Integrity - Triggers

Overview

- What are Triggers?
- Procedures Vs Triggers
- Trigger Issues
- When to use Triggers?
- ECA Model
- Syntax
- Row vs Statement Triggers
- Before Vs After Triggers

What are Triggers?

- Triggers are action that executes (*fire*) automatically whenever some specified event occurs.
- Trigger is a PL/SQL block or PL/SQL procedure associated with a table, view, schema, or the database.
- Can be either:
  - Application trigger: fires when an event occurs with a particular application
  - Database trigger: fires when a data event (DML) or system event (logon or shutdown) occurs on schema or database.

Procedures Vs Triggers

- Procedures and triggers differ in the way that they are invoked:
  - A procedure is explicitly executed by a user, application, or trigger.
  - Triggers are implicitly fired when a triggering event occurs, no matter which user is connected or which application is being used.
- Note that the database stores triggers separately from their associated tables.

Trigger Issues

- Triggers are not recommended way to implement integrity constraints.
  - Declarative constraints are checked on all relevant updates, whereas triggers are invoked only when the specified event occurs.
- Trigger T1 could cause trigger T2 to fire, which could cause trigger T3 and so on (a *trigger chain*).
- Trigger T might even cause itself to fire again (*recursive*).

When to use Triggers?

- Some useful purpose of triggers:
  - Alterting the user if some exception occurs
  - Debugging (e.g., monitoring references to, and/or state changes in, designated variables)
  - Auditing (e.g., tracking who performed what updates to which relations when)
  - Performance measurement (e.g., timing or tracking specified database events)
  - Carrying out compensating actions.

ECA Model

- Trigger specifies, an *event*, a *condition*, and an *action*:
  - The **event** is an operation on the database (any update operations).
  - The **condition** is a boolean expression that has to evaluate to TRUE in order for the action to be executed.
    (if no condition, then the default is just TRUE)
  - The **action** is the triggered procedure.
- The event and condition together are called the *triggering event*
- Triggers are also known as *event-condition-action rules* [ECA]

ECA Model

- Events include INSERT, DELETE, UPDATE, end-of-transaction, reaching specified time-of-day, violating a specified constraint, and so on.
- The action can be performed BEFORE, AFTER, or INSTEAD OF the specified event.
- The action can be performed FOR EACH ROW or FOR EACH STATEMENT.
- A database that has associated triggers is sometimes called an *active database.*

Syntax

- CREATE [OR REPLACE] TRIGGER trigg_name
- [AFTER | BEFORE] [INSERT | UPDATE | DELETE]
- ON table_name [FOR EACH ROW]
- [WHEN (CONDITION)]
  - BEGIN
    - ... .
    - PL/SQL Block
    - ... ...
  - END;

The event

The condition

The action

Trigger Types

- Row Triggers and Statement Triggers
- BEFORE and AFTER Triggers
- INSTEAD OF Triggers
- Triggers on System Events and User Events

Row Vs Statement Trigger

- A **row trigger** is fired each time the table is affected by the triggering statement.
- For example, if an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement.
- The FOR EACH ROW option determines whether the trigger is a *row* trigger or a *statement* trigger.

Row Vs Statement Trigger

- A **statement trigger** is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects, even if no rows are affected.
- The absence of the FOR EACH ROW option indicates that the trigger fires only once for each applicable statement, but not separately for each row affected by the statement.

Using Predicates

- More than one type of DML operation can fire a trigger:

  Ex: ON INSERT OR DELETE OR UPDATE OF salary

- The trigger body can use the conditional predicates INSERTING, DELETING, and UPDATING to check which type of statement fire the trigger.

  Ex:

  IF INSERTING THEN .... END IF;

  IF DELETING THEN .... END IF;

  IF UPDATING ('SAL') THEN ... ... END IF;

Column values in Row triggers

- The PL/SQL code and SQL statements have access to the old and new column values of the current row affected by the triggering statement.
- The new column values are referenced using the *new* qualifier while the old column values are referenced using the *old* qualifier before the column name.
- Example:

  IF :new.sal > 50000 ....
- IF :new.sal < :old.sal ....

Before Vs After Triggers

- When defining a trigger, you can specify the trigger timing – whether the trigger action is to be executed before or after the triggering statement.
- BEFORE and AFTER apply to both statement and row triggers.
- BEFORE – For row triggers, the trigger is fired before each affected row is changed.
- AFTER – For row triggers, the trigger is fired after each affected row is changed.

Before Vs After Triggers

- Restrictions on AFTER Triggers:
  - You cannot specify an AFTER trigger on a view or an object view.

    You cannot write either the :OLD or the :NEW value.
- Restrictions on BEFORE Triggers:
  - You cannot specify a BEFORE trigger on a view or an object view.
  - You can write to the :NEW value but not to the :OLD value.

References

- An introduction to database systems, *CJ. Date*
- *www.oracle.com*