ORACLE®

Academy

Database Programming with PL/SQL

13-1
Introduction to Triggers

ORACLE
Academy

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

This lesson covers the following objectives:

- Describe database triggers and their uses
- Define a database trigger
- Recognize the difference between a database trigger and an application trigger
- List two or more guidelines for using triggers
- Compare and contrast database triggers and stored procedures

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Purpose

- In this lesson, you learn about a database trigger.
- Triggers allow specified actions to be performed automatically within the database, without having to write extra application code.
- Triggers increase the power of the database, and the power of your application.
- You will learn more about triggers in the following lessons.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Need For A Trigger

- Let’s start with an example: A business rule states that whenever an employee’s salary is changed, the change must be recorded in a logging table.
- You could create two procedures to do this: UPD_EMP_SAL to update the salary, and LOG_SAL_CHANGE to insert the row into the logging table.
- You could invoke LOG_SAL_CHANGE from within UPD_EMP_SAL, or invoke LOG_SAL_CHANGE separately from the calling environment.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Need For A Trigger

- But you do not have to do this.
- Instead, you create a trigger.
- The next slide shows how.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Example of a Simple Trigger

- From now on, whenever an SQL statement updates a salary, this trigger executes automatically, inserting the row into the logging table.

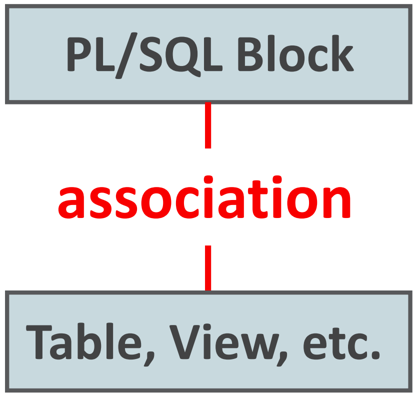
```
CREATE OR REPLACE TRIGGER log_sal_change_trigg
AFTER UPDATE OF salary ON employees
BEGIN
  INSERT INTO log_table (user_id, logon_date)
  VALUES (USER, SYSDATE);
END;
```

- This means the trigger automatically fires (that is, executes) whenever the triggering event (updating a salary) occurs.
- Cause and effect: The event occurs, and the trigger fires.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

What Is a Trigger?

- A database trigger:
- Is a PL/SQL block associated with a specific action (an event) such as a successful logon by a user, or an action taken on a database object such as a table or view
- Executes automatically whenever the associated action occurs
- Is stored in the database
- In the example on the previous slide, the trigger is associated with this action: UPDATE OF salary ON employees



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

What Is a Trigger?

- Triggers allow specified actions to be performed automatically in the database without having to write any extra front-end application code.
- The key difference between procedures/functions and triggers is that procedures/functions have to be invoked explicitly
- Triggers are blocks of code that are always "listening" for something to happen in the database.
- Also, since triggers are never explicitly invoked, they cannot receive parameters.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Database Triggers Compared to Application Triggers

- Database triggers execute automatically whenever a data event (such as DML or DDL) or a system event (such as a user connecting or the DBA shutting down the database) occurs on a schema or database.
- Database triggers are created and stored in the database just like PL/SQL procedures, functions, and packages.
- Application triggers execute whenever a particular event occurs within an application.
- They may lead to a database event, but they are not part of the database.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Which Events Can Cause a Database Trigger to Fire?

The following events in the database can cause a trigger to fire:

- DML operations on a table
- DML operations on a view, with an INSTEAD OF trigger
- DDL statements, such as CREATE and ALTER
- Database system events, such as when a user logs on or the DBA shuts down the database

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Types of Triggers

- Triggers can be either row-level or statement-level.
- A row-level trigger fires once for each row affected by the triggering statement
- A statement-level trigger fires once for the whole statement.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Possible Uses for Triggers

You can use triggers to:

- Enhance complex database security rules
- Create auditing records automatically
- Enforce complex data integrity rules
- Create logging records automatically
- Prevent tables from being accidentally dropped
- Prevent invalid DML transactions from occurring

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Possible Uses for Triggers

You can use triggers to:

- Generate derived column values automatically
- Maintain synchronous table replication
- Gather statistics on table access
- Modify table data when DML statements are issued against views

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Example 1: Creating Logging Records Automatically

- The Database Administrator wants to keep an automatic record (in a database table) of who logs onto the database, and when.
- He/she could create the log table and a suitable trigger as follows:

```
CREATE TABLE log_table (
  user_id    VARCHAR2(30),
  logon_date  DATE);

CREATE OR REPLACE TRIGGER logon_trigg
AFTER LOGON ON DATABASE
BEGIN
  INSERT INTO log_table (user_id, logon_date)
  VALUES (USER, SYSDATE);
END;
```

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Example 2: Enforcing Complex Data Integrity Rules

- Imagine a business rule that states no employee’s job can be changed to a job that the employee has already done in the past.

```
CREATE OR REPLACE TRIGGER check_sal_trigg
BEFORE UPDATE OF job_id ON employees
FOR EACH ROW
DECLARE
  v_job_count   INTEGER;
BEGIN
  SELECT COUNT(*) INTO v_job_count
  FROM job_history
  WHERE employee_id = :OLD.employee_id
  AND job_id = :NEW.job_id;
  IF v_job_count > 0 THEN
  RAISE_APPLICATION_ERROR
  (-20201,'This employee has already done this job');
  END IF;
END;
```

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Guidelines for Triggers

- Do not define triggers to duplicate or replace actions you can do easily in other ways.
- For example, implement simple data integrity rules using constraints, not triggers.
- Excessive use of triggers can result in slower processing and complex interdependencies, which can be difficult to maintain.
- Use triggers only when necessary and be aware of recursive (trigger that calls itself) and cascading effects.
- Avoid lengthy trigger logic by creating stored procedures or packaged procedures that are invoked in the trigger body.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Comparison of Database Triggers and Stored Procedures

Triggers	Procedures
Defined with `CREATE TRIGGER`	Defined with `CREATE PROCEDURE`
Data Dictionary contains source code in `USER_TRIGGERS`	Data Dictionary contains source code in `USER_SOURCE`
Implicitly invoked	Explicitly invoked
`COMMIT`, `SAVEPOINT`, and `ROLLBACK` are not allowed	`COMMIT`, `SAVEPOINT`, and `ROLLBACK` are allowed

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Terminology

Key terms used in this lesson included:

- Application triggers
- Database triggers
- Triggers

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe database triggers and their uses
- Define a database trigger
- Recognize the difference between a database trigger and an application trigger
- List two or more guidelines for using triggers
- Compare and contrast database triggers and stored procedures

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

ORACLE®
Academy