



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

Design of Rule Scheduler for Trigger Rule Conflict in Active Object-Relational Database Systems

Ms. S.Meenakshi¹, Dr. V. Thiagarasu²

¹Associate Professor in Computer Science, Gobi Arts & Science College, Gobichettipalayam, India

²Syndicate Member, Bharathiar University, Coimbatore & Associate Professor in Computer Science, Gobi Arts & Science College, Gobichettipalayam, India

Abstract: Active object-relational database systems (ORDBMS) extend the normal functionality of ORDBMS with support to monitor the changes to the database state and react to specific situations automatically without user intervention. The reactive behavior of active ORDBMS is represented by triggers in the form of Event-Condition-Action (ECA) rule paradigm. The behavior of triggers is based on the execution model and it specifies how trigger rules are executed at runtime. An important issue concerned with the development of rule execution model of an active ORDBMS is to process trigger rule conflict, when multiple rules are triggered at the same time. In order to address this issue, active ORDBMS provide a mechanism known as rule scheduling, which performs the ordering and execution semantics of multiple triggered rules. The focus of this work is to design and introduce a rule scheduling mechanism for active ORDBMS, which supports concurrent execution of rules with equal priority which allows also having sequential execution among rules with different priorities, whenever an event occurrence causes the triggering of multiple rules at the same time.

Keywords: Object-relational database systems, active database systems, triggers, ECA rules, rule conflicts, rule scheduling

I. INTRODUCTION

Database management systems are the vital part of current information system technology and they provide reliable and effective mechanisms for storing and managing large volumes of information in a multiuser environment. The limited modeling and manipulating capabilities of relational database systems and the increasing complexity of advanced database applications, recently object-relational database systems (ORDBMS) have been developed to extend relational database systems with the capabilities of supporting object-oriented programming features for satisfying the requirements of advanced database applications.

The field of database research is aimed at increasing the functionality and performance of advanced DBMS to accommodate the requirements of modern database applications. In the recent past, active database systems (ADBS) with active rules are considered to be one of the important features needed for integrating active mechanisms into database systems for supporting advanced applications that require monitoring changes to the database state and initiating appropriate actions automatically without user intervention. Integrating ADBS with object-relational database systems are commonly referred as active ORDBMS that support to monitor and react to specific situations relevance to an application. The object-relational database standard SQL3 and most of the today's commercial object-relational database products such as Oracle, Informix, MySQL and DB2 support active mechanism in the form of triggers [6].

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 syntax and specification of ECA rules for describing reactive behavior generally referred to as the knowledge model and also possess an execution model that determines the processing of rules at runtime [14]. 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 behavior of active ORDBMS depends on the rule execution model and hence rule execution semantics is an important aspect of active ORDBMS. The rule execution model of the active ORDBMS has been characterized by the various phases such as event detection, signaling of events, rule scheduling and rule execution.

In a generic active rule system, an event occurrence can cause the multiple rules to be triggered at the same time and hence the execution needs to be consistent, which is an important issue faced by the rule execution model of an active ORDBMS. In order to address this issue, an 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 [13]. In active ORDBMS standard, rule scheduling has been performed based on the priority with sequential execution of all rules [9, 13].

Whenever an event can trigger multiple rules that execute over distributed sources of data or multiple rules that are to be triggered by many different events at the same time for advanced applications, it is important to make use of rule scheduling with concurrent execution of all rules to speed up the ADBS rule execution process for improving the system performance in active ORDBMS. Rule scheduling has considerable impact on the performance and efficiency of ADBS [15]. To maintain the deterministic property of the active rule processing, the database management system has to provide rule scheduling [10]. Hence the focus of this work is to design and introduce a rule scheduling mechanism for active ORDBMS, which supports concurrent execution of rules with same priority which allows also having sequential execution among rules with different priorities, after detecting the occurrence of an event.

The rest of the paper is organized as follows. Section 2 briefly reviews the concept of triggers and their related issues in ORDBMS. Section 3 describes the proposed rule scheduling mechanism for handling rule conflict set in rule processing of an active ORDBMS and section 4 concludes the paper.

II. Triggers in Object-Relational Database Systems

The important functionality supported by the object-relational DBMS has been the support of active mechanisms represented by triggers in the form of Event-Condition-Action (ECA) rule paradigm that enhance the object-relational database with reactive capabilities. Triggers are the important component of current commercial database systems since active features are required for advanced applications to perform (react) certain operations automatically when specified events occur and particular conditions are met. The utility and functionality of active capability has been established in the context of object-relational database systems in the form of triggers [6]. The SQL3 standard has extensive support of triggers [6, 13]. An SQL trigger is a facility that allows database designers to instruct the database system to perform certain operations each and every time an application performs specified operations on particular tables.

The triggers are being used in a variety of significant ways in database systems and applications. The potential applications of triggers are significant [2]: signal integrity constraint violation, maintain consistency across system catalogs, gather statistics on table access provide auditing and prevent invalid transactions, automatic propagation of modifications and implement business rules or workflow management. The SQL3 standard captures active behavior in the form of triggers by using two kinds of constructs such as knowledge model and execution model.

Knowledge model

In SQL3, triggers are identified by name and expressed by means of ECA rules using the syntax as presented in Figure 1. A trigger is activated whenever a specified event occurs, and the event specifies the database operations usually an insert, delete or update operation on a particular table monitored by the trigger. The condition is expressed as an arbitrary SQL predicate involving complex queries. The action can be any sequence of database operations that include SQL manipulation statement, user defined functions and procedures. It is possible to specify whether the trigger must be executed BEFORE or AFTER its triggering event. The WHEN clause specifies an additional condition to be checked once the trigger rule is fired and before the action is executed. If the WHEN clause is missing, the condition is supposed to be true and the trigger action is executed as soon as the trigger event occurs. The action is executed when the rule is triggered and its condition is true.

```

<Trigger definition >::= CREATE TRIGGER <trigger-name>
                           {BEFORE|AFTER}<trigger-event> ON <table-name>
                           [REFERENCING {OLD [AS] <old-value-tuple-name>
                           |NEW [AS] <new-value-tuple-name>
                           |OLD_TABLE [AS] <old-value-table-name>
                           |NEW_TABLE [AS] <new-value-table-name>}]
<Triggered event >::=     INSERT|DELETE|UPDATE [OF <column-names>]
<Triggered action >::=    [FOR EACH {ROW |STATEMENT}]
                           [WHEN <condition>] <triggered SQL statement>
    
```

Figure 1 Definition of SQL triggers

Execution model

The behavior of triggers is based on the execution model and it specifies how triggers are evaluated and treated at runtime. The various phases that are performed during trigger execution are described as the following [13]:

- Event signaling – The signaling phase detects and signals the occurrence of the event.
- Rule triggering – The event activates the corresponding active rules in the triggering phase.
- Condition evaluation – The condition part of the triggered rules are evaluated in the evaluation phase. The rule conflict problem occurs when the conditions of more than one trigger rules are evaluated to be true.

- Rule scheduling – The scheduling phase indicates the order to process the rule conflict set when multiple rules are triggered at the same time.
- Execution phase – The execution phase processes the scheduled triggered rules.

The rule processing behavior can be quite complex and one of the important issue concerned with the development of rule execution model of an active ORDBMS is to process trigger rule conflict, when multiple rules triggered at the same time. In order to solve trigger rule conflict issue, active ORDBMS should provide a mechanism known as rule scheduling which determines the order and the execution semantics of multiple triggered rules caused by an event. By means of trigger scheduling, deterministic behavior of the trigger processing can be guaranteed [10]. Deterministic trigger processing guarantees the same order of execution when several trigger rules are activated simultaneously. The research works in ADBS integrated with relational and object-relational database systems standard support rule scheduling using static rule priorities with sequential execution of all rules [17, 7, 20, 3, 9, 15]. However an event occurrence causes multiple rules to be triggered at the same time over distributed sources of data for modern large scale applications [4, 5, 16], an enhanced rule scheduler is required for ordering and the execution of multiple triggered rules in active ORDBMS.

III. Proposed Rule Scheduling Mechanism

Active object-relational database systems can react to the occurrence of predefined events automatically by the definition of active rules. Whenever an event occurrence causes multiple rules to be triggered at the same time, scheduling mechanism determines their execution priority and the execution semantics: sequential or concurrent. The sequential rule execution enforces a single rule execution at a time and supports that rules have to be in a predefined sequence of execution order enforced by priorities. Also sequential approach does not use concurrency or maximize performance. However concurrent rule execution improves system performance and thereby reducing the overall execution time for rule transaction. Concurrent rule execution speeds up the ADBS rule execution process for improving the system performance [8]. Thus, the rule scheduler should support both sequential and concurrent rule execution based on the priority mechanism in active ORDBMS. When a rule is triggered, the time to start the execution of that rule is determined by the semantics of the coupling mode involved. Thus when multiple rules are triggered together, the order of execution among those rules can be determined by assigning priorities and rules are executed based on their coupling modes.

The integrated architectural design to entail active behavior into ORDBMS with rule processing module is shown in Figure 2. In this work, the major component extensions incorporated for modeling rule execution includes the transaction manager, lock manager and rule scheduler. Transaction manager ensures that transactions request and release locks according to a suitable locking protocol and schedules the execution of user transactions. Transaction manager is also responsible for supporting nested transactions for concurrent execution of rules. The lock manager which keeps the track of requests for locks and grants locks on database objects when they become available. Rule scheduler is responsible for handling rule conflicts when multiple rules are triggered at the same time by an event. The rule scheduler implements the multi-threaded mechanism for the prioritized concurrent rule execution. A rule scheduling mechanism has been designed for ordering rules based on static priority and the execution of rules concurrently using nested transaction model as well as sequentially for active ORDBMS. The nested transaction model[11] is considered as a suitable tool to implement rule execution in ADBS since it provides a good model for concurrent rule execution and it can handle multiple triggering of rules together with nested rule firing [1, 13]. Multiple firing of rule occurs, when an event causes more than one rule to be fired and nested rule firing occurs when the condition evaluation or action execution of a rule causes other rules to be fired. Thus, in this work nested transaction model is used for modeling concurrent rule execution and it can handle multiple triggering of rules that occurs when an event causes more than one rule to be fired. Concurrent execution of rules can also bring along conflict with respect to data access and the nested transaction synchronization scheme is used to handle the conflicts with locking protocol.

Functioning of Rule Scheduler

In active ORDBMS, an application defines rules and specifies the desired behavior of each rule in the form of ECA. In an application, the user specifies the rule definition associated with event, condition and action, priority and the coupling mode. If the priority and the coupling mode are not specified, the default priority and coupling mode has taken into consideration. Once a set of rules is defined, the active database system monitors occurrences of events pre-specified by ECA rules. Whenever a specified event occurrence can cause the multiple triggering of rules at the same time, the rule conflict occurs and thus rule scheduler determines the order and the execution semantics of multiple triggered rules caused by an event. In this work, the two important factors considered for designing the rule scheduler is the priority and coupling mode. The rule scheduler uses the user-defined priority information to execute the triggered rules sequentially with different priorities and the rules with the same priority are executed concurrently. The scheduler allows rules with the highest priority to execute first and after their completion, it allows the next rule with lower priority to execute. The coupling mode specifies the time when a triggered transaction is executed with respect to the triggering transaction or an

event. This work supports rule execution with immediate coupling mode and the algorithm for rule scheduling is shown in Table 1.

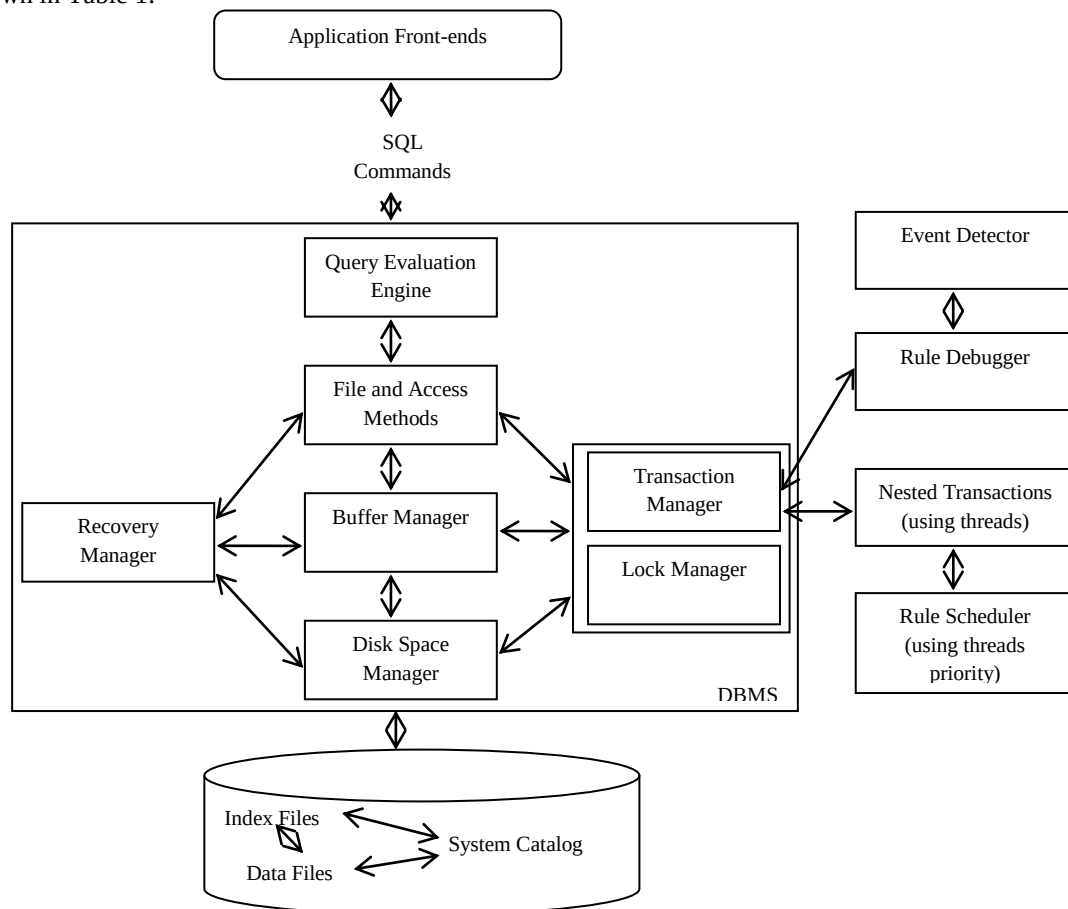


Figure 2 Integrated Architectural Design of Active ORDBMS

Table 1 Rule Scheduling Algorithm

- Step 1: Check whether all the rules $R_1, R_2, R_3 \dots R_n$ belongs to D , where D is a domain.
- Step 2: If there is only one rule R_i is raised by an event E , then print "No rule conflicts"; go to step 7. Otherwise go to next step.
- Step 3: If an event E triggers multiple rules $R_1, R_2, R_3 \dots R_n$ then print "Rules are conflict"
 $R_p = \{\text{empty}\}$ initially, form a process-rule-list $R_p = \{R_1, R_2, \dots, R_n\}$ go to next step.
- Step 4: Check the priority number to each rule in R_p . If the priorities to each rule is different, then print "Sequential scheduling"; go to step 6. Otherwise go to next step.
- Step 5: If the priority of the rules in R_p are equal then print "Concurrent scheduling"; execute all rules concurrently. Otherwise go to step 8.
- Step 6: Execute the rules in R_p sequentially one by one based on the highest priority until the R_p set is empty; go to step 9.
- Step 7: Check the condition of R_i ; If $R_i = \text{true}$ then print "ready for execution"; Otherwise go to step 9.
- Step 8: If the rules are not prioritized, then execute the rules in R_p sequentially one by one based on the system priority.
- Step 9: Exit rule processing and resume the transaction.

The functional features of the proposed rule scheduler that operates on multiple rules with nested transaction model, whenever an event triggering multiple rules at same time is demonstrated with the example ECA- rule definition as shown below.

- R1: ON update to emp(rank) IF new emp(rank) < 5 THEN update bonus(raise-amt)
 R2: ON update to emp(rank) IF TRUE THEN retrieve emp(empid,name,salary,rank)
 R3: ON update to bonus(raise-amt) IF TRUE THEN update emp(salary)
 R4: ON update to emp(salary) IF emp(salary) > 50000 THEN retrieve emp(empid, name, salary, rank)

Table 2 Process-rule-list

Rule List	R1	R3	R4	R2
Coupling Mode	1	1	1	1
Priority	7	4	5	7
Parent	1	R1	R1	1

Based on the example rule definition, suppose an event, update on employee rank is detected and it causes the triggering of rules R1 and R2 eligible to execute. Whenever an event triggered multiple rules, the proposed scheduler forms the process-rule-list based on the scheduling algorithm and the coupling mode. The top level rules that are triggered from the application R1 and R2 are placed in the process-rule-list with priorities and coupling mode. R1 and R2 have the same user-defined priority ((7) Ref. Table 2) and the coupling mode as immediate. Also R3, R4 have been spawned from the action execution of R1. R3 and R4 have been placed after R1 based on their priorities. Thus, if the rule has been triggered within the action of another rule, then the parent rule is searched for and the new rule is placed after the parent in the process-rule-list. The parent rule has to wait for the termination of all child rules before it can proceed since its coupling mode is immediate. Table 2 shows the process-rule-list for the above example and rules R1 and R2 are executed concurrently because of their equal priority and also for identifying the rules as top level rules by giving the value as 1. Hence in this work, the rule scheduler allows multiple rule execution associated with the same event to execute concurrently, if they have the same priorities. The rule scheduling design is the initial work toward the development and implementation of rule scheduler for improving the performance of scheduling mechanism in active ORDBMS.

IV. Conclusion

In recent years, there has been a growing need for active or event-driven systems that react automatically to events [19]. The event-driven system in the database has been impacted to manage and react to events in advanced applications. The research works in active database systems integrated with relational and object-relational database systems support rule scheduling using static priorities with sequential execution of all rules. This research work designs and proposes a rule scheduler that enhances the rule scheduling mechanism of active ORDBMS by supporting both sequential and concurrent rule execution based on user-defined priority scheme. Rule scheduler that supports concurrent rule execution improves system performance by reducing the overall execution time of rules than all rules executing in sequential. The research work is further extended to develop and implement the functionality of the rule scheduler using multi-threaded mechanism for the concurrent execution of rules in active ORDBMS.

V. References

- [1] A.P.Buchmann, "Architecture of Active Database Systems", In N.Paton (ed.) Active Rules in Database Systems, Chapter 2, pp.29-48, Springer, 1999.
- [2] S. Ceri, R.J. Cochrane, J. Widom, "Practical applications of triggers and constraints: Successes and lingering issues", In Proceedings of the 26th International Conference on Very Large Data Bases, pp. 254- 262, Cairo, Egypt, 2000.
- [3] R. Cochrane, H. Pirahesh, N. Mattos, "Integrating Triggers and Declarative Constraints in SQL Database Systems", In Proceedings of the 22nd VLDB Conference, Mumbai, India, 1996.
- [4] H. Ding, G. Trajcevski, P. Scheuermann, "Efficient Maintenance of Continuous Queries for Trajectories", Geoinformatica, Springer, 2007.
- [5] 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.
- [6] A. Eisenberg, J. Melton, "SQL: 1999, formerly known as SQL3", SIGMOD Record, Vol. 28, No.1, pp. 131-138, 1999.
- [7] E.N. Hanson, "The Design and Implementation of the Ariel Active Database Rule System", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, Issue 1, pp.157-153, 1996.
- [8] 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.
- [9] K. 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.
- [10] 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.
- [11] J.E.B. Moss, "Nested Transactions: An Approach to Reliable Distributed Computing", Massachusetts Institute of Technology Press, Cambridge, England, 1985.
- [12] 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.
- [13] N.W. Paton, O. Diaz, "Active database systems", ACM Computing Surveys, Vol. 31, Issue 1, pp.63-103, 1999.
- [14] N.W. Paton, J. Campin, A.A.A. Fernandes, M.H. Williams, "Formal Specification of Active Database Functionality: A Survey", In T. Sellis, editor, Proceedings of the 2nd International Workshop on Rules in Database Systems, pp.21-35, Springer-Verlag, 1995.
- [15] 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.
- [16] 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.
- [17] M. Stonebraker, G. Kemnitz, "The POSTGRES next-generation database management system", Communications of the ACM, Vol.34, No.10, pp.78-92, 1991.
- [18] 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.
- [19] S. Wasserkug, 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.
- [20] J.Widom, "The Starburst Active Database Rule System", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, Issue 4, pp.583-595, 1996.