CSE 532 – Theory of Database Systems

Lecture 13 (Chapter 7)
Triggers and Active Databases

Ilchul Yoon
Assistant Professor
State University of New York, Korea

Trigger Overview

- Element of the database schema
- General form:
  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:
  ON updating maximum course enrollment
  IF number registered > new max enrollment limit
  THEN deregister students using LIFO policy
Trigger Details

- **Activation** - Occurrence of the event

- **Consideration** - The point, after activation, when condition is evaluated
  - Immediate or deferred (when the transaction requests to commit)
  - Condition might refer to both the state before and the state after event occurs

---

Trigger Details

- **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

**Granularity**
- *Row-level granularity:* change of a single row is an event (a single UPDATE statement might result in multiple events)
- *Statement-level granularity:* events are statements (a single UPDATE statement that changes multiple rows is a single event).

**Multiple Triggers**
- How should multiple triggers activated by a single event be handled?
  - *Ordered conflict resolution:* Evaluate one condition at a time and if true immediately execute action or
  - *Group conflict resolution:* Evaluate all conditions, then schedule action execution
- The execution of an action can affect the truth of a subsequently evaluated condition so the choice is significant.
Triggers in SQL:1999

- **Events**
  - INSERT, DELETE, or UPDATE statements or changes to individual rows caused by these statements

- **Condition**
  - Anything that is 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)

**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 the triggering event
- Action of before trigger cannot modify the database

**Granularity**: Both row-level and statement-level
Trigger Syntax

```sql
CREATE TRIGGER name
   {Before | After | Instead of }
   {INSERT | DELETE | UPDATE [ OF column-list ]}
   ON table-name

   [referencing [ OLD AS old.tuple.var ]
   [ NEW AS new.tuple.var ]
   [ OLD TABLE AS old.table.var ]
   [ NEW TABLE AS new.table.var ]]

   [ FOR EACH ROW ]
   [ WHEN (precondition) ]
   action
```

Trigger in Major DBMSs

- **Postgres >**
  - Expressiveness/behavior = full standard
    - row-level + statement-level, old/new row & table
  - Cumbersome & awkward syntax

- **SQLite >>**
  - Row-level only, immediate activation
    - no old/new table

- **MySQL**
  - Row-level only, immediate activation
  - no old/new table
  - Only one trigger per event type
  - Limited trigger chaining
Before Trigger Example (row granularity)

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)

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 -- action
  SET Salary = 1.05 * O.Salary
  WHERE Id = O.Id

salary raise cap is 5%

Note: The action itself is a triggering event (but in this case a chain reaction is not possible)
After Trigger Example (statement granularity)

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

CURRENT_DATE is a built-in SQL function that returns the current date

Trigger to Maintain Inclusion Dependencies

CREATE TRIGGER NoEmptyCourses
AFTER DELETE, UPDATE OF CrsCode, Semester ON Transcript
FOR EACH STATEMENT
DELETE FROM Teaching
WHERE EXISTS ( SELECT *
FROM IdleTeaching T
WHERE Semester = T.Semester AND CrsCode = T.CrsCode

IdleTeaching is a view that includes tuples from the Teaching relation that describes course offerings with no corresponding tuples in the Transcript relation

ASSERTION is not supported by major DBMS vendors
Instead-of Trigger Example

- Not in SQL standard

```sql
CREATE VIEW WorksIn (ProfId, DeptName) AS
    SELECT P.Id, D.Name
    FROM Professor P, Department D
    WHERE P.DeptId = D.DeptId
```

```sql
DELETE FROM WorksIn
WHERE Id = 1111111
```

```sql
CREATE TRIGGER WorksInTrigger
    INSTEAD OF DELETE ON WorksIn
    REFERENCING OLD AS O
    FOR EACH ROW
    UPDATE Professor
    SET DeptId = NULL
    WHERE Id = O.ProfId
```

Recall the example used in the (Updating View) slide (Chapter 5)

Summary of Trigger Evaluation Procedure

- Assume an event \( e \) activated triggers \( T = \{ T_1, \ldots, T_k \} \) during the execution of a database update statement \( S \).
  1. Put triggers to Trigger Queue, \( Q \)
  2. Suspend the execution of \( S \)
  3. Compute OLD and NEW if row-level, or OLD TABLE and NEW TABLE if statement-level
  4. Consider all BEFORE triggers in \( T \), execute with true preconditions, and place all AFTER triggers with true preconditions on \( Q \)
  5. Apply the updates specified in \( S \)
  6. Consider each AFTER trigger on \( Q \) in an order, and execute if precondition is true. If the execution activates another triggers, recursively apply this algorithm.
  7. Resume the execution of statement \( S \)
Avoiding Chain Reaction

- Possibility of never-ending chain reaction
  - Impose an upper limit on the length of chain reactions
  - Trigger systems that terminate anyway are called safe
    - But, no algorithm to tell the safety of a given set of triggers

- A sufficient condition for trigger safety
  - Triggering graph: nodes are triggers, an edge from trigger T to T’ if and only if execution of T is an event that can activate T’
  - Very simple. Only allow if the graph is acyclic
  - However, this condition rejects many safe trigger systems
    - e.g., LimitSalaryRaise (cyclic, but executed only once)

---

Trigger Chain Example - Self Trigger

```sql
CREATE TRIGGER R1
AFTER INSERT ON T1
FOR EACH ROW
BEGIN
  INSERT INTO T1 values (New. A+1);
END;

INSERT INTO T1 values (1);
```

For this example,
pragma recursive_triggers = off

This example is for SQLite
Trigger Chain Example - Self Trigger

CREATE TRIGGER R1
AFTER INSERT ON T1
FOR EACH ROW
WHEN (SELECT COUNT(*) FROM T1) < 10
BEGIN
    INSERT INTO T1 values (New.A+1);
END;

INSERT INTO T1 values (1);

For this example,
pragma recursive_triggers = on

Trigger Chain Example

CREATE TRIGGER R1
AFTER INSERT ON T1
FOR EACH ROW
BEGIN
    INSERT INTO T2 values (New.A+1);
END;

CREATE TRIGGER R2
AFTER INSERT ON T2
FOR EACH ROW
BEGIN
    INSERT INTO T3 values (New.A+1);
END;

INSERT INTO T1 values (1);

CREATE TRIGGER R3
AFTER INSERT ON T3
FOR EACH ROW
WHEN (SELECT COUNT(*) FROM T1) < 100
BEGIN
    INSERT INTO T1 values (New.A+1);
END;

For this example,
pragma recursive_triggers = on

T1: 1, 4, 7, ..., 298 (100 tuples)
T2: 2, 5, ...
T3: 3, 6, ...
Trigger Chain Example

CREATE TRIGGER R1
AFTER INSERT ON T1
FOR EACH ROW
BEGIN
    INSERT INTO T2 values (New.A + 1);
END;

CREATE TRIGGER R2
AFTER INSERT ON T2
FOR EACH ROW
BEGIN
    INSERT INTO T3 values (New.A + 1);
END;

CREATE TRIGGER R3
AFTER INSERT ON T3
FOR EACH ROW
BEGIN
    INSERT INTO T1 values (New.A + 1);
END;

INSERT INTO T1 values (1);

This example is for SQLite

For this example, 
pragma recursive_triggers = off

T1: 1, 4
T2: 2
T3: 3

Trigger Chain Example – Order?

CREATE TRIGGER R1
AFTER INSERT ON T1
FOR EACH ROW
BEGIN
    INSERT INTO T1 values A = 2;
END;

CREATE TRIGGER R2
AFTER INSERT ON T1
FOR EACH ROW
WHEN (SELECT * FROM T1 WHERE A = 2);
BEGIN
    INSERT INTO T1 values A = 3;
END;

INSERT INTO T1 values (1);

This example is for SQLite
Trigger Chain Example – Nested

```sql
CREATE TRIGGER R1
AFTER INSERT ON T1
FOR EACH ROW
BEGIN
    INSERT INTO T2 values (1);
    INSERT INTO T3 values (1);
END;

CREATE TRIGGER R2
AFTER INSERT ON T2
FOR EACH ROW
BEGIN
    INSERT INTO T3 values (2);
    INSERT INTO T4 values (2);
END;

CREATE TRIGGER R3
AFTER INSERT ON T3
FOR EACH ROW
BEGIN
    INSERT INTO T4 values (3);
END;
```

This example is for SQLite

---

Trigger Chain Example – Nested

```sql
CREATE TRIGGER R1
AFTER INSERT ON T1
FOR EACH ROW
BEGIN
    INSERT INTO T2 SELECT AVG(A) FROM T1;
END;

Originally, T1 had 4 tuples each with A=1
INSERT INTO T1 SELECT A+1 FROM T1;
```

This example is for SQLite

---

```
T1: 0
T2: 1.2, 1.33333, 1.428…, 1.5
T3: 2 1
T4: 3 2 3
```

```
T1: four 1s and 4 2s
T2: 1.2, 1.33333, 1.428…, 1.5
```