CHAPTER 5

5. DEVELOPMENT OF PROTOTYPE

5.1 INTRODUCTION

Active object-relational database systems react to the occurrence of predefined events automatically by the definition of active rules. Since events trigger 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. In order to solve the trigger rule conflicts, the rule execution model of an active ORDBMS provides a mechanism known as rule scheduling, which performs the ordering and execution semantics of multiple triggered rules. Thus design and development of a rule scheduler is an important research work in active object-relational database systems.

Conventional rule schedulers have been developed to execute the multiple triggered rules sequentially based on system defined priority at rule creation time. However, the concurrent execution of multiple triggered rules is a required research work for advanced applications for improving the system performance. Thus the focus of this work is to design and develop a rule scheduler for executing multiple triggered rules sequential, concurrent and the combination of both based on user-defined priority at rule creation time.

Designing a scheduler to manage trigger rule conflicts involves the systematic transformation of description of the trigger rule processing into a number of tasks and coordinating the modules such as transaction manager, rule manager and rule scheduler. The focus of this work is about the rule ordering and rule execution of multiple triggered rules that are activated by an event. The ordering of rules is based on the user-defined priority provided by the user at rule definition time and rules are ordered based on the absolute numeric value. The flexibility of rule execution is guaranteed through the definition of various coupling modes that define the execution of rules. Thus the rule scheduler uses the priority and coupling mode given by the user in the rule definition for scheduling rules.
5.2 ACTIVE ORDBMS PROTOTYPE ARCHITECTURE

The aim of an active ORDBMS is to perform automatic monitoring of situations that defined over database state and the ability to take action when the state of the database changes (transaction-triggered processing) relevant to the application system. For this an event based rule definition system is used, in which a rule is triggered by an event such as insertion, deletion, modification or selection of data. The efficiency of an active ORDBMS depends on the event detection and rule scheduling and execution mechanisms after the event is raised. These mechanisms must be well integrated with the underlying DBMS and at the same time they should be implemented as a modular way which is compatible with any database system. Thus, the component architecture method provides the extensibility mechanism for rule processing in active ORDBMS. The component architecture provides the flexible architectural environment that allows the addition of new components without significantly modifying the existing ones.

The event-condition-action (ECA) rule paradigm is used to incorporate the active behavior into ORDBMS for modeling trigger rule processing. The major component extensions required for providing active capability includes the event detector, rule manager and rule scheduler related with transaction manager of the underlying ORDBMS as shown in Figure 5.1. Event detectors responsible for detecting primitive events and the rule manager retrieves the appropriate event from the database and determines the rules that are associated to the primitive event based on the information provided by the event detector.

Rule scheduler is an important component of the architecture that is developed to support multiple trigger rule execution. The rule scheduling mechanism which is presented is an implementation of trigger rule conflicts where priorities and coupling modes are the important factors for rule scheduling and execution. To provide an effective rule execution mechanism, a rule scheduler is developed to determine which set of rules is triggered by an event. Thus the rule scheduler is aware of the factors such as priority for selecting rules and coupling mode for the execution time of rules. Whenever an event is detected by an event detector, a notification is sent to the rule manager. Rule manager determines the conflict rules and calls the transaction
manager to create a transaction for the conflict triggered rules for scheduling and execution. Triggered rules are executed as a part of triggering transactions because of immediate and deferred coupling modes and to support concurrent rule execution using nested transaction model. The important extension of this architecture is the rule definition tool for defining the ECA rule system for applications that require trigger capabilities and the rule scheduler that is responsible for handling the multiple rule execution based on user defined priority scheme with sequential, concurrent and the combination of both executions of rules.

5.3 SOFTWARE PROTOTYPE SYSTEM

A software prototype system has been designed and developed to test validity of the proposed concepts and algorithm for rule scheduling component. The software prototype system provides a user interface system that includes two phases such as rule maintenance and rule processing as shown in Figure 5.2.
The rule maintenance phase is used to define and maintain the specification of ECA rules for applications that require automatic monitoring and reactions. The rule processing phase is used to show the functions of the rule scheduler that includes the ordering of rules based on priority and the execution of rules with sequential, concurrent and both. The prototype integrates the rule definition and the functions of the rule scheduler. The prototype has been developed using object-oriented language C# with ASP.NET as the front-end and SQL server as the back-end.

### 5.3.1 Rule maintenance

This component is used as a tool to define and maintain the active rules in the form of ECA for specifying the reactive behavior of a system which describes the situations to be monitored and the reactions of the system when these situations are encountered. By using this component the user specifies the rule definition associated with event, condition and action, priority and the coupling mode based on the requirement of an application system using the syntax as shown in Figure 5.3 and the description of the keywords are the following:
• Rule name – represents the name of the rule.

• Event – represents the set of modification operations on the database state (insert, update, delete).

• Condition – represents the query on the current database state.

• Action – represents the database operations on the current database state.

• Coupling mode – represents the time to execute the triggered rules, before or after the triggering rules.

• Priority – represents the priority value. The rule with the highest priority has higher precedence than the lower one.

```
Define Rule <Rule name>
On     <Event>
if     <Condition>
then   <Action>
Coupling mode = <Immediate| Deferred>
Priority   = <Number>
```

Figure 5.3 Syntax of an active rule

The user interface screen that is designed and developed for creating the rules is shown in Figure 5.4. Each rule in the system has been created with the details such as rule number, name of the rule, table name, event, condition, action, priority and coupling mode. A rule number is generated automatically for each rule and the rule name is a unique name that identifies the rule to the user. Event is the any one of the event such as insert, update or delete that triggers the rule. Coupling mode specifies when the trigger rule should be evaluated, i.e. before or after. Priority specifies the numeric absolute value that is used for ordering of rules, whenever a rule conflict occurs. The rule on field specifies the table name that is to be monitored for the event occurrence. The procedure specifies the corresponding condition is evaluated when the rule is triggered by a given event and the action is the name of the rule procedure to be invoked.
After providing the specification for rule definition, the save option is used to store the details under a unique rule name and rule number. The home and logout options are used to switch to the home screen and to exit to the system. This component is also used to maintain the rules with the facilities such as edit, view and delete. The rule definition of each rule can be modified at any time using edit option and the modified data is updated and stored in the same table with the same rule name. The screen that is designed and developed for changing the rule definition details using edit option is shown in Figure 5.5.

![Figure 5.4 User interface screen for rule creation](image1)

![Figure 5.5 User interface screen for rule edit](image2)

The view option is used to show the details of all rules that are created. This option provides an opportunity for the user to view the all created rule definitions as a whole and that are used to modify the rules dynamically at any time based on the requirements of the application system if required. The screen that is designed and developed for viewing the details of rule definitions is shown in figure 5.6. The delete option is used to delete the rule definitions that are not required and the screen that is designed and developed for delete option is shown in Figure 5.7. By using the rule maintenance component, the user can easily define and maintain the rule system based on the requirement of any application system that require automatic monitoring and respond automatically to the occurrence of database events in any commercial object-relational database system.
5.3.2 Rule processing

In order to show the function of the rule scheduler the rule processing component is developed to perform the tasks of the rule scheduler such as ordering of rules and the execution semantics of multiple triggered rules. After the specification of rule definition associated with event, condition, action, priority and coupling mode, rule definitions are stored in the knowledge base and the rule execution model of an active object-relational database management system monitors the occurrence of relevant events and takes the action automatically for applications that require active mechanism.

A rule is triggered when an event occurs that matches the specified event in the rule definition. Event detectors responsible for detecting the primitive events such as database operations or application defined events. After detecting the occurrence of an event, an event causes several rules to be eligible for execution. If several rules are eligible for execution rule conflict occurs and the rule scheduler component is activated for ordering and execution of multiple triggered rules raised by a primitive event to handle trigger rule conflicts.

The developed rule scheduler first collects the rules and forms a process-rule-list, which contains conflict triggered rules that are eligible for execution. In order to handle triggered rule conflicts, the rule scheduler first orders the rules by search through the rules that are triggered by an event. The result from the search of
the rules provides the list of ordered rules according to highest priority and coupling mode for execution. If the priorities of the rules in the process-rule-list are equal, then all rules are executed concurrently at a time based on the coupling mode before (immediate) or after (deferred). If the priorities of the rules are different then the rules are executed one by one by sequentially with the highest priority has higher precedence than the lower one based on the coupling mode.

Suppose that the rules in the process-rule-list have both equal and different priorities, then rules with equal priorities are executed concurrently first and then rules with different priorities are executed sequentially one by one based on the coupling mode. If the process-rule-list contains rules with both before and after coupling modes, the rules with the before coupling mode is executed first and then rules with after coupling mode is executed. The developed rule scheduler implements the multi-threaded mechanism for the concurrent and sequential execution of multiple triggered rules. Each rule triggered during the execution of an application has been created as a separate thread and the execution of the thread is controlled by the scheduler based on its priority and coupling mode. A multi-threading facility allows for concurrent operation with multiple simultaneous points of execution. The triggering transaction and the triggered transactions are all different threads of operation.

The screens that are designed and developed to show the function of the rule processing component includes the conflict triggered rules waiting for scheduling as shown in Figure 5.8 and the multiple rules that are scheduled based on priority and coupling mode for execution as shown in Figure 5.9. Figure 5.8 shows the list of conflict triggered rules raised by an event related to a table with the details such as rule number, rule name, rule on, priority and coupling mode. Figure 5.9 shows the scheduled rules based on priority and coupling mode for execution related to a table with the details such as rule number, rule name, rule on, priority, coupling mode and the rule execution mode.

Using a sequential schedule generating scheme, different priority values are used to construct an active schedule by executing each rule one-at-a-time immediately or later before the commit of the triggering transaction. Using a concurrent schedule generating scheme, equal priority values are used to construct an active schedule by
executing all rules at-a-time immediately or later before the commit of the triggering transaction. Using a hybrid (both sequential and concurrent) schedule generating scheme, both equal and different priority values are used to construct an active schedule by executing all rules at-a-time with equal priority first and then rules with different priorities are executed one by one immediately or later before the commit of the triggering transaction. Thus, the developed rule scheduler generates the different possible schedules based on the user-defined priority scheme and the coupling mode for the execution of multiple triggered rules and optimizes the rule execution in a better manner than the conventional rule scheduler with sequential scheduling based on system defined priority scheme.

Figure 5.8 Conflict triggered rules for scheduling  Figure 5.9 Scheduled rules for execution

5.4. SUMMARY

Active ORDBMS monitors the occurrences of events that are defined over the database state and react automatically by the pre-specified definition of rules, when the state of the database changes relevant to the application system. In order to incorporate reactive capabilities into an object-relational database system, various functional components are required which include rule definition system, event detector, rule manager and rule scheduler. Rule scheduler component is responsible for ordering and executing rules in a correct manner to handle multiple triggered rule conflicts according to an event.
In this chapter, a software prototype system has been designed and developed to test validity of the proposed concepts and algorithm for rule scheduling component. The prototype system has been developed in two phases such as rule maintenance for defining the rule system and rule processing for scheduling multiple triggered rules for execution. The two principle issues that are considered for developing a rule scheduler are: ordering of the next rule to be executed and the number of rules to be executed. The user-defined priority scheme is used for ordering the rules and the rules are executed with the options such as sequential, concurrent and both execution. Thus the developed rule scheduler effectively handles triggered rule conflicts.

The prototype is consistent with the object-oriented paradigm and is implemented using an object-oriented language C# which supports the component based approach for software development. The developed rule scheduler implements the multi-threaded mechanism for the concurrent execution of multiple triggered rules. A multi-threading facility allows for concurrent operation with multiple simultaneous points of execution. The software prototype system for rule scheduling component can be used with any active object-relational database system for ordering and executing multiple triggered rules both sequential and concurrent based on user-defined priority scheme and coupling mode.

In conventional rule scheduler, multiple triggered rules are executed sequentially with system-defined priority scheme. However in this research work, the developed rule scheduler improves the execution time of multiple triggered rules by supporting concurrent execution of rules and also the user has the facility to define and maintain the rule system for monitoring and reacting to the occurrence of events in advanced applications.