第9章

触发器与主动数据库系统

2012. 04

目录 Contents

9.1 主动数据库系统

- 一、被动与主动数据库系统
  - 被动数据库系统
    - 传统数据库系统只能按用户或应用程序（用户事务）的要求对数据库进行操作，而不能根据发生的事件或数据库的状态主动进行某些操作，这样的系统称为被动数据库系统（Passive Database System）。
  - 主动数据库系统
    - 理想的数据库系统应能根据发生的事件（如：用户事务对数据库进行某种操作、时间事件、外来事件等）或数据库的状态，主动地进行某些操作（称具有主动数据库功能），而且，这种主动数据库功能是用户/DBA事先可定义的，这样的系统称为主动数据库系统（Active Database System）。

9.1 主动数据库系统

- 二、主动数据库系统的实现
  - 实现主动数据库系统的基本方法
    - 在数据库系统中引入规则(Rule)机制。因此，主动数据库系统有时也称规则系统(Rules System)。

9.1 主动数据库系统

- 二、主动数据库系统的实现(cont.)
- 当前，主动数据库系统的主要规则
  - 条件－动作规则 (Condition-Action rule, CA rule)
    - 当数据库达到某种状态时(即“条件”满足时)，触发DBMS执行“动作”。
    - **注:** CA规则作为主动数据库规则有缺陷，因此，当前大多数DBMS并不支持。
  - 事件－条件－动作规则 (Event-Condition-Action rule, ECA rule)
    - 当某个“事件”发生时，DBMS检测“条件”，若满足，则其执行“动作”。ECA规则也称触发器(Trigger)。
    - **注:** SQL标准从SQL:1999开始增设了触发器；当前，大多数DBMS已支持触发器，但实现功能和语法不尽相同，与SQL标准语法也不尽一致。

目录 Contents

9.2.1 ECA规则的表示

- ECA规则的种类

触发器类型		当“条件”满足时，“动作”的执行频度	
		Each ROW	Each STATEMENT
当“条件”满足时，“动作”的执行时刻/方式	BEFORE	行(层、事件)前触发器	语句前触发器
	AFTER	行后触发器	语句后触发器
	INSTEAD OF	行替代触发器	语句替代触发器

- **“事件”：SQL操纵语句(INSERT, DELETE, UPDATE)**
- **“动作”执行频度**
  - **Each ROW：对“事件”导致的每个“行”操纵，“动作”执行一次。**
  - **Each STATEMENT：对整个“事件”，“动作”执行一次。**
- **“动作”执行时刻/方式**
  - **BEFORE：“动作”在“事件”前执行。**
  - **AFTER：“动作”在“事件”后执行。**
  - **INSTEAD OF：“动作”替代“事件”而执行(“事件”语句不执行)。（SQL:1999中没有INSTEAD OF触发器，ORACLE系统中**

9.2.1 ECA规则的表示

- ECA规则在SQL中的表示
- SQL:1999触发器定义语法：

<触发器定义> ::= CREATE TRIGGER <触发器名>
{BEFORE｜AFTER} <事件> ON <表名>
[REFERENCING
OLD {ROW｜TABLE} AS <过渡行/表标识符>,
NEW {ROW｜TABLE} AS <过渡行/表标识符>]
FOR EACH {ROW｜STATEMENT}
[WHEN <条件>] <动作>;

<事件>::= INSERT｜DELETE｜UPDATE [OF <属性表>]

<条件>::= <SQL谓词>

<动作>::= <SQL DML语句>｜
BEGIN <SQL DML语句>[; <SQL DML语句>]… END

9.2.1 ECA规则的表示

- ECA规则在SQL中的表示(cont.)
  - <过渡行/表标识符>用于引用内存中“事件”操纵语句导致数据库表/行更新时的过渡值(Transition Value)。
  - 与OLD对应的是更新前旧值，与NEW对应的是更新后新值；当引用行值时，称过渡变量(Transition Variable)，当引用整个表时，称过渡表(Transition Table)。
  - 触发器/ECA规则定义作为模式对象存放数据字典中。
  - 注触发器/ECA规则可暂停(用DEACTIVE TRIGGER语句)、复活(用ACTIVE TRIGGER语句)、撤消(用DROP TRIGGER语句)。

9.2.1 ECA规则的表示

- 举例说明

```
CREATE TRIGGER sal_never_lower
AFTER UPDATE OF sal ON emp   /* 事件：更新emp表上sal列 */
REFERENCING
        OLD ROW AS oldtuple,  /*建立过渡变量（transition variable），
                                表示旧元组*/
        NEW ROW AS newtuple  /* 建立过渡变量，表示新元组*/
FOR EACH ROW
    WHEN oldtuple.sal > newtuple.sal      /* 条件：当薪水值变低时 */
    UPDATE emp
    SET sal = oldtuple.sal                /* 动作：薪水值恢复 */
    WHERE empno = newtuple.empno;
```

9.2.2 ECA规则的执行

- ECA规则的执行方式
  - 任何“事件”操纵语句的执行总是用户事务中的一部分或全部，一个用户事务中可能有能够触发多个触发器的(多个)“事件”。那么，这些触发器的“动作”操纵语句的执行与此用户事务的关系如何处理呢？
  - 这决定了ECA规则的不同执行方式。

9.2.2 ECA规则的执行

- ECA规则的执行方式(cont.)
  - 结合方式(Coupled Mode)：“动作”操纵语句作为“事件”操纵语句所在的用户事务的一部分被执行。根据执行时间的不同，进一步分为
    - 立即执行(Immediate Execution)方式：一旦触发“事件”发生，“动作”操纵语句立即作为此用户事务中的一部分而被执行。由于这种方式实现简单，大多数DBMS(如：ORACLE、IBM DB2)采用此方式。
    - 推迟执行(Deferred Execution)方式：“动作”操纵语句推迟到此用户事务末尾(EOT)而被执行。这种方式虽理想(因为用户事务结束前，可能一些原先破坏完整性约束的现象已被消除，因此，有些ECA规则事实上无需执行了)，但实现太复杂，很少有DBMS能实现此方式。

9.2.2 ECA规则的执行

- ECA规则的执行方式(cont.)
  - 分离方式(Decoupled/Detached Mode)
    - “动作”操纵语句组合成一个与“事件”操纵语句所在的用户事务有因果依赖关系（Causal Dependency）的衍生事务而被执行。这个衍生事务只有在此用户事务提交（Commit）后才能提交；若此用户事务撤消（Rollback），衍生事务也要撤消。

9.2.2 ECA规则的执行

- 连锁触发及其对策
  - 极端情况下，用户事务中的“事件”触发了ECA规则中的“动作”，由于“动作”也是操纵语句，因此，进一步触发了其他ECA规则中的“动作”，…，此时称发生了连锁触发(Cascading Triggering)。
  - 对于连锁触发，一方面需正确控制ECA规则的嵌套执行，另一方面需有效防止因循环触发而无休止执行(Nontermination)。
  - 通常做法是：为连锁触发次数规定一个上限，如16～64，当达到此上限时，DBMS强行撤消所有相关的ECA规则和用户事务。从这种意义上来说，ECA规则/触发器的定义要十分小心！

9.2.3 ECA规则的实现

- 主动数据库系统在传统的数据库系统基础上进行扩充或改造，引入ECA规则、实现主动数据库功能时，有不同的策略。
  - **松耦合法(Loose Coupling)**
    - 在应用层和传统DBMS之间加一层主动数据库功能模块。该模块捕获用户事务中的触发“事件”，检测“条件”，若满足，则向DBMS提交“动作”。—此法的优点是无需修改传统DBMS。但缺点是：主动数据库功能和DBMS功能分离，通信开销大、性能差，功能受限。乃早期方法。
  - **紧耦合法(Close Coupling)**
    - 将主动数据库功能集成到DBMS中，因此，需彻底改造传统的DBMS，大型DBMS(如：ORACLE、IBM DB2)均已完成此种改造。
  - **嵌入法(Embedded)**
    - 上述两种方法的折衷。由DBMS的查询处理子系统在适当时刻将ECA规则嵌入到用户事务的查询执行计划中，由DBMS执行。这种方法只能处理简单的规则。

目录 Contents

9.3.1 触发器的内部应用

- 这类应用是为DBMS本身服务的，如：
  - 完整性约束的维护
    - 是触发器的主要应用！
  - 导出数据(Derived Data)实时更新
    - 如：实视图(Materialized View)刷新
  - 数据库多副本一致性的维护

完整性约束的维护

- 选课SC表的INSERT操作

```
CREATE TRIGGER  sc_insert
 BEFORE INSERT ON SC
 REFERENCING
          OLD ROW AS  old_row,
          NEW ROW AS new_row
 FOR EACH ROW
 WHEN (NOT (EXISTS (SELECT * FROM STUDENT
                          WHERE old_row.SNO = new_row.SNO)
               AND
               EXISTS (SELECT * FROM COURSE
                    WHERE old_row.CNO = new_row.CNO)))
 ROLLBACK;
```

导出数据(Derived Data)实时更新

- 实视图(Materialized View)刷新
  - 女生成绩表FGRADE由SELECT语句导出
    - **SELECT SNAME, CNO, GRADE**
      FROM STUDENT, SC
      WHERE STUDENT.SNO= SC.SNO AND SEX = ‘女’；

SC表DELETE操作

```
CREATE TRIGGER sc_delete
 AFTER DELETE ON SC
 REFERENCING
        OLD TABLE AS old_table
FOR EACH STATEMENT
WHEN (EXISTS (SELECT * FROM old_table, STUDENT
          WHERE old_table.SNO = STUDENT.SNO
                    AND  SEX='女'))
BEGIN
   DELETE FROM FGRADE;
   INSERT INTO FGRADE
        SELECT SNAME, CNO, GRADE
        FROM STUDENT, SC
        WHERE STUDENT.SNO = SC.SNO AND SEX= '女'
END;
```

9.3.2 触发器的外部应用

- 这类应用是为用户应用服务的，如：基于库存量的自动订购单产生，数据的自动归档等。
- 这类应用实际上是将特定应用领域的业务规则(Business Rule)，抽象成ECA规则，以触发器而非传统应用逻辑的形式实现业务功能，大大简化了应用开发和维护。
- 【例】用触发器实现基于库存量的自动订购单产生。假设数据库中有如下表：
  - inventory(item, amount) /* 商品item的当前库存量amount */
  - min_level(item, min_amount) /* 商品item应保持的最小库存量min_amount */
  - reorder_level(item, order_amount) /* 商品item低于最小库存量时，需再次订购的数量order_amount */
  - purchase_orders(item, amount) /* 商品item的订购单*/

9.3.2 触发器的外部应用

- 定义行后触发器：

```
CREATE TRIGGER reorder_trigger
AFTER UPDATE OF amount ON inventory
REFERENCING
            OLD ROW AS old_row,
            NEW ROW AS new_row
WHEN new_row.amount <=
                        (SELECT min_amount  FROM min_level
                         WHERE min_level.item = old_row.item)
   AND  NOT EXISTS (SELECT  * FROM purchase_orders
                    WHERE new_row.item = old_row.item)
BEGIN
   INSERT INTO purchase_orders
   SELECT item, order_amount FROM reorder_level
   WHERE reorder_level.item = old_row.item
END;
```

作业

- 职员（emp）基表定义见课件。试用SQL:1999/SQL3语法定义一个名为empBandh的触发器来实现：一旦在emp表中删除一个员工的数据，只要此员工的工种不是“bandh”，就在emp表中恢复（插入）此员工的数据，将其工种（job）置为“bandh”，月薪（sal）置为2000.0，佣金（comm）置为NULL，其余属性不变。