Triggers & Active Databases

Chapter 7

Topics

• Introduction
  • Triggers in SQL Standard
  • Triggers in Oracle

What are Active DBMS?

• Traditionally procedures are executed when explicitly invoked by an application
• But some applications must react to various external events
  – E.g., an application that monitors physical objects, such as a power plant. If a sensor reports an elevated level of carbon monoxide, the ventilation system must be turned on
• If such reactive applications are data intensive, the DBMS should support reaction to external events
• Active DBMS: a DBMS that can react to external events

Trigger Overview

• A mechanism to code reactive capability
• Element of the database schema
• General form:
  \[
  \text{ON <event> IF <condition> THEN <action>}
  \]
  – Event - request to execute database operation
  – Condition - predicate evaluated on database state
  – Action – execution of procedure that might involve database updates
• Example:
  \[
  \text{ON updating maximum course enrollment}
  \text{IF number registered > new max enrollment limit}
  \text{THEN deregister students using LIFO policy}
  \]
Activation & Consideration

- **Activation** - Occurrence of the event
  - Event is a request to execute database operation
- **Consideration** - The point, after activation, when condition is evaluated
  - Condition might refer to both the state before and the state after event occurs
  - Condition checking can be:
    - Immediate, or
    - Deferred: when the transaction requests to commit

Execution

- Point at which action occurs
  - With deferred consideration, execution is also deferred
  - With immediate consideration, execution can occur immediately after consideration or it can be deferred
    - If execution is immediate, execution can occur before, after, or instead of triggering event
    - Before triggers adapt naturally to maintaining integrity constraints: violation results in rejection of event

Trigger Details

Trigger Granularity

- **Row level Granularity**
  - Change of a single row is an event
  - E.g., a single UPDATE statement might result in multiple events
- **Statement-level granularity**:
  - Events are statements
  - E.g., a single UPDATE statement that changes multiple rows is a single event
Activation of Multiple Triggers

- How should multiple triggers activated by a single event be handled?
  - Evaluate one condition at a time and if true immediately execute action or
  - Evaluate all conditions, then execute actions

- The choice is significant
  - The execution of an action can affect the truth of a subsequently evaluated condition

Cascading Triggers

- The execution of one trigger might activate another trigger, and so on
  - Sometimes called a chain reaction
  - Such an execution might be non-terminating

- Actual DBMS impose an upper limit on the length of cascading triggers
- In general, excessive use of triggers can result in complex interdependencies
  - Can be difficult to maintain in a large application

Topics

- Introduction
- Triggers in SQL Standard
- Triggers in Oracle

Triggers in SQL:1999

- Events:
  - INSERT, DELETE, or UPDATE statements or changes to individual rows caused by these statements
- Condition:
  - Anything allowed in a WHERE clause
- Action:
  - An individual SQL statement or a program written in the language of Procedural Stored Modules (PSM)
    (which can contain embedded SQL statements)
Triggers in SQL:1999

- **Consideration**: Immediate
  - Condition can refer to both the state of the affected row or table before and after the event occurs
- **Execution**: Immediate
  - Can be before or after the execution of triggering event
  - Action of before trigger cannot modify the database
- **Granularity**: Both row-level and statement-level granularity

### Before Trigger Example (row granularity)

```sql
CREATE TRIGGER Max_EnrollCheck
BEFORE INSERT ON Transcript
REFERENCING NEW AS N --row to be added
FOR EACH ROW
WHEN ((SELECT COUNT (T.StudId)
    FROM Transcript T
    WHERE T.CrsCode = N.CrsCode
    AND T.Semester = N.Semester)
  >= (SELECT C.MaxEnroll
    FROM Course C
    WHERE C.CrsCode = N.CrsCode))
ABORT TRANSACTION
```

Check that enrollment ≤ limit

### After Trigger Example (row granularity)

```sql
CREATE TRIGGER LimitSalaryRaise
AFTER UPDATE OF Salary ON Employee
REFERENCING OLD AS O
NEW AS N
FOR EACH ROW
WHEN (N.Salary - O.Salary > 0.05 * O.Salary)
UPDATE Employee
SET Salary = 1.05 * O.Salary
WHERE Id = O.Id
```

No salary raises greater than 5%

Note: The action itself is a triggering event (but in this case a chain reaction is not possible)

### After Trigger Example (statement granularity)

```sql
CREATE TRIGGER RecordNewAverage
AFTER UPDATE OF Salary ON Employee
FOR EACH STATEMENT
INSERT INTO Log
VALUES (CURRENT_DATE,
    SELECT AVG (Salary)
    FROM Employee)
```

Keep track of salary average in the log
Topics

- Introduction
- Triggers in SQL Standard
- Triggers in Oracle

Trigger Syntax

CREATE [OR REPLACE] TRIGGER
<trigger_name>
{BEFORE|AFTER} {INSERT|DELETE|UPDATE}
on <table_name>
[REFERENCING [NEW AS <new_row_name>]
[OLD AS <old_row_name>]]
[FOR EACH ROW
    [WHEN (<trigger_condition>)]]
<trigger_body> -- can be PL/SQL procedure

Example

CREATE TRIGGER activity_logging
    AFTER INSERT ON emp
    REFERENCING NEW AS newRow
    FOR EACH ROW
    WHEN (newRow.dept_id <=10)
    BEGIN
    INSERT INTO activity_log VALUES(SYSDATE,
        :newRow.emp_id, :newRow.dept_id);
    END activity_logging;
/

Alternative syntax: Don’t include REFERENCING NEW.
Can access new row by using :new and old row by
using :old

Triggering Events

- In Oracle a triggering event can be:
  1. INSERT, UPDATE, or DELETE on a specific table (or view in some cases)
  2. A CREATE, ALTER, or DROP on any schema object
  3. Database startup or shutdown
  4. A specific error message
  5. A user logon or logoff
When Not to Use Triggers

- Use integrity constraints where possible
- So don’t use a trigger if you can enforce a constraint using:
  - NOT NULL, UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
  - DELETE CASCADE
  - DELETE SET NULL

Hints on Use

- Row-level vs. Statement-level
  - If the code in the trigger action depends on the data in the affected rows, use row-level.
  - Otherwise use statement-level triggers
    - Cannot refer to row-level data in statement-level triggers
- BEFORE vs. AFTER
  - A common use of BEFORE triggers is when the trigger action determines whether the triggering event should be allowed to happen
  - E.g., to enforce inclusion dependency

Example: Inclusion Dependency

```
DROP TABLE PROF CASCADE CONSTRAINTS;
CREATE TABLE PROF(
  ID INTEGER PRIMARY KEY,
  AREA CHAR(10));
DROP TABLE COURSE CASCADE CONSTRAINTS;
CREATE TABLE COURSE(
  CNUM INTEGER PRIMARY KEY,
  CNAME CHAR(20) NOT NULL,
  AREA CHAR(10));
INSERT INTO PROF VALUES(1, 'database');
INSERT INTO PROF VALUES(2, 'compilers');
-- Constraint to enforce: COURSE.AREA must refer to PROF.AREA
```

Example (cont’d)

```
CREATE OR REPLACE TRIGGER course_expertise
BEFORE INSERT ON COURSE
FOR EACH ROW
DECLARE
  prof_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO prof_count
  FROM prof
  WHERE area = :new.area;
  IF(prof_count < 1) THEN
    raise_application_error(-20001, 'No prof with this expertise');
  END IF;
END;
```
Oracle’s Execution Model

• Constraint checking is also done in conjunction with trigger execution
  – Algorithm is recursive
1. Execute all BEFORE statement triggers that apply to the statement
2. Loop for each row affected by the SQL statement
   a. Run all BEFORE row triggers
   b. Lock and change row and perform integrity constraint checking
   c. Run all AFTER row triggers
3. Run all AFTER statement triggers

Restrictions

• An important restriction if not observed will result in a ‘mutating table error’ when the trigger executes
• A "mutating table"
  – either the table whose modification activated the trigger, or
  – a table that might need to be updated because of a foreign key constraint with a CASCADE policy
• To avoid mutating table errors:
  – A row-level trigger must not query or modify a mutating table
    • Of course, NEW and OLD still can be accessed by the trigger
  – A statement-level trigger must not query or modify a mutating table if the trigger is fired as the result of a CASCADE delete