只能是顺序查找。

由此,索引分块查找的算法即为确定块的查找算法和块中查找算法的简单合成。

定义 1 为确定记录在查找表中的位置,需和给定值进行比较的关键字个数的期望值称为查找算法在查找成功时的平均查找长度(Average Search Length,ASL)[5]。

索引分块查找平均查找长度为:

$ASL = Ln + La + Lb$

其中:Ln 为查找索引表的平均查找长度,La 为在 Event-list 链表中查找指定事件的平均查找长度,Lb 为在 Trigger-list 链表中查找对应触发器的平均查找长度。

一般情况下,为进行分块查找,可以将数据分为 n 块(即索引长度为 n),Event - list 链表的长度为 a,Trigger - list 链表的长度为 b。又假定块中的每个记录的查找概率相等,则每块查找的概率为 $1/n$,块中每个记录的查找概率分别为 $1/a$,$1/b$。

若用折半查找确定所在块,则索引分块查找的平均查找长度为:

$$\begin{aligned} ASL &= Ln + La + Lb \\ &= \sum_{i=1}^{n} Pn_i Cn_i + \sum_{i=1}^{n} P\ a_i Ca_i + \sum_{i=1}^{n} Pb_i Cb_i \\ &= 1/n \sum_{j=1}^{h} j2^{j-1} + 1/a \sum_{i=1}^{a} i + 1/b \sum_{j=1}^{b} j \\ &= (n+1)/n\log_2(n+1) - 1 + (a+1)/2 + \\ &\quad (b+1)/2 \end{aligned}$$

(其中:$h = \log_2(n+1)$)

当 n 较大时,可有下列近似结果:

$ASL = \log_2(n+1) + a/2 + b/2$

可见,索引分块查找的平均查找长度,与索引长度 n、Event-list 链表长度 a 以及 Trigger-list 链表长度 b 有关。

3　ADBS 应用实例分析

下面以一个钢管质检系统中有关检验结果判定的一部分数据库为例,在文章提出的新模型的基础之上,简要介绍一下这种基于触发器的机理。

质检中心的检验信息都是以数字的形式来反映的,质量检验的最终结果需要根据不同的检验标准来判定。由于质量检验复杂多项,对不同规格不同材质的钢管检验,根据合同的不同检验项目又有所不同,而且国内与国外的检验标准更是有所区别。所以,该系统对检验结果的判定复杂性是一个不可回避的难点。

与普通数据库相比,该系统数据库的主动性体现在它能够根据不同的工艺卡,基于不同的钢管检验标准,随着钢管检验信息记录的变化而自动进行检验结果的分析与判断,并可以执行事先定义的指定操作。

钢管质检系统的组成如图 3 所示。

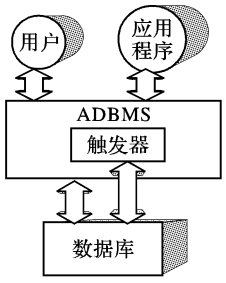

图 3　质检主动数据库系统

应用程序负责采集检验信息数据并写入机械性能记录总表 Jxxn_Record 中,而检验结果 result 是由数据库系统根据采集的检验信息(ten_str、yield、elongation)和与之对应的检验标准表 Chk_Std 中的范围信息来自动判定。当检验记录的某些字段被修改时,检验结果也会根据修改后的检验记录随之调整。同时在对表 Jxxn_Record 进行 insert、delete、update 操作之后,数据库系统亦会对不合格检验记录表 Unpass_Record 执行事先定义的操作。

其中 Update 操作的处理过程如图 4 所示。

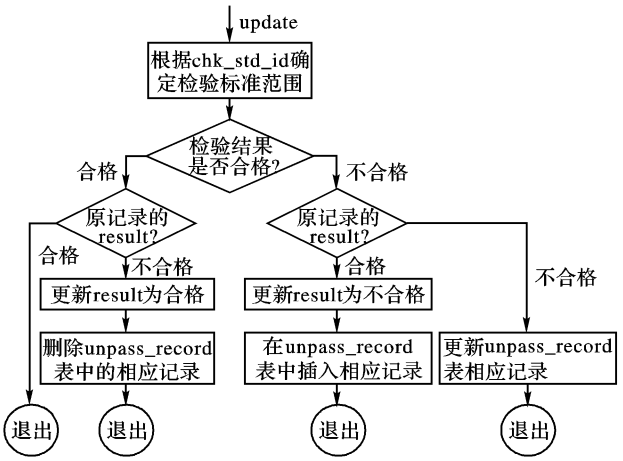

图 4　Update 处理过程

Update 规则可以表示为:

RULE ＜Update_Jxxn＞ [Insert]
　WHEN ＜UPDATE＞
　IF ＜检验结果为合格＞ THEN ＜系列动作 1＞;
　IF ＜检验结果为不合格＞ THEN ＜系列动作 2＞;
END - RULE [Update_Jxxn]

系统所用到的几个表如下:

表 1　检验标准表 Chk_Std

Chk_std_id	Tens_min	Tens_max	Yield_min	Yield_max	Elong_min	Elong_max
API5L	565	758	483	621	24	100
GB/T97	515	652	447	599	37	100

表 2　机械性能记录总表 Jxxn_Record

Pipe_no	tens	yield	elong	Chk_std_id	result
10206	611	498	36	API5L	合格
10240	743	511	48	GB/T97	不合格
10273	547	623	39	GB/T97	不合格

表 3　不合格检验记录表 Unpass_Record

Pipe_ no	tens	yield	elong	Chk_std_id	kind	reason
10240	743	511	48	GB/T97	jxxn	tens
10273	547	623	39	GB/T97	jxxn	yield

Jxxn_Record 的触发器 Tri_Jxxn_Update 的设计如下:

```
CREATE TRIGGER Tri_Jxxn_Update
ON Jxxn_Record FOR UPDATE
AS
    DECLARE @tens_str float, @yield float, @elongation float, @chk
    _std_id varchar(30), @tens_min float, @tens_max float, @yield
    _min float, @yield_max float, @elongation_min float, @
    elongation_max float, @result varchar(20), @result_new varchar
    (20), @reason varchar(50), @B1 varchar(10), @B2 varchar
    (10), @B3 varchar(10)
    SET @reason =''
    Select @chk_std_id = chk_std_id, @tens_str = tens_str, @yield
    =yield, @elongation = elongation, @pipe_no = pipe_no From
    insert
    Select @tens_min = tens_min, @tens_max = tens_max, @yield_
```

```
    min = yield_min, @yield_max = yield_max, @elongation_min =
    elongation _min, @elongation_max = elongation_max From chk_
    std
  Where chk_std_id = @ chk_std_id
    If @tens_str between @tens_min and @tens_max
      SET@ B1 = 'TRUE'
    Else
      SET @reason = 'tens_str ,'
    If @yield between @yield_min and @yield_max
      SET@ B2 = 'TRUE'
    Else
      SET @reason = @reason + 'yield,'
    If @elongation between @elongation_min and @ elongation_max
      SET@ B3 = 'TRUE'
    Else
      SET @reason = @reason + 'elongation'
    If (@B1 = 'TRUE') and (@B2 = 'TRUE') and (@B3 = 'TRUE')
      SET @result_new = '合格'
    Else
      SET @result_new = '不合格'
    Select @result = result From delete
    If @ result_new = '合格'
      Begin
        If @result = '不合格'
        Begin
          Update jxxn_record set result = '合格' where pipe_no = @
          pipe_no
          Delete from unpass_record where pipe_no = @pipe_no
        End
      End
    Else
    Begin
      If @result = '不合格'
        Update unpass_record Set tens_str = @tens_str, yield = @
        yield, elongation = @elongation, chk_std_id = @chk_std_id,
        chk_kind = 'jxxn', reason = @reason
      where pipe_no = @pipe_no
      else
      begin
        Update jxxn_record set result = '不合格' where pipe_no = @
        pipe_no
        Insert into unpass_record (pipe_no, tens_str, yield,
        elongation, chk_std_id, chk_kind) Values (@pipe_no, @tens_
        str, @yield, @elongation, @chk_std_id, 'jxxn')
      end
    End
```

以上触发器在表 Jxxn_Record 进行 Update 操作时被触发。此事件属于非时间事件中的事务操作事件,当事件探测器收到该事件的触发消息后,根据消息中的参数可以确定该事件在事件知识库中已被定义。由于只有一个触发器 Tri_Jxxn_Update 与该事件相关联,所以在触发器的评价时该触发器默认为优先级最高。在进行 Update 操作的过程中,如果发生错误,触发器将执行事务回滚操作。

4 结语

文章提出的主动数据库模型与方法,利用数据库本身的 DBMS 提供的触发器机制即可实现。当然由于 SQL 语言自身的不足和触发器功能的有限,这一方法所能实现的主动性也是有限的。然而,随着标准化工作的进行,触发器机制将越来越完善,可移植性也将越来越好,触发器的主动性功能将会越来越强。在文章提出的新模型的基础之上,可以通过引入面向对象的思想以及采用面向人工智能的语言(如 LISP 或 PROLOG)编程等方法来表示更完备的知识和在更高的层次上进行推理,从而建立起一个具有高度主动性的数据库系统。

参考文献:

[1] 姜跃平, 汪卫, 施伯乐, 等. ECA 规则的模型和行为特定理论[J]. 软件学报, 1997, 8(3): 190 - 197.

[2] Information technology - Databaselanguage - SQL - Part 2: Foundation (SQL/Foundation)[S]. ISO/IEC 9075 - 2: 2003(Z), 2003 - 07 - 25.

[3] 张沪寅, 陈珉, 文小军, 等. 主动实时数据库系统触发器模型的研究[J]. 武汉大学学报, 2002, 27(6): 642 - 646.

[4] 刘云生. 现代数据库技术[M]. 北京: 国防工业出版社, 2001.

[5] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社, 2001.

(上接第 2416 页)

[3] http://fimi.cs.helsinki.fi/data/, of the ICDM workshop on Frequent Itemset Mining, 2003.

[4] ZHANG SC, ZHANG CQ. Anytime Mining for Multi-User Applications[J]. IEEE Transactions on Systems, Man and Cybernetics(Part A), 2002, 32(4): 515 - 521.

[5] YAO J, LIU H. Searching Multiple Databases for Interesting Complexes[A]. In: Proc. of PAKDD[C]. 1997. 198 - 210.

[6] LIU H, LU H, YAO J. Identifying Relevant Databases for Multi - database Mining[A]. In: Proc. of PAKDD[C]. 1998. 210 - 221.

[7] ZHANG CQ, ZHANG SC. Database Clustering for Mining Multi-Databases [A]. In: Proc. of the 11th IEEE International Conference on Fuzzy Systems[C]. Honolulu, Hawaii, USA, May 2002.

[8] ZHONG N, YAO Y, OHSUGA S. Peculiarity oriented multi-database mining[A]. In: Principles of DMKD[C]. 1999. 136 - 146.

[9] ARONIS J, *et al*. The WoRLD: Knowledge discovery from multiple distributed databases[A]. In: Proc. of 10th International Florida AI Research Symposium[C]. 1997. 337 - 341.

[10] WROBEL S. An algorithm for multi-relational discovery of subgroups [J]. In: J. Komorowski and J. Zytkow(eds.) Principles of DMKD, 1997: 367 - 375.

[11] ZHANG SC, WU XD, ZHANG CQ. Multi-Database Mining[J]. IEEE Computational Intelligence Bulletin, 2003, 2(1): 5 - 13.

[12] ZHANG SC, ZHANG CQ, WU XD. Knowledge Discovery in Multiple Databases[M]. Springer, 2004.

[13] ZHANG CQ, LIU ML, NIE WL, *et al*. Identifying Global Exceptional Patterns in Multi-database Mining[J]. IEEE Computational Intelligence Bulletin, February 2004, 3(1): 19 - 24.

[14] ZHANG SC, ZHANG CQ, JEFFREY XY. An efficient strategy for mining exceptions in multi-databases [J]. Information Sciences. 2004, 165/1 - 2: 1 - 20.