

11

Applications for Triggers

Objectives

After completing this lesson, you should be able to do the following:

- Create additional database triggers
- Explain the rules governing triggers
- Implement triggers

Creating Database Triggers

- Triggering a user event:
 - CREATE, ALTER, or DROP
 - Logging on or off
- Triggering database or system event:
 - Shutting down or starting up the database
 - A specific error (or any error) being raised

Creating Triggers on DDL Statements

Syntax:

```
CREATE [OR REPLACE] TRIGGER trigger_name  
Timing  
[ddl_event1 [OR ddl_event2 OR ...]]  
ON {DATABASE|SCHEMA}  
trigger_body
```

Creating Triggers on System Events

Syntax:

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing
[database_event1 [OR database_event2 OR ...]]
ON {DATABASE|SCHEMA}
trigger_body
```

LOGON and LOGOFF Triggers: Example

```
CREATE OR REPLACE TRIGGER logon_trig
AFTER LOGON ON SCHEMA
BEGIN
    INSERT INTO log_trig_table(user_id,log_date,action)
    VALUES (USER, SYSDATE, 'Logging on');
END;
/
```

```
CREATE OR REPLACE TRIGGER logoff_trig
BEFORE LOGOFF ON SCHEMA
BEGIN
    INSERT INTO log_trig_table(user_id,log_date,action)
    VALUES (USER, SYSDATE, 'Logging off');
END;
/
```

CALL Statements

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing
event1 [OR event2 OR event3]
ON table_name
[REFERENCING OLD AS old | NEW AS new]
[FOR EACH ROW]
[WHEN condition]
CALL procedure_name
/
```

```
CREATE OR REPLACE TRIGGER log_employee
BEFORE INSERT ON EMPLOYEES
CALL log_execution
/
```

Note: There is no semicolon at the end of the CALL statement.

Reading Data from a Mutating Table

```
UPDATE employees  
  SET salary = 3400  
  WHERE last_name = 'Stiles';
```

EMPLOYEES table

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|-------------|-------------|----------|--------|
| 125 | Nayer | ST_CLERK | 3200 |
| 126 | Mikkilineni | ST_CLERK | 2700 |
| 127 | Landry | ST_CLERK | 2400 |
| 128 | Markle | ST_CLERK | 2200 |

...

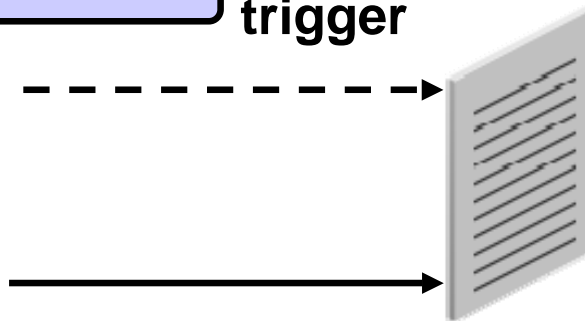
| | | | |
|-----|--------|----------|------|
| 138 | Stiles | ST_CLERK | 3400 |
|-----|--------|----------|------|

...

**Triggered table or
mutating table**

Failure

**CHECK_SALARY
trigger**



BEFORE UPDATE row

Trigger event

Mutating Table: Example

```
CREATE OR REPLACE TRIGGER check_salary
  BEFORE INSERT OR UPDATE OF salary, job_id
  ON employees
  FOR EACH ROW
  WHEN (NEW.job_id <> 'AD_PRES')
DECLARE
  minsalary employees.salary%TYPE;
  maxsalary employees.salary%TYPE;
BEGIN
  SELECT MIN(salary), MAX(salary)
    INTO minsalary, maxsalary
   FROM employees
   WHERE job_id = :NEW.job_id;
  IF :NEW.salary < minsalary OR
     :NEW.salary > maxsalary THEN
    RAISE_APPLICATION_ERROR(-20505, 'Out of range');
  END IF;
END;
/
```

Mutating Table: Example

```
UPDATE employees
  SET salary = 3400
  WHERE last_name = 'Stiles';
```

Error report:

| | |
|--|--|
| SQL Error: ORA-04091: table TEACH_D.EMPLOYEES is mutating, trigger/function may not see it ORA-06512: at "TEACH_D.CHECK_SALARY", line 5 ORA-04088: error during execution of trigger 'TEACH_D.CHECK_SALARY' 04091. 00000 - "table %s.%s is mutating, trigger/function may not see it" | |
|--|--|

Benefits of Database Triggers

- Improved data security provides enhanced and complex:
 - Security checks
 - Auditing
- Improved data integrity:
 - Enforces dynamic data integrity constraints
 - Enforces complex referential integrity constraints
 - Ensures that related operations are performed together implicitly

Managing Triggers

The following system privileges are required to manage triggers:

- The `CREATE/ALTER/DROP (ANY) TRIGGER` privilege that enables you to create a trigger in any schema
- The `ADMINISTER DATABASE TRIGGER` privilege that enables you to create a trigger on `DATABASE`
- The `EXECUTE` privilege (if your trigger refers to any objects that are not in your schema)

Note: Statements in the trigger body use the privileges of the trigger owner, not the privileges of the user executing the operation that fires the trigger.

Business Application Scenarios for Implementing Triggers

You can use triggers for:

- Security
- Auditing
- Data integrity
- Referential integrity
- Table replication
- Computing derived data automatically
- Event logging

Note: Appendix C covers each of these examples in more detail.

Viewing Trigger Information

You can view the following trigger information:

- USER_OBJECTS data dictionary view: Object information
- USER_TRIGGERS data dictionary view: Text of the trigger
- USER_ERRORS data dictionary view: PL/SQL syntax errors (compilation errors) of the trigger

Using USER_TRIGGERS

| Column | Column Description |
|-------------------|---------------------------------------|
| TRIGGER_NAME | Name of the trigger |
| TRIGGER_TYPE | The type is BEFORE, AFTER, INSTEAD OF |
| TRIGGERING_EVENT | The DML operation firing the trigger |
| TABLE_NAME | Name of the database table |
| REFERENCING_NAMES | Name used for :OLD and :NEW |
| WHEN_CLAUSE | The when_clause used |
| STATUS | The status of the trigger |
| TRIGGER_BODY | The action to take |

* **Abridged column list**

Listing the Code of Triggers

```
SELECT trigger_name, trigger_type, triggering_event,  
       table_name, referencing_names,  
       status, trigger_body  
FROM   user_triggers  
WHERE  trigger_name = 'RESTRICT_SALARY';
```

| TRIGGER_NAME | TRIGGER_TYPE | TRIGGERING_E... | TABLE... | REFERENCING_NAMES | STA... | TRIGGER_BODY |
|-------------------|-----------------|------------------|-----------|-----------------------------------|---------|---|
| 1 RESTRICT_SALARY | BEFORE EACH ROW | INSERT OR UPDATE | EMPLOYEES | REFERENCING NEW AS NEW OLD AS OLD | ENABLED | BEGIN IF NOT (:NEW.job_id IN ('AD_PRES', 'AD_ |

Summary

In this lesson, you should have learned how to:

- Use advanced database triggers
- List mutating and constraining rules for triggers
- Describe real-world applications of triggers
- Manage triggers
- View trigger information

Practice 11: Overview

This practice covers the following topics:

- Creating advanced triggers to manage data integrity rules
- Creating triggers that cause a mutating table exception
- Creating triggers that use package state to solve the mutating table problem