ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 5, May 2014

Development of Rule Scheduler for Multiple Triggered Rules in Active Object-Relational Database Systems

S.Meenakshi[1], V.Thiagarasu[2]

Associate Professor, Dept. of Computer Science, Gobi Arts & Science College, Gobichettipalayam, India[1]
Associate Professor, Dept. of Computer Science, Gobi Arts & Science College, Gobichettipalayam, India[2]

ABSTRACT: Active object-relational database management systems (ORDBMS) react to the occurrence of predefined events automatically by the definition of Event-Condition-Action (ECA) rules. Whenever an event occurrence causes the triggering of multiple rules at the same time, the execution model of an active ORDBMS requires an explicit trigger rule conflict policy which regulates the selection and execution of multiple triggered rules. The rule execution model provide a mechanism known as rule scheduling which performs the ordering and execution semantics of multiple triggered rules to handle the trigger rule conflicts. Conventional rule schedulers have been developed to execute the multiple triggered rules sequentially based on system defined priority. However advanced applications require a rule scheduler to support both sequential and concurrent execution of multiple triggered rules based on static priority mechanism in active ORDBMS. Thus the objective of this research work is to develop an effective rule scheduler for executing multiple triggered rules both sequential and concurrent based on user-defined priority scheme.

KEYWORDS: Active object-relational database management systems, ECA-rules, triggers, rule scheduling, rule conflict

I. INTRODUCTION

Active ORDBMS extend the normal functionality of ORDBMS with support to monitor the changes in the database state and react to specific situations (events) automatically without user intervention. The reactive behaviour of active ORDBMS is represented by triggers in the form of Event-Condition-Action (ECA) rule paradigm [1]. In order to support the active functionality for monitoring and reacting to specific event occurrences, each active object-relational database system has a rule definition language that is used to define the specification of ECA rules for describing reactive behaviour generally referred to as the knowledge model and also possess an execution model that determines the rule processing at runtime [2]. The knowledge model describes the structural characteristics of rules such as types of events, context of conditions and actions, where as the execution model captures the runtime characteristics of the rule processing. The reactive behaviour of active ORDBMS depends on the rule execution model and hence rule execution semantics is an important aspect of active ORDBMS. In active ORDBMS, rule processing occurs when a database state is changed by the execution of update, insert, and delete operations requested by the user application.

Active ORDBMS react to the occurrence of predefined events automatically by the definition of ECA rules. In a generic active rule system, an event occurrence causes multiple rules to be triggered at the same time and hence their execution needs to be consistent which is an important issue faced by the rule execution model of active ORDBMS and this issue is known as trigger rule conflict. In order to solve trigger rule conflict issue, the rule execution model of active ORDBMS provides a mechanism known as rule scheduling which supports the execution of multiple triggered rules based on the two aspects: (i) the ordering of next rule to be fired is based on the conflict resolution policies such as dynamic and static priority schemes and (ii) the number of rules to be fired for execution by using either sequential or concurrent execution of all rules [2]. To maintain the deterministic property of the active rule processing, the database management system has to provide rule scheduling [3].

In this research work, section 2 presents the review of related work in rule scheduling. Section 3 specifies the mechanism of rule scheduling. Section 4 describes the development of rule scheduler to handle trigger rule conflicts in rule processing of an active ORDBMS. Section 5 shows the performance results of the developed rule scheduler for executing multiple triggered rules. The key aspect of this research is to reduce the over all execution time of rules by supporting both concurrent and sequential rule execution for improving the performance of trigger processing in active ORDBMS.

II. RELATED WORK

Management of multiple triggered rules is a major scheduling issue in active database systems (ADBS). Kerdprasop et al., [3] proposed rule scheduling based on the knowledge induced from the database state modification. In [4] rule scheduling has been introduced based on the estimation of rule execution probability using learning automata. Jin et al., [5] introduced concurrent rule scheduling using the logic of generated priority graphs. The research works in ADBS integrated with relational database systems support rule scheduling based on static priorities using numeric or relative scheme with sequential execution of all rules [2, 4]. In active ORDBMS standard (SQL3), multiple rule triggering has been performed by the rule scheduler based on the system defined priority with sequential execution of all rules [6, 2]. However, modern large scale applications demand support for handling complex data, detecting events come from different sources and the evaluation of conditions and the execution of actions can be performed on different situations, an enhanced rule scheduler is required to improve the ordering and execution of multiple triggered rules in active ORDBMS [7, 8, 9]. Thus in the presence of multiple triggered rules, the rule scheduler should support both sequential and concurrent rule execution based on user-defined priority scheme is an important research work in active ORDBMS.

III. RULE SCHEDULING MECHANISM

In active ORDBMS, an application defines rules and specifies the desired behaviour of each rule in the form of ECA. Once a set of rules is defined, the active database system monitors the occurrences of events pre-specified by ECA rules. Whenever a specified event occurrence can cause the triggering of multiple rules at the same time, the rule conflict occurs and the rule scheduler determines the order and the execution semantics of rule execution. When multiple rules are triggered at a time, the order of execution among the rules can be determined by assigning priorities and the execution of rules is based on their coupling modes.

A rule scheduling mechanism has been designed to handle trigger rule conflicts in active object-relational database systems [10]. In this work, the two important factors such as the priority and coupling mode are considered for designing the rule scheduler. The rule scheduler uses the user-defined priority scheme for ordering the rules. Rules can be placed in order using a numerical scheme in which each rule is given an absolute value. User-defined priority scheme provides better opportunity to order the rules based on the requirement of the application than system defined priority. Since user-defined priorities are dynamic – rules can be added, changed and dropped at any time during the existence of rule set.

The coupling mode specifies the time when a triggered transaction is executed with respect to the triggering transaction or an event. If the rule is in immediate coupling mode, then the triggered transaction can execute immediately and the triggering transaction has to wait until the completion of the triggered transaction. However if the rule is in deferred mode, then triggering (main) transaction can continue and the triggered rule has to wait for the completion of the triggering transaction. The rule scheduler has been designed to execute the multiple triggered rules concurrently with the same priority and sequentially with different priorities based on the coupling mode. This work uses the integrated architectural design to entail active behaviour into ORDBMS and the major component extensions incorporated for providing active capability include the rule manager and rule scheduler related with the underlying database management systems transaction manager.

IV. DEVELOPMENT OF RULE SCHEDULER

The rule scheduler is responsible for handling trigger rule conflicts when multiple rules are triggered at the same time by an event. The rule scheduler has been developed by introducing the design of rule scheduling mechanism [10]. The rule scheduler prototype has been developed in two phases such as rule maintenance and rule processing. The rule maintenance phase is used to define and maintain the specification of ECA rules for applications that require automatic monitoring and reactions. Based on the requirement of an application system, the user specifies the rule definition associated with event, condition and action, priority and the coupling mode as shown in Fig. 1. The rule processing phase is used to show the functions of the rule scheduler that includes the ordering of rules based on user-defined priority and the execution of rules with sequential and concurrent based on the immediate and deferred coupling modes as shown in Fig. 2. The developed rule scheduler supports concurrent execution of rules with equal priorities and

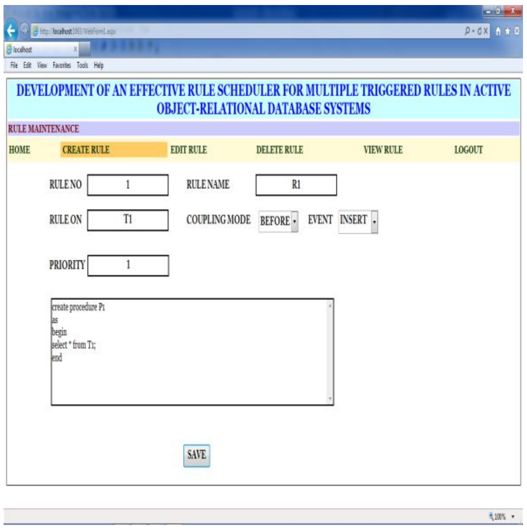
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 5, May 2014

sequential execution of rules with different priorities, whenever a primitive event occurrence causes the triggering of multiple rules at the same time.

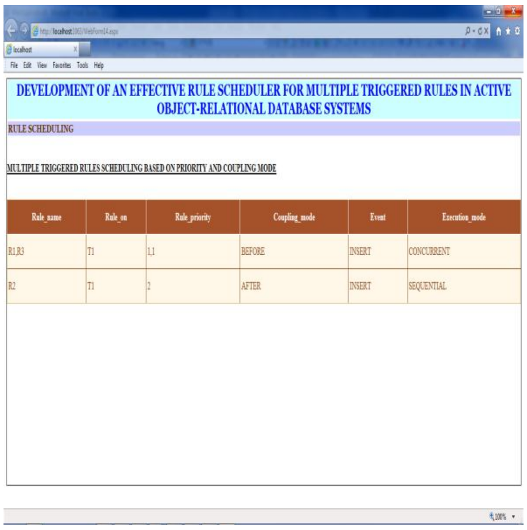

DEVELOPMENT OF AN EFFECTIVE RULE SCHEDULER FOR MULTIPLE TRIGGERED RULES IN ACTIVE OBJECT-RELATIONAL DATABASE SYSTEMS

RULE MAINTENANCE

RULE NO	1	RULE NAME	R1
RULE ON	T1	COUPLING MODE	BEFORE
		EVENT	INSERT
PRIORITY	1		

```
create procedure P1
as
begin
select * from T1;
end
```
http://localhost:1063/WebForm1.aspx
localhost
File Edit View Favorites Tools Help
HOME
CREATE RULE
EDIT RULE
DELETE RULE
VIEW RULE
LOGOUT
SAVE
100%

Fig. 1. User interface screen for rule creation


DEVELOPMENT OF AN EFFECTIVE RULE SCHEDULER FOR MULTIPLE TRIGGERED RULES IN ACTIVE OBJECT-RELATIONAL DATABASE SYSTEMS

RULE SCHEDULING

MULTIPLE TRIGGERED RULES SCHEDULING BASED ON PRIORITY AND COUPLING MODE

Rule_name	Rule_on	Rule_priority	Coupling_mode	Event	Execution_mode
R1,R3	T1	1,1	BEFORE	INSERT	CONCURRENT
R2	T1	2	AFTER	INSERT	SEQUENTIAL
http://localhost:1063/WebForm14.aspx
localhost
File Edit View Favorites Tools Help
100%

Fig. 2. Scheduled rules for execution

The developed rule scheduler implements the C# multi-threading system for concurrent rule execution. The important feature required by the scheduler is the ability to start and stop threads (rules) and this feature can be achieved using thread functions. Each rule triggered during the execution of an application has to be a separate thread. The execution of the thread is controlled by the scheduler depending on its priority and coupling mode. Whenever an event triggering multiple rules with different priorities and coupling modes at the same time the various steps taken into account are: The scheduler segregate the rules based on their coupling modes because immediate rule has to be executed first even if a deferred rule has a higher priority. After the segregation, the scheduler has to suspend the main (triggering) transaction. The scheduler can use the thread functions to suspend and continue rule processing. Thread execution can be controlled after the creation of the thread and the various parameters that are to be known when forming a thread to execute a rule are parent rule-id, priority and coupling mode.

The data structure named as process-rule-list is used to store the information includes the parent rule-id, priority and coupling mode that is obtained when creating a thread for that rule. The rule-id of the rule is the thread id's that are obtained when creating a thread for that rule. The system assigns a unique-id to every thread it creates. Rules triggered in the top-level transaction have the parent-id as 1. The priority and coupling mode can be obtained from the rule as they are provided during rule creation. The application proceeds normally until it researches a procedure call that is detected by an event and the overview of thread implementation is shown in Fig. 3. When an event is signaled, the event detector detects which of the primitive event is occurred. Once an event is detected, a set of rules triggered are as shown in the rule processing stage of Fig. 3. Every event detected in an application is wrapped with a Notify call that creates a thread for each rule. For each qualifying rules, a separate thread is created and inserted into the rule list. Then the Notify calls the function Process_imm-rules.

The function Process_imm-rules activates the scheduler and the rules are executed based on the priorities and coupling modes. The function then looks at the process-rule-list and finds the triggered rules in the immediate coupling mode based on the current thread-id. When the scheduler activates a rule, it wakes up the thread associated with the thread-id in the process-rule-list. Then the thread checks the condition of the rule and executes the action associated with the rule based on the priority if the condition evaluates to true. The scheduler is activated and the function Process_imm-rules waits until all the triggered rules in the immediate coupling mode complete its execution.

In order to process deferred rules, the primitive event Notify_def-rules is used and this triggers a pre-defined rule immediately and the action part of this rule calls the function Process_def-rules which indicate the scheduler to process deferred rules. The Notify_def-rules event waits for the completion of all deferred rules and a flag variable is used to block the parent transaction from commit until all the rules spawned complete its execution. When a rule thread completes its execution the corresponding rule is removed from the process-rule-list and at the end of the top-level transaction, the rule process-rule-list is empty. The transaction commits and the application continues further when the process-rule-list has to execute.

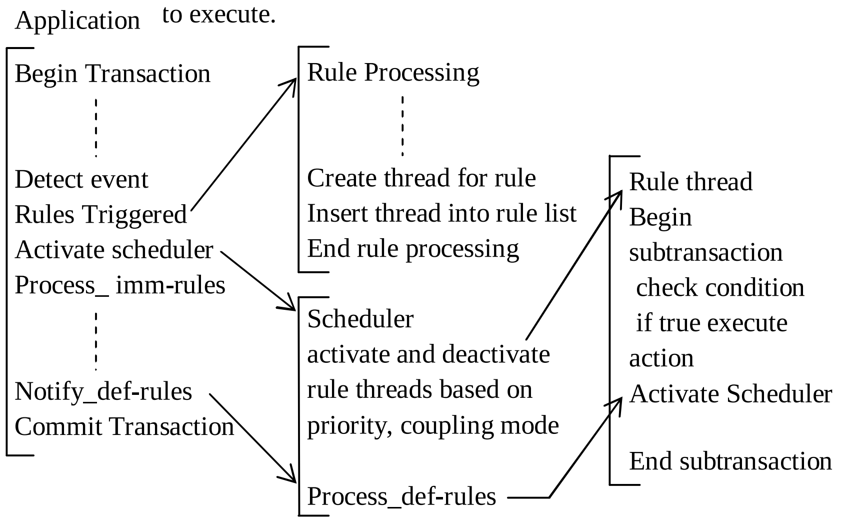


Fig. 3. Overview of thread implementation

The psuedocode for rule scheduling and execution using threads and how the thread function packaged is shown in Fig. 4. As detailed above, when a primitive event is signaled the event detector notifies the occurrence of the primitive events and determine the triggered rules for execution. Whenever a rule is triggered in immediate coupling mode, a thread is created with the thread-id and transforms the function which checks the condition and performs the action to a thread with the appropriate priority. Once all the immediate rules are in the form of threads, the main application is suspended and the rule scheduler is activated. After the execution of all triggered rules, the triggering transaction resumed execution. Since a deferred rule is executed only when Notify_def-rules primitive event is signaled and hence deferred rules treated in the same way as an immediate rule.

```
Initiate_thread( )
{
If(there is rule to be scheduled)
{
If (thread is available)
{
priority = assign_priority;
thread-id = get-thread( );
create_thread(thread-id,priority,cond_action);
}
}
}
//condition and action packaged as the body of the thread
cond_action(rule_cond, rule_action)
{
 void (cond)( ) =  rule_cond;
 void (action)( ) =  rule_action;
 sub= begin_subtransaction(current)
 if ((cond)( ) == true)
   (action)( );
 end_subtransaction(sub);
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 5, May 2014

```
}
```

Fig. 4. Psuedocode for rule scheduling and execution using threads

V. PERFORMANCE AND RESULT ANALYSIS

In advanced application systems, an event occurrence causes the triggering of multiple rules at the same time, the rule scheduler need to support sequential,concurrent and the both execution of rules based on user-defined priority scheme is a required research work. To evlauate the performance of the developed rule scheduler that supports both concurrent and sequential rule execution, the thread execution time for sequential, concurrent and both rule executions have been compared and is shown in Table 1. The values shown in Table 1 are plotted on bar chart (Fig. 5) and the results show that the developed scheduler takes less execution time for executing rules from 3 to 7 concurrently than the time taken for executing rules from 3 to 7 sequentially and both rule execution. Also from Table 1, it is found that the developed scheduler takes less execution time for executing rules from 3 to 7 in both rule executions than the time taken for executing rules only sequential.

Table 1 Comparison of sequential, concurrent and both rule execution

Number of conflicted rules	3	4	5	6	7
Sequential execution (Time in milliseconds)	122	166	210	257	303
Concurrent execution (Time in milliseconds)	44	58	75	91	104
Both concurrent and sequential (Time in milliseconds)	75	115	168	209	257

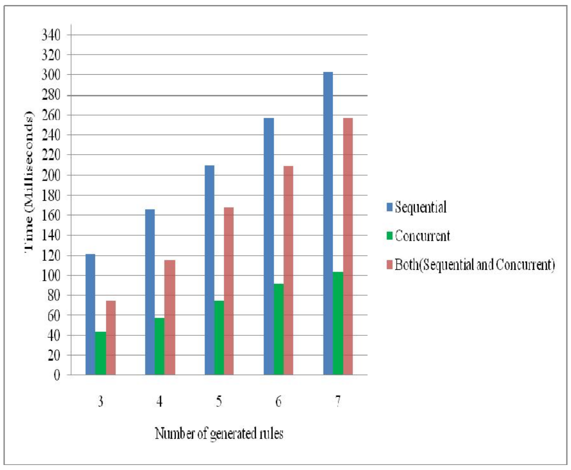


Fig. 5. Performance comparison of sequential, concurrent and both rule execution with rules from 3 to 7

The Fig. 5 reveals that the developed scheduler improves the system performance by reducing the over all execution time of rules by supporting concurrent and both rule execution than the conventional rule scheduler that supports only sequential rule execution for trigger processing in active ORDBMS. The developed rule scheduler prototype system provides the facility to define and maintain the rule system and supports both concurrent and sequential execution of rules based on user-defined prioity scheme for handling trigger rule conflicts, whenever a primitive event occurrence causes the triggering of mutiple rules at the same time in advanced applications.

VI. CONCLUSION

Active ORDBMS monitor the occurrences of events that defined over database state and react automatically by the pre-specified definition of ECA rules, when the state of the database changes relevant to the application system. Whenever a specified event occurrence causes the triggering of multiple rules at the same time, the rule scheduler is responsible for handling trigger rule conflicts that determines the order and the execution semantics of rules. In active ORDBMS standard, rule scheduler has been developed to execute the multiple triggered rules sequentially based on system defined priority. This research work develops an effective rule scheduling mechanism for active ORDBMS which supports the execution of multiple triggered rules both sequential and concurrent based on user-defined priority scheme. In this research work, the developed rule scheduler improves the performance of trigger processing by reducing the over all execution time of rules by supporting both concurrent and sequential rule execution than the rule scheduler that supports only sequential rule execution to handle trigger rule conflicts in active ordbms.

REFERENCES

1. A. Eisenberg, J. Melton, “SQL: 1999, formerly known as SQL3”, SIGMOD Record, Vol. 28, No.1, pp. 131-138, 1999.
2. N.W. Paton, O. Diaz, “Active database systems”, ACM Computing Surveys, Vol. 31, Issue 1, pp.63-103, 1999.
3. N. Kerdprasop, S. Pilabutr, K. Kerdprasop, “Improving Medical Database Consistency with Induced Trigger Rules”, New Advances in Intelligent Decision Technologies, Studies in Computational intelligence, Vol. 199, pp. 265–274, Springer-Verlag, 2009.
4. A.Rasoolzadegan, M.R. Meybodi, “Rule Scheduling in Active Database Using Learning Automata”, International Journal of Computer, Mathematical Sciences and Applications, Vol. 4, No. 1-2, pp. 239-262, ISSN: 0973-6786, Serials Publications, Jan-June 2010.
5. Y. Jin, S.D. Urban, S.W. Dietrich, “A concurrent rule scheduling algorithm for active rules”, Data & Knowledge Engineering, Elsevier, Vol. 60, Issue 3, pp.530-546, 2007.
6. Kulkarni, N. Mattos, R. Cochrane, “Active database features in sql-3”, In N. Paton (ed.), Active Rules in Database Systems, pp.197-219, Springer-Verlag, 1999.
7. K. Dube, B. Wu, “A generic approach to computer-based Clinical Practice Guideline management using the ECA Rule paradigm and active databases”, International Journal of Technology Management (IJTM), Vol.47, Nos. 1/2/3,pp.75-95, 2009.
8. M. Schaff, A. Koschel, S.G. Grivas, I. Astrova, “An Active DBMS Style Activity Service for Cloud Environments”, The International Conference on Cloud Computing, GRIDs, and Virtualization, pp.80-85, IARIA, ISBN: 978-1-61208-106-9, 2010.
9. Wai Yin Mok, C.F. Hickman, C.D. Allport, “Implementing Business Processes: A Database Trigger Approach”, International Journal of Knowledge-Based Organizations, Vol. 3, No.2, pp.36-55, Apr-June 2013.
10. S. Meenakshi, V. Thiagarasu, “Design of Rule Scheduler for Trigger Rule Conflict in Active Object-Relational Database Systems”, International Journal of Emerging Technologies in Computational and Applied Sciences, Vol. 7, No.2, pp.185-189, ISSN 2279-0055, February 2014.
11. F. Najafabadi, H. Reza, A.H. Navin, “Rule scheduling methods in active database systems: A brief survey”, In Proceedings of the 6th International Conference on Application of Information and Communication Technologies, IEEE, pp.1-5, 2012.
12. S. Wasserkrug, A. Gal, O. Etzion, Y. Turchin, “Efficient Processing of Uncertain Events in Rule-Based Systems”, IEEE Transactions on Knowledge and Data Engineering , Vol. 24, No.1, pp.45-58, 2012.

BIOGRAPHY

S. Meenakshi received M.C.A degree from University of Madras in the year 1990 and M.Phil degree from Bharathiyar University in 2001. She is having around 24 years of teaching experience in Gobi Arts & Science College, Gobichettipalayam. Her area of interest includes Object-Oriented Programming Systems, Advanced Database Systems and Data Mining.

V. Thiagarasu received his Master Degree in Mathematics from Gobi Arts & Science College, Gobichettipalayam in 1985, M.Phil and Ph.D in Computer Science from Bharathiar University, Coimbatore in 1996 and 2010 respectively. He is presently working as an Associate Professor in Computer Science, Gobi Arts & Science College since 1989. He has also completed a UGC sponsored minor research project during 2004. His current research centered on the Networking, Multi-agent system, Concurrent Engineering and Project Scheduling.