Enhanced Data Models for Advanced Applications

Active Database Concepts
Topics To Be Discussed

- Generalized Model for Active Databases and Oracle Triggers
- Design and Implementation Issues for Active Databases
- Potential Applications for Active Databases
Active Rules

- Active databases provide additional functionality for specifying active rules. These rules can be automatically triggered by events that occur, such as a database update or a certain time being reached, and can initiate certain actions that have been specified in the rule declaration if certain conditions are met.
Generalized Model for Active Databases

The model that has been used for specifying active database rules is referred to as the Event-Condition-Action, or ECA model.

A rule in the ECA model has three components:

1. The event
2. The condition
3. The action
ECA Model

1. The event (or events) that trigger the rule: These events are usually database update operations that are explicitly applied to the database.

2. The condition that determines whether the rule action should be executed: Once the triggering event has occurred, an optional condition may be evaluated. If no condition is specified, the action will be executed once the event occurs.
ECA Model (Contd...)

If a condition is specified, it is first evaluated, and only if it evaluates to true will the rule action be executed.

3. The action to be taken: The action is usually a sequence of SQL statements, but it could also be a database transaction or an external program that will be automatically executed.
Example

**EMPLOYEE**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SSN</th>
<th>SALARY</th>
<th>DNO</th>
<th>SUPERVISOR_SSN</th>
</tr>
</thead>
</table>

**DEPARTMENT**

<table>
<thead>
<tr>
<th>DNAME</th>
<th>DNO</th>
<th>TOTAL_SAL</th>
<th>MANAGER_SSN</th>
</tr>
</thead>
</table>

Note:
- SUPERVISOR_SSN is a (recursive) foreign key to EMPLOYEE.
- We assume that null is allowed for DNO, indicating that an employee may be temporarily unassigned.
- MANAGER_SSN is a foreign key to EMPLOYEE.
- TOTAL_SAL attribute is a derived attribute, whose value should be the sum of the salaries of all employees who are assigned to the particular department.
Example (Contd...)

• We first have to determine the events that may cause a change in the value of TOTAL_SAL, which are as follows:
  
  • 1. Inserting (one or more) new employee tuples.
  
  • 2. Changing the salary of (one or more) existing employees.
  
  • 3. Changing the assignment of existing employees from one department to another.
Example (Contd...)  

**Condition** for events:

- **Event 1:** we only need to recompute TOTAL_SAL if the new employee is immediately assigned to a department—that is, if the value of the DNO attribute for the new employee tuple is not null (assuming null is allowed for DNO)
Example (Contd...)

- **Events 2 and 4:** we need to check whether the employee whose salary is changed (or who is being deleted) is currently assigned to a department

- **Event 3:** we will always execute an action to maintain the value of TOTAL_SAL correctly, so no condition is needed
Example (Contd...)

- **The action** for events 1, 2, and 4 is to automatically update the value of TOTAL_SAL for the employee’s department to reflect the newly inserted, updated, or deleted employee’s salary. In the case of event 3, a twofold action is needed; one to update the TOTAL_SAL of the employee’s old department and the other to update the TOTAL_SAL of the employee’s new department.
Example (Contd...)

Syntax in Oracle:

R1: CREATE TRIGGER TOTALSAL1
    AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    WHEN (NEW.DNO IS NOT NULL)
        UPDATE DEPARTMENT
        SET TOTAL_SAL=TOTAL_SAL + NEW.SALARY
        WHERE DNO=NEW.DNO;

R2: CREATE TRIGGER TOTALSAL2
    AFTER UPDATE OF SALARY ON EMPLOYEE
    FOR EACH ROW
    WHEN (NEW.DNO IS NOT NULL)
        UPDATE DEPARTMENT
        SET TOTAL_SAL=TOTAL_SAL + NEW.SALARY - OLD.SALARY
        WHERE DNO=NEW.DNO;

R3: CREATE TRIGGER TOTALSAL3
    AFTER UPDATE OF DNO ON EMPLOYEE
    FOR EACH ROW
    BEGIN
        UPDATE DEPARTMENT
        SET TOTAL_SAL=TOTAL_SAL + NEW.SALARY
        WHERE DNO=NEW.DNO;
        UPDATE DEPARTMENT
        SET TOTAL_SAL=TOTAL_SAL-OLD.SALARY
        WHERE DNO=OLD.DNO;
    END;

R4: CREATE TRIGGER TOTALSAL4
    AFTER DELETE ON EMPLOYEE
    FOR EACH ROW
    WHEN (OLD.DNO IS NOT NULL)
        UPDATE DEPARTMENT
        SET TOTAL_SAL=TOTAL_SAL-OLD.SALARY
        WHERE DNO=OLD.DNO;
Example (Contd...) 

Keywords used in creation of a Trigger:

- The CREATE TRIGGER statement specifies a trigger (or active rule) name—TOTALSAL1 for R1.
- The AFTER-clause specifies that the rule will be triggered after the events that trigger the rule occur.
- The ON-clause specifies the relation on which the rule is specified—EMPLOYEE for R1.
• It is important to note the effect of the optional FOR EACH ROW clause, which signifies that the rule is triggered separately for each tuple. This is known as a row-level trigger. If this clause was left out, the trigger would be known as a statement-level trigger and would be triggered once for each triggering statement.

• Note that the keywords NEW and OLD can only be used with row-level triggers.
Design and Implementation Issues for Active Databases

**Issue 1:** activation, deactivation, and grouping of rules.

- **A deactivated rule** will not be triggered by the triggering event. This feature allows users to selectively deactivate rules for certain periods of time when they are not needed.

- **The activate command** will make the rule active again.
Design and Implementation Issues for Active Databases (Contd…)

- Rules can be **Grouped** into named rule sets, so the whole set of rules could be activated, deactivated, or dropped.

- It is also useful to have a command that can trigger a rule or rule set via an explicit PROCESS RULES command issued by the user.
Design and Implementation Issues for Active Databases (Contd...)

**Issue 2:** whether the triggered action should be executed before, after, or concurrently with the triggering event.

A related issue is whether the action being executed should be considered as a separate transaction or whether it should be part of the same transaction that triggered the rule.
Design and Implementation Issues for Active Databases (Contd...)

Assuming that the triggering event occurs as part of a transaction execution, following are the various options for how the triggering event is related to the evaluation of the rule’s condition. The rule condition evaluation is also known as rule consideration, since the action is to be executed only after considering whether the condition evaluates to true or false.
Design and Implementation Issues for Active Databases (Contd...)

There are three main possibilities for rule consideration:

1. Immediate consideration: The condition is evaluated as part of the same transaction as the triggering event, and is evaluated immediately. This case can be further categorized into three options:
   - Evaluate the condition before executing the triggering event.
   - Evaluate the condition after executing the triggering event.
   - Evaluate the condition instead of executing the triggering event.

2. Deferred consideration: The condition is evaluated at the end of the transaction that included the triggering event. In this case, there could be many triggered rules waiting to have their conditions evaluated.

3. Detached consideration: The condition is evaluated as a separate transaction, spawned from the triggering transaction.
The next set of options concern the relationship between evaluating the rule condition and executing the rule action. Here, again, three options are possible: immediate, deferred, and detached execution. However, most active systems use the first option. That is, as soon as the condition is evaluated, if it returns true, the action is immediately executed.
Design and Implementation Issues for Active Databases (Contd...)

**Issue 3:** Another issue concerning active database rules is the distinction between row-level rules versus statement-level rules. Because SQL update statements (which act as triggering events) can specify a set of tuples, one has to distinguish between whether the rule should be considered once for the whole statement or whether it should be considered separately for each row (that is, tuple) affected by the statement.
Design and Implementation Issues for Active Databases (Contd...)  

Drawbacks of Active Rules:  

• there are no easy-to-use techniques for designing, writing, and verifying rules. For example, it is quite difficult to verify that a set of rules is consistent, meaning that two or more rules in the set do not contradict one another.  

• It is also difficult to guarantee termination of a set of rules under all circumstances.  

• It is also difficult to guarantee termination of a set of rules under all circumstances.
Example Illustrating non-termination of rules

R1:  
CREATE TRIGGER T1
    AFTER INSERT ON TABLE1
    FOR EACH ROW
    UPDATE TABLE2
    SET ATTRIBUTE1=...;

R2:  
CREATE TRIGGER T2
    AFTER UPDATE OF ATTRIBUTE1 ON TABLE2
    FOR EACH ROW
    INSERT INTO TABLE1 VALUES (...);

Here, rule R1 is triggered by an INSERT event on TABLE1 and its action includes an update event on ATTRIBUTE1 of TABLE2. However, rule R2’s triggering event is an UPDATE event on ATTRIBUTE1 of TABLE2, and its action includes an INSERT event on TABLE1. It is easy to see in this example that these two rules can trigger one another indefinitely, leading to nontermination.
Potential Applications for Active Databases

1. to allow notification of certain conditions that occur. For example, an active database may be used to monitor, say, the temperature of an industrial furnace

2. to maintain replicated tables consistent by specifying rules that modify the replicas whenever the master table is modified.
Potential Applications for Active Databases (Contd...)

3. automatic maintenance of derived data

4. to enforce integrity constraints by specifying the types of events that may cause the constraints to be violated and then evaluating appropriate conditions that check whether the constraints are actually violated by the event or not. For example, in the UNIVERSITY database application, one rule may monitor the grade point average of students whenever a new grade is entered, and it may alert the advisor if the GPA of a student falls below a certain threshold.