



Database Programming with PL/SQL

13-2

Creating DML Triggers: Part I



Objectives

This lesson covers the following objectives:

- Create a DML trigger
- List the DML trigger components
- Create a statement-level trigger
- Describe the trigger firing sequence options

Purpose

- Suppose you want to keep an automatic record of the history of changes to employees' salaries.
- This is not only important for business reasons, but is a legal requirement in many countries.
- To do this, you create a DML trigger.
- DML triggers are the most common type of trigger in most Oracle databases.
- In this and the next lesson, you learn how to create and use database DML triggers.



What Is a DML Trigger?

- A DML trigger is a trigger that is automatically fired (executed) whenever an SQL DML statement (INSERT, UPDATE, or DELETE) is executed.
- You classify DML triggers in two ways:
 - By when they execute: BEFORE, AFTER, or INSTEAD OF the triggering DML statement.
 - By how many times they execute: Once for the whole DML statement (a statement trigger), or once for each row affected by the DML statement (a row trigger).

Creating DML Statement Triggers

- The sections of a CREATE TRIGGER statement that need to be considered before creating a trigger:

```
CREATE [OR REPLACE] TRIGGER trigger_name  
    timing  
    event1 [OR event2 OR event3] ON object_name  
    trigger_body
```

- timing: When the trigger fires in relation to the triggering event.
 - Values are BEFORE, AFTER, or INSTEAD OF.
- event: Which DML operation causes the trigger to fire. Values are INSERT, UPDATE [OF column], and DELETE.

Creating DML Statement Triggers

```
CREATE [OR REPLACE] TRIGGER trigger_name  
    timing  
    event1 [OR event2 OR event3] ON object_name  
    trigger_body
```

- *object_name*: The table or view associated with the trigger.
- *trigger_body*: The action(s) performed by the trigger are defined in an anonymous block.



Statement Trigger Timing

When should the trigger fire?

- **BEFORE:** Execute the trigger body before the triggering DML event on a table.
- **AFTER:** Execute the trigger body after the triggering DML event on a table.
- **INSTEAD OF:** Execute the trigger body instead of the triggering DML event on a view.
- Programming requirements will dictate which one will be used.

Trigger Timings and Events Examples

- The first trigger executes immediately before an employee's salary is updated:

```
CREATE OR REPLACE TRIGGER sal_upd_trigg  
BEFORE UPDATE OF salary ON employees  
BEGIN ... END;
```

- The second trigger executes immediately after an employee is deleted:

```
CREATE OR REPLACE TRIGGER emp_del_trigg  
AFTER DELETE ON employees  
BEGIN ... END;
```

Trigger Timings and Events Examples

- You can restrict an UPDATE trigger to updates of a specific column or columns:

```
CREATE OR REPLACE TRIGGER sal_upd_trigg  
BEFORE UPDATE OF salary, commission_pct ON employees  
BEGIN ... END;
```

- A trigger can have more than one triggering event:

```
CREATE OR REPLACE TRIGGER emp_del_trigg  
AFTER INSERT OR DELETE OR UPDATE ON employees  
BEGIN ... END;
```

How Often Does a Statement Trigger Fire?

A statement trigger:

- Fires only once for each execution of the triggering statement (even if no rows are affected)
- Is the default type of DML trigger
- Fires once even if no rows are affected
- Useful if the trigger body does not need to process column values from affected rows

```
CREATE OR REPLACE TRIGGER log_emp_changes
AFTER UPDATE ON employees BEGIN
    INSERT INTO log_emp_table (who, when)
    VALUES (USER, SYSDATE);
END;
```

How Often Does a Statement Trigger Fire?

- Now an UPDATE statement is executed:

```
UPDATE employees  
SET ... WHERE ...;
```

- How many times does the trigger fire, if the UPDATE statement modifies three rows?
- Ten rows?
- One row?
- No rows?



And When Does the Statement Trigger Fire?

- This slide shows the firing sequence for a statement trigger associated with the event INSERT INTO departments:

```
INSERT INTO departments  
  (department_id, department_name, location_id)  
VALUES (400, 'CONSULTING', 2500);
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
400	CONSULTING	2500

→ **BEFORE statement trigger**

→ **AFTER statement trigger**

Trigger-Firing Sequence

- A statement trigger fires only once even if the triggering DML statement affects many rows:

```
UPDATE employees
  SET salary = salary * 1.1
  WHERE department_id = 50;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
124	Mourgos	50
141	Rajs	50
142	Davies	50
143	Matos	50

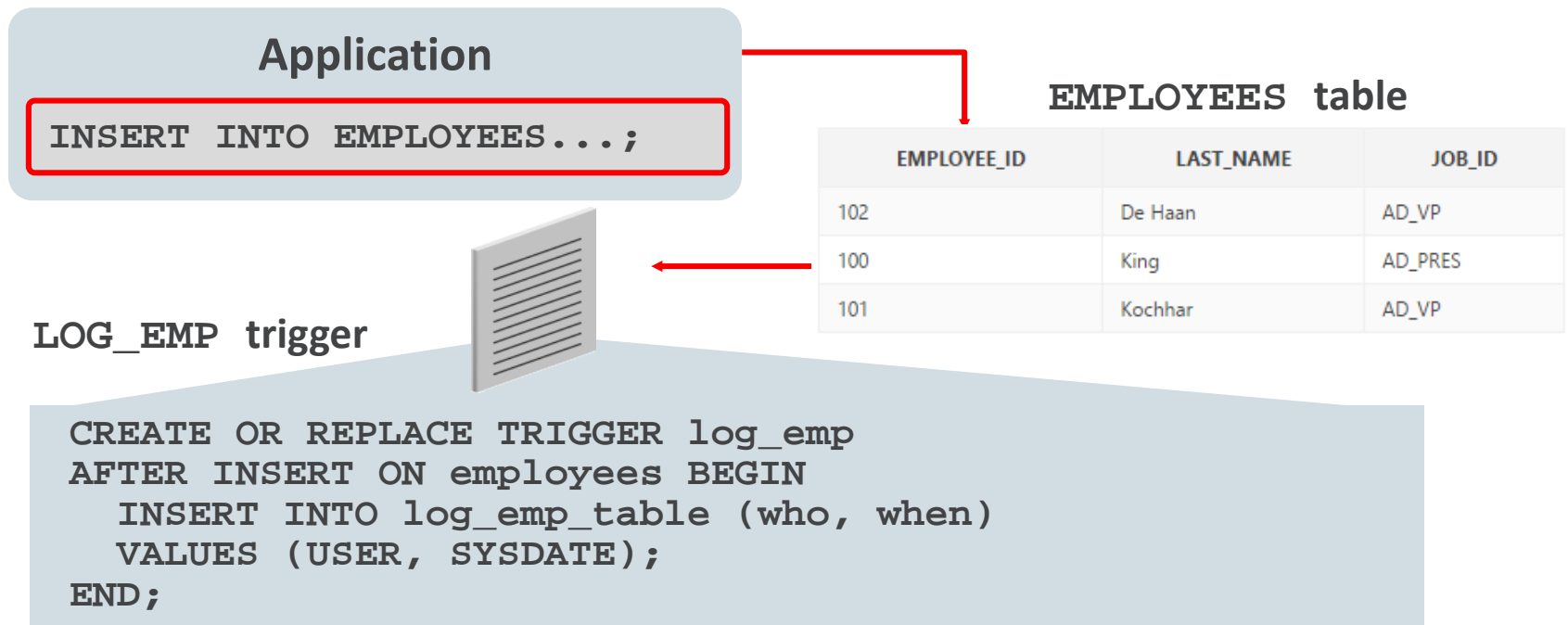
→ **BEFORE statement trigger**

144	Vargas	50
-----	--------	----

→ **AFTER statement trigger**

DML Statement Triggers Example 1

- This statement trigger automatically inserts a row into a logging table every time one or more rows are successfully inserted into EMPLOYEES.



DML Statement Triggers Example 2

- This statement trigger automatically inserts a row into a logging table every time a DML operation is successfully executed on the DEPARTMENTS table.

```
CREATE OR REPLACE TRIGGER log_dept_changes
AFTER INSERT OR UPDATE OR DELETE ON DEPARTMENTS
BEGIN
    INSERT INTO log_dept_table (which_user, when_done)
    VALUES (USER, SYSDATE);
END;
```


DML Statement Triggers Example 3

- This example shows how you can use a DML trigger to enforce complex business rules that cannot be enforced by a constraint.
- You want to allow INSERTs into the EMPLOYEES table during normal working days (Monday through Friday), but prevent INSERTs on the weekend (Saturday and Sunday).



DML Statement Triggers Example

- If a user attempts to insert a row into the EMPLOYEES table during the weekend, then the user sees an error message, the trigger fails, and the triggering statement is rolled back.
- The next slide shows the trigger code needed for this example.



DML Statement Triggers: Example 3

Application

```
INSERT INTO EMPLOYEES...;
```

EMPLOYEES table

EMPLOYEE_ID	LAST_NAME	JOB_ID
102	De Haan	AD_VP
100	King	AD_PRES
101	Kochhar	AD_VP

SECURE_EMP trigger

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON employees
BEGIN
  IF TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN') THEN
    RAISE_APPLICATION_ERROR(-20500,
      'You may insert into EMPLOYEES table only during
      business hours');
  END IF;
END;
```

Testing SECURE_EMP

- A user tries to INSERT a row on the weekend:

```
INSERT INTO employees (employee_id, last_name, first_name,  
    email, hire_date, job_id, salary, department_id)  
VALUES (300, 'Smith', 'Rob', 'RSMITH', SYSDATE, 'IT_PROG',  
    4500, 60);
```

```
ORA-20500: You may insert into EMPLOYEES table only  
    during business hours.  
ORA-06512: at "USVA_TEST_SQL01_T01.SECURE_EMP", line 4  
ORA_04088: error during execution of trigger  
'USVA_TEST_SQL01_T01.SECURE_EMP' 2. VALUES (300,  
'Smith', 'Rob', 'RSMITH', SYSDATE, 'IT_PROG', 4500, 60);
```

A Final Example

- This trigger does not compile successfully.
- Why not?

```
CREATE OR REPLACE TRIGGER log_dept_changes
AFTER INSERT OR UPDATE OR DELETE ON DEPARTMENTS
BEGIN
    INSERT INTO log_dept_table (which_user, when_done)
    VALUES (USER, SYSDATE);
    COMMIT;
END;
```



Terminology

Key terms used in this lesson included:

- DML trigger
- Row trigger
- Statement trigger

Summary

In this lesson, you should have learned how to:

- Create a DML trigger
- List the DML trigger components
- Create a statement-level trigger
- Describe the trigger firing sequence options

