

PLSQL

Lesson 04 :Triggers and its types

Lesson Objectives

- To understand the following topics:
 - Describe database triggers and their uses
 - Describe the different types of triggers
 - Create database triggers
 - Describe database trigger-firing rules
 - Remove database triggers
 - Display trigger information



4.1: Triggers

Triggers

- An event which leads to action
 - Types of Triggers
 - Application : triggers when an event occurs in application
 - Database
- Database triggers are stored procedures that are implicitly executed when an triggering event occurs
- The triggering event could be (Database triggers):
 - DML statements on the table
 - DDL statements
 - System events such as startup, shutdown, and error messages
 - User events such as logon and logoff

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 3

4.1: Triggers

Application and Database Triggers

- Database trigger (covered in this course):
 - Fires whenever a DML, a DLL, or system event occurs on a schema or database
- Application trigger:
 - Fires whenever an event occurs within a particular application



Application Trigger Database Trigger

 Capgemini

Copyright © Capgemini 2015. All Rights Reserved. 4

Types of Triggers

Application triggers execute implicitly whenever a particular data manipulation language (DML) event occurs within an application. An example of an application that uses triggers extensively is an application developed with Oracle Forms Developer.

Database triggers execute implicitly when any of the following events occur:

- DML operations on a table
- DML operations on a view, with an INSTEAD OF trigger
- DDL statements, such as CREATE and ALTER

This is the case no matter which user is connected or which application is used. Database triggers also execute implicitly when some user actions or database system actions occur (for example, when a user logs on or the DBA shuts down the database).

Database triggers can be system triggers on a database or a schema (covered in the next lesson). For databases, triggers fire for each event for all users; for a schema, they fire for each event for that specific user. Oracle Forms can define, store, and run triggers of a different sort. However, do not confuse Oracle Forms triggers with the triggers discussed in this lesson.

4.1: Triggers

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

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 5

Business Application Scenarios for Implementing Triggers

Develop database triggers in order to enhance features that cannot otherwise be implemented by the Oracle server or as alternatives to those provided by the Oracle server.

Security: The Oracle server allows table access to users or roles. Triggers allow table access according to data values.

Auditing: The Oracle server tracks data operations on tables. Triggers track values for data operations on tables.

Data integrity: The Oracle server enforces integrity constraints. Triggers implement complex integrity rules.

Referential integrity: The Oracle server enforces standard referential integrity rules. Triggers implement nonstandard functionality.

Table replication: The Oracle server copies tables asynchronously into snapshots. Triggers copy tables synchronously into replicas.

Derived data: The Oracle server computes derived data values manually. Triggers compute derived data values automatically.

Event logging: The Oracle server logs events explicitly. Triggers log events transparently.

4.1: Triggers

Available Trigger Types

- Simple DML triggers
 - BEFORE
 - AFTER
 - INSTEAD OF
- Compound triggers
 - Non-DML triggers
 - DDL event triggers
 - Database event triggers



Copyright © Capgemini 2015. All Rights Reserved 6

Note

In this lesson, we will discuss the BEFORE, AFTER, and INSTEAD OF triggers. The other trigger types are discussed in the lesson titled “Creating Compound, DDL, and Event Database Triggers.”

4.1: Triggers

Parts of a Trigger

- Triggering event or statement
- Trigger restriction
- Trigger action

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 7

4.1: Triggers

Triggering Event or statement

- A triggering event or statement is the SQL statement, database event, or user event that causes a trigger to fire.
- A triggering event can be one or more of the following:
- An INSERT, UPDATE, or DELETE statement on a specific table/view
- A CREATE, ALTER, or DROP statement on any schema object
- A database startup or instance shutdown
- A specific error message or any error message
- A user logon or logoff



Copyright © Capgemini 2015. All Rights Reserved

8

4.1: Triggers

Trigger Event Types and Body

- A trigger event type determines which DML statement causes the trigger to execute. The possible events are:
 - INSERT
 - UPDATE [OF column]
 - DELETE
- A trigger body determines what action is performed and is a PL/SQL block or a CALL to a procedure.

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 9

Triggering Event Types

The triggering event or statement can be an INSERT, UPDATE, or DELETE statement on a table.

When the triggering event is an UPDATE statement, you can include a column list to identify which columns must be changed to fire the trigger. You cannot specify a column list for an INSERT or for a DELETE statement because it always affects entire rows.

... UPDATE OF salary ...

The triggering event can contain one, two, or all three of these DML operations.

... INSERT or UPDATE or DELETE

... INSERT or UPDATE OF job_id ...

The trigger body defines the action—that is, what needs to be done when the triggering event is issued. The PL/SQL block can contain SQL and PL/SQL statements, and can define PL/SQL constructs such as variables, cursors, exceptions, and so on. You can also call a PL/SQL procedure or a Java procedure.

Note: The size of a trigger cannot be greater than 32 KB.

4.1: Triggers

Statement-Level Triggers Versus Row-Level Triggers

Statement-Level Triggers	Row-Level Triggers
Is the default when creating a trigger	Use the FOR EACH ROW clause when creating a trigger.
Fires once for the triggering event	Fires once for each row affected by the triggering event
Fires once even if no rows are affected	Does not fire if the triggering event does not affect any rows

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 10

Types of DML Triggers

You can specify that the trigger will be executed once for every row affected by the triggering statement (such as a multiple row UPDATE) or once for the triggering statement, no matter how many rows it affects.

Statement Trigger

A statement trigger is fired once on behalf of the triggering event, even if no rows are affected at all. Statement triggers are useful if the trigger action does not depend on the data from rows that are affected or on data provided by the triggering event itself (for example, a trigger that performs a complex security check on the current user).

Row Trigger

A row trigger fires each time the table is affected by the triggering event. If the triggering event affects no rows, a row trigger is not executed. Row triggers are useful if the trigger action depends on data of the rows that are affected or on data provided by the triggering event itself.

Note: Row triggers use correlation names to access the old and new column values of the row being processed by the trigger.

4.1: Triggers

Trigger Restriction

- Specifies a Boolean expression that must evaluate to TRUE for the trigger to fire



Copyright © Capgemini 2015. All Rights Reserved 11

4.1: Triggers

Trigger Action

- A trigger action is the procedure that contains either PL/SQL block, Java program or C code which contains SQL statements and code to be executed



4.1: Triggers

Creating DML Triggers Using the CREATE TRIGGER Statement

```

CREATE [OR REPLACE] TRIGGER trigger_name
  timing -- when to fire the trigger
  event1 [OR event2 OR event3]
  ON object_name
  [REFERENCING OLD AS old | NEW AS new]
  FOR EACH ROW -- default is statement level trigger
  WHEN (condition)])
  DECLARE]
  BEGIN
    ... trigger_body -- executable statements
  [EXCEPTION . . .]
  END [trigger_name];

```

timing = BEFORE | AFTER | INSTEAD OF

event = INSERT | DELETE | UPDATE | UPDATE OF column_list

 Capgemini

Copyright © Capgemini 2015. All Rights Reserved. 13

Creating DML Triggers

The components of the trigger syntax are:

- *trigger_name* uniquely identifies the trigger.
- *timing* indicates when the trigger fires in relation to the triggering event. Values are BEFORE, AFTER, and INSTEAD OF.
- *event* identifies the DML operation causing the trigger to fire. Values are INSERT, UPDATE [OF column], and DELETE.
- *object_name* indicates the table or view associated with the trigger.

For row triggers, you can specify:

A REFERENCING clause to choose correlation names for referencing the old and new values of the current row (default values are OLD and NEW)
 FOR EACH ROW to designate that the trigger is a row trigger

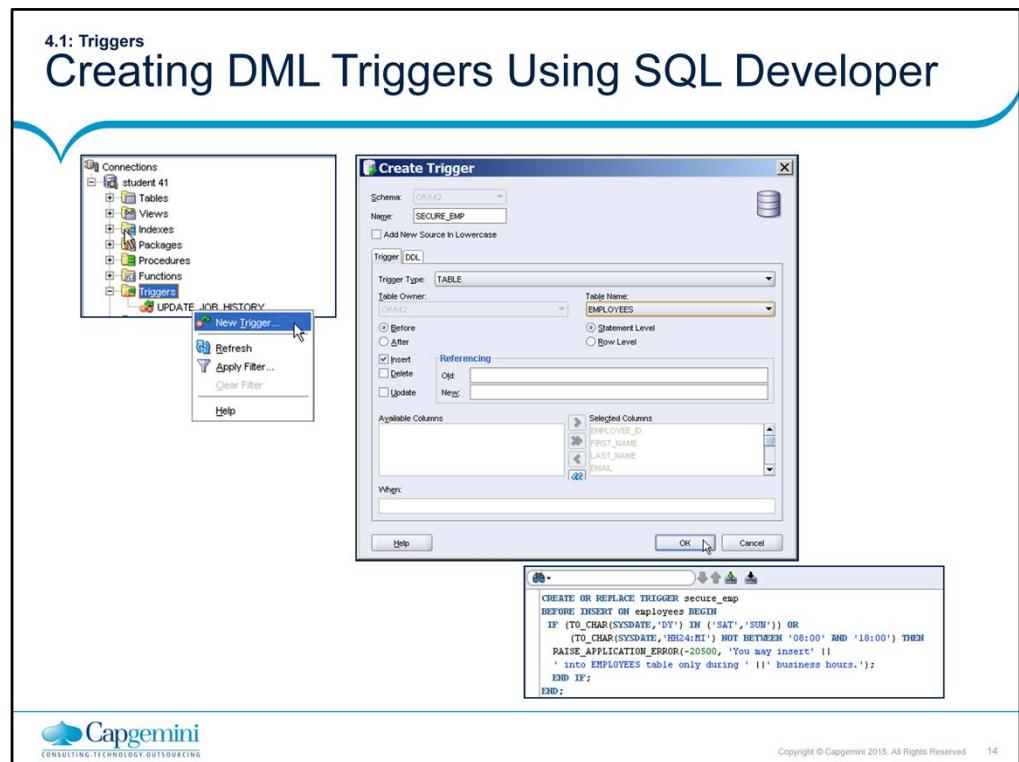
A WHEN clause to apply a conditional predicate, in parentheses, which is evaluated for each row to determine whether or not to execute the trigger body

The *trigger_body* is the action performed by the trigger, implemented as either of the following:

An anonymous block with a DECLARE or BEGIN, and an END

A CALL clause to invoke a stand-alone or packaged stored procedure, such as:

CALL my_procedure;



4.1: Triggers

Trigger-Firing Sequence:Single-Row Manipulation

- Use the following firing sequence for a trigger on a table when a single row is manipulated:

```
INSERT INTO departments
(department_id,department_name,location_id)
VALUES (400,'CONSULTING', 2400);
```

BEFORE statement trigger

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
30	Purchasing	1700
...		
400	CONSULTING	2400

BEFORE row trigger

AFTER row trigger

AFTER statement trigger

Capgemini

Copyright © Capgemini 2015. All Rights Reserved. 15

Trigger-Firing Sequence: Single-Row Manipulation

Create a statement trigger or a row trigger based on the requirement that the trigger must fire once for each row affected by the triggering statement, or just once for the triggering statement, regardless of the number of rows affected.

When the triggering DML statement affects a single row, both the statement trigger and the row trigger fire exactly once.

Example

The SQL statement in the slide does not differentiate statement triggers from row triggers because exactly one row is inserted into the table using the syntax for the `INSERT` statement shown in the slide.

4.1: Triggers

Trigger-Firing Sequence: Multirow Manipulation

- Use the following firing sequence for a trigger on a table when many rows are manipulated:

```
UPDATE employees
SET salary = salary * 1.1
WHERE department_id = 30;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
114	Raphaely	30
115	Khoo	30
116	Baida	30
117	Tobias	30
118	Himuro	30
119	Colmenares	30

Copyright © Capgemini 2015. All Rights Reserved. 16

Trigger-Firing Sequence: Multirow Manipulation

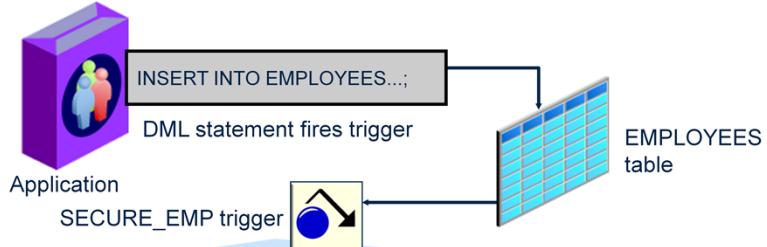
When the triggering DML statement affects many rows, the statement trigger fires exactly once, and the row trigger fires once for every row affected by the statement.

Example

The SQL statement in the slide causes a row-level trigger to fire a number of times equal to the number of rows that satisfy the WHERE clause (that is, the number of employees reporting to department 30).

4.1: Triggers

Creating a DML Statement Trigger Example: SECURE_EMP



```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON employees
BEGIN
  IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
  (TO_CHAR(SYSDATE,'HH24:MI') NOT BETWEEN '08:00' AND '18:00') THEN
    RAISE_APPLICATION_ERROR(-20500, 'You may insert
    '' into EMPLOYEES table only during
    '' normal business hours.');
  END IF;
END;
```



Copyright © Capgemini 2015. All Rights Reserved. 17

Creating a DML Statement Trigger

In the example in the slide, the SECURE_EMP database trigger is a BEFORE statement trigger that prevents the INSERT operation from succeeding if the business condition is violated. In this case, the trigger restricts inserts into the EMPLOYEES table during certain business hours, Monday through Friday.

If a user attempts to insert a row into the EMPLOYEES table on Saturday, then the user sees an error message, the trigger fails, and the triggering statement is rolled back. Remember that the

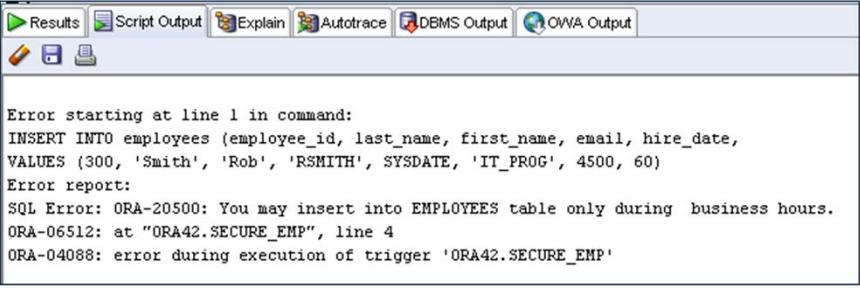
RAISE_APPLICATION_ERROR is a server-side built-in procedure that returns an error to the user and causes the PL/SQL block to fail.

When a database trigger fails, the triggering statement is automatically rolled back by the Oracle server.

4.1: Triggers

Testing Trigger SECURE_EMP

```
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);
```



Results Script Output Explain Autotrace DBMS Output OWA Output

Capgemini

Copyright © Capgemini 2015. All Rights Reserved. 18

Testing SECURE_EMP

To test the trigger, insert a row into the EMPLOYEES table during nonbusiness hours. When the date and time are out of the business hours specified in the trigger, you receive the error message shown in the slide.

4.1: Triggers

Using Conditional Predicates

```

CREATE OR REPLACE TRIGGER secure_emp BEFORE
INSERT OR UPDATE OR DELETE ON employees
BEGIN
  IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
  (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN '08' AND '18') THEN
    IF DELETING THEN RAISE_APPLICATION_ERROR(
      -20502,'You may delete from EMPLOYEES table'|||
      'only during normal business hours.');
    ELSIF INSERTING THEN RAISE_APPLICATION_ERROR(
      -20500,'You may insert into EMPLOYEES table'|||
      'only during normal business hours.');
    ELSIF UPDATING ('SALARY') THEN
      RAISE_APPLICATION_ERROR(-20503, 'You may '|||
      'update SALARY only normal during business hours.');
    ELSE RAISE_APPLICATION_ERROR(-20504, 'You may'|||
      ' update EMPLOYEES table only during'|||
      ' normal business hours.');
    END IF;
  END IF;
END;

```

 Capgemini

Copyright © Capgemini 2015. All Rights Reserved. 19

Detecting the DML Operation That Fired a Trigger

If more than one type of DML operation can fire a trigger (for example, ON INSERT OR DELETE OR UPDATE OF Emp_tab), the trigger body can use the conditional predicates INSERTING, DELETING, and UPDATING to check which type of statement fired the trigger.

You can combine several triggering events into one by taking advantage of the special conditional predicates INSERTING, UPDATING, and DELETING within the trigger body.

Example

Create one trigger to restrict all data manipulation events on the EMPLOYEES table to certain business hours, 8 AM to 6 PM, Monday through Friday.

4.1: Triggers

Creating a DML Row Trigger

```
CREATE OR REPLACE TRIGGER restrict_salary
BEFORE INSERT OR UPDATE OF salary ON employees
FOR EACH ROW
BEGIN
  IF NOT (:NEW.job_id IN ('AD_PRES', 'AD_VP'))
  AND :NEW.salary > 15000 THEN
    RAISE_APPLICATION_ERROR (-20202,
      'Employee cannot earn more than $15,000.');
  END IF;
END;
```

```
UPDATE employees
SET salary = 15500
WHERE last_name = 'Russell';
```

```
Error starting at line 1 in command:
UPDATE employees
SET salary = 15500
WHERE last_name = 'Russell'
Error report:
SQL Error: ORA-20202: Employee cannot earn more than $15,000.
ORA-06512: at "ORA62.RESTRICT_SALARY", line 4
ORA-04088: error during execution of trigger 'ORA62.RESTRICT_SALARY'
```



Copyright © Capgemini 2015. All Rights Reserved. 20

Creating a DML Row Trigger

You can create a BEFORE row trigger in order to prevent the triggering operation from succeeding if a certain condition is violated.

In the first example in the slide, a trigger is created to allow only employees whose job IDs are either AD_PRES or AD_VP to earn a salary of more than 15,000. If you try to update the salary of employee Russell whose employee ID is SA_MAN, the trigger raises the exception displayed in the slide.

Note: Before executing the first code example in the slide, make sure you disable the secure_emp and secure_employees triggers.

4.1: Triggers

Using OLD and NEW Qualifiers

- When a row-level trigger fires, the PL/SQL run-time engine creates and populates two data structures:
 - OLD: Stores the original values of the record processed by the trigger
 - NEW: Contains the new values
- NEW and OLD have the same structure as a record declared using the %ROWTYPE on the table to which the trigger is attached.

Data Operations	Old Value	New Value
INSERT	NULL	Inserted value
UPDATE	Value before update	Value after update
DELETE	Value before delete	NULL



Copyright © Capgemini 2015. All Rights Reserved. 21

Using OLD and NEW Qualifiers

Within a ROW trigger, you can reference the value of a column before and after the data change by prefixing it with the OLD and NEW qualifiers.

Note

The OLD and NEW qualifiers are available only in ROW triggers. Prefix these qualifiers with a colon (:) in every SQL and PL/SQL statement.

There is no colon (:) prefix if the qualifiers are referenced in the WHEN restricting condition.

Row triggers can decrease the performance if you perform many updates on larger tables.

4.1: Triggers

New and Old Values

- New and old values of the DML statements can be processed with :NEW.column_name and :OLD.column_name in the trigger restriction and trigger action .
- Insert will have values in New variable
- Update will have values in New and Old variables
- Delete will have values in Old variable

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 22

4.1: Triggers

Database Triggers - DML

- Create or replace trigger <trigger_name>
- after/before
- insert/update of <column_list>/delete on <table_name/view_name>
- for each row
- When (<condition>)
- <pl_sql >

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 23

4.2: Trigger Types

Types of Trigger

- Row triggers and Statement triggers
- Before and After triggers
- Instead of triggers
- Triggers on system events and user events



Copyright © Capgemini 2015. All Rights Reserved 24

4.2: Trigger Types

Row and Statement Triggers

- Row triggers fire once for every row affected by the triggering statement
- Statement triggers fire once on behalf of the triggering event



Copyright © Capgemini 2015. All Rights Reserved

25

4.2: Trigger Types

Before and After triggers

- Specify the trigger timing
- Are fired by DML statements either before or after the execution of the DML statements
- Apply to row and statement triggers
- Cannot be specified on views
- Trigger type combinations :
- Before statement trigger
- Before row trigger
- After row trigger
- After statement trigger
- You can have multiple triggers of the same type for the same statement for any given table
-



Copyright © Capgemini 2015. All Rights Reserved

26

4.2: Trigger Types

Firing Sequence

- Before Statement trigger
- Before Row trigger
- After Row trigger
- After Statement trigger

Fires for all the affected rows

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 27

4.2: Trigger Types

Trigger Firing Order

- Starting from 11g Oracle allows you to specify trigger firing order if more than one trigger is created .
- It is done using **FOLLOW**s keyword followed by trigger name after which current trigger is to be invoked.



Contd.,

- create or replace trigger test_trg2 before insert on test for each row follows test_trg1 begin insert into testlog values ('From test_trig2'); end;



4.2: Trigger Types

Instead of triggers

- Complex views which cannot be modified by DML statements can be modified by using Instead of trigger
- It provides a transparent way of modifying the base tables through the views
- Trigger is fired instead of executing the triggering statement



Copyright © Capgemini 2015. All Rights Reserved 30

4.2: Trigger Types

Triggers on System Events and User Events

- Certain system events like database startup and shutdown and server error messages can be traced through triggers
- User events like user logon and logoff, DDL and DML can also be traced through triggers



4.2: Trigger Types

Example of DML trigger

```
create or replace trigger emp_del
before delete on emp
for each row
begin
    insert into emp_history values
    (:old.empno,:old.ename,:old.job,:old.mgr,:old.hiredate,
    :old.sal,:old.comm,:old.deptno);

end ;
/
```



Copyright © Capgemini 2015. All Rights Reserved 32

4.2: Trigger Types

Example

```
create or replace trigger dept_tot_emp
after insert on emp
for each row
begin

    update dept set tot_emp = tot_emp + 1
    where deptno = :new.deptno;
end;
/
```



Copyright © Capgemini 2015. All Rights Reserved 33

4.2: Trigger Types

Example of DML trigger

```
create or replace trigger dept_tot_emp
after insert or delete or update of deptno on emp
for each row
begin

if inserting or updating then
    update dept set tot_emp = tot_emp + 1
    where deptno = :new.deptno;

end if;
```



Copyright © Capgemini 2015. All Rights Reserved 34

4.2: Trigger Types

Example of DML trigger

if updating or deleting then

```
update dept set tot_emp = tot_emp - 1  
where deptno = :old.deptno;
```

```
end if;
```

```
end;
```



Copyright © Capgemini 2015. All Rights Reserved 35

4.2: Trigger Types

Example of DML trigger

```
create or replace trigger dept_tot_emp
after insert or delete or update of deptno on emp
for each row
when(old.deptno <> new.deptno)
begin
if inserting or updating then
    update dept set tot_emp = tot_emp + 1
    where deptno = :new.deptno;
end if;
if updating or deleting then
    update dept set tot_emp = tot_emp - 1
    where deptno = :old.deptno;
end if;
end;
```



Copyright © Capgemini 2015. All Rights Reserved

36

4.2: Trigger Types Using OLD and NEW Qualifiers: Example

```
CREATE TABLE audit_emp (
  user_name  VARCHAR2(30),
  time_stamp  date,
  id        NUMBER(6),
  old_last_name VARCHAR2(25),
  new_last_name VARCHAR2(25),
  old_title  VARCHAR2(10),
  new_title  VARCHAR2(10),
  old_salary  NUMBER(8,2),
  new_salary  NUMBER(8,2) )
/
CREATE OR REPLACE TRIGGER audit_emp_values
AFTER DELETE OR INSERT OR UPDATE ON employees
FOR EACH ROW
BEGIN
  INSERT INTO audit_emp(user_name, time_stamp, id,
  old_last_name, new_last_name, old_title,
  new_title, old_salary, new_salary)
  VALUES (USER, SYSDATE, :OLD.employee_id,
  :OLD.last_name, :NEW.last_name, :OLD.job_id,
  :NEW.job_id, :OLD.salary, :NEW.salary);
END;
```



Copyright © Capgemini 2015. All Rights Reserved. 37

Using OLD and NEW Qualifiers: Example

In the example in the slide, the AUDIT_EMP_VALUES trigger is created on the EMPLOYEES table. The trigger adds rows to a user table, AUDIT_EMP, logging a user's activity against the EMPLOYEES table. The trigger records the values of several columns both before and after the data changes by using the OLD and NEW qualifiers with the respective column name.

4.2: Trigger Types

Using OLD and NEW Qualifiers:Example

```

INSERT INTO employees (employee_id, last_name, job_id, salary, email,
hire_date)
VALUES (999, 'Temp emp', 'SA_REP', 6000, 'TEMPEMP',
TRUNC(SYSDATE))
/
UPDATE employees
SET salary = 7000, last_name = 'Smith'
WHERE employee_id = 999
/
SELECT *
FROM audit_emp;

```

Results:

USER_NAME	TIME_STAMP	ID	OLD_LAST_NAME	NEW_LAST_NAME	OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
1 ORA61	04-JUN-09	(null)	(null)	Temp emp	(null)	SA_REP	(null)	6000
2 ORA61	04-JUN-09	999	Temp emp	Smith	SA_REP	SA_REP	6000	7000



Copyright © Capgemini 2015. All Rights Reserved 38

Using OLD and NEW Qualifiers: Example the Using AUDIT_EMP Table

Create a trigger on the EMPLOYEES table to add rows to a user table, AUDIT_EMP, logging a user's activity against the EMPLOYEES table. The trigger records the values of several columns both before and after the data changes by using the OLD and NEW qualifiers with the respective column name.

The following is the result of inserting the employee record into the EMPLOYEES table:

...

The following is the result of updating the salary for employee "Smith":

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
999 (null)	Smith	TEMPEMP	(null)	04-JUN-09	SA_REP	7000	(null)	(null)	(null)	(null)
300 Rob	Smith	RSMITH	(null)	04-JUN-09	IT_PROG	4500	(null)	(null)	(null)	60
206 William	Gietz	WGJETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300	(null)	205	110	

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
999 (null)	Smith	TEMPEMP	(null)	04-JUN-09	SA_REP	7000	(null)	(null)	(null)	(null)

4.2: Trigger Types

Using the WHEN Clause to Fire a Row Trigger Based on a Condition

```
CREATE OR REPLACE TRIGGER derive_commission_pct
BEFORE INSERT OR UPDATE OF salary ON employees
FOR EACH ROW
WHEN (NEW.job_id = 'SA_REP')
BEGIN
  IF INSERTING THEN
    :NEW.commission_pct := 0;
  ELSIF :OLD.commission_pct IS NULL THEN
    :NEW.commission_pct := 0;
  ELSE
    :NEW.commission_pct := :OLD.commission_pct+0.05;
  END IF;
END;
/
```



Copyright © Capgemini 2015. All Rights Reserved. 39

Restricting a Row Trigger: Example

Optionally, you can include a trigger restriction in the definition of a row trigger by specifying a Boolean SQL expression in a WHEN clause. If you include a WHEN clause in the trigger, then the expression in the WHEN clause is evaluated for each row that the trigger affects.

If the expression evaluates to TRUE for a row, then the trigger body executes on behalf of that row. However, if the expression evaluates to FALSE or NOT TRUE for a row (unknown, as with nulls), then the trigger body does not execute for that row. The evaluation of the WHEN clause does not have an effect on the execution of the triggering SQL statement (in other words, the triggering statement is not rolled back if the expression in a WHEN clause evaluates to FALSE).

Note: A WHEN clause cannot be included in the definition of a statement trigger.

In the example in the slide, a trigger is created on the EMPLOYEES table to calculate an employee's commission when a row is added to the EMPLOYEES table, or when an employee's salary is modified.

The NEW qualifier cannot be prefixed with a colon in the WHEN clause because the WHEN clause is outside the PL/SQL blocks.

4.2: Trigger Types

Summary of the Trigger Execution Model

1. Execute all BEFORE STATEMENT triggers.
2. Loop for each row affected by the SQL statement:
 - a. Execute all BEFORE ROW triggers for that row.
 - b. Execute the DML statement and perform integrity constraint checking for that row.
 - c. Execute all AFTER ROW triggers for that row.
3. Execute all AFTER STATEMENT triggers.



Copyright © Capgemini 2015. All Rights Reserved. 40

Trigger Execution Model

A single DML statement can potentially fire up to four types of triggers:

- BEFORE and AFTER statement triggers
- BEFORE and AFTER row triggers

A triggering event or a statement within the trigger can cause one or more integrity constraints to be checked. However, you can defer constraint checking until a COMMIT operation is performed.

Triggers can also cause other triggers—known as cascading triggers—to fire.

All actions and checks performed as a result of a SQL statement must succeed. If an exception is raised within a trigger and the exception is not explicitly handled, then all actions performed because of the original SQL statement are rolled back (including actions performed by firing triggers). This guarantees that integrity constraints can never be compromised by triggers.

When a trigger fires, the tables referenced in the trigger action may undergo changes by other users' transactions. In all cases, a read-consistent image is guaranteed for the modified values that the trigger needs to read (query) or write (update).

Note: Integrity checking can be deferred until the COMMIT operation is performed.

4.2: Trigger Types

Implementing an Integrity Constraint with an After Trigger

```
-- Integrity constraint violation error -2991 raised.
UPDATE employees SET department_id = 999
WHERE employee_id = 170;
```

```
CREATE OR REPLACE TRIGGER employee_dept_fk_trg
AFTER UPDATE OF department_id ON employees
FOR EACH ROW
BEGIN
  INSERT INTO departments VALUES(:new.department_id,
    'Dept'||:new.department_id, NULL, NULL);
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    NULL; -- mask exception if department exists
END;
/
```

```
-- Successful after trigger is fired
UPDATE employees SET department_id = 999
WHERE employee_id = 170;
```

1 rows updated


Copyright © Capgemini 2015. All Rights Reserved.
41

Implementing an Integrity Constraint with an After Trigger

The example in the slide explains a situation in which the integrity constraint can be taken care of by using an AFTER trigger. The EMPLOYEES table has a foreign key constraint on the DEPARTMENT_ID column of the DEPARTMENTS table.

In the first SQL statement, the DEPARTMENT_ID of the employee 170 is modified to 999. Because department 999 does not exist in the DEPARTMENTS table, the statement raises exception -2291 for the integrity constraint violation.

The EMPLOYEE_DEPT_TRG trigger is created and it inserts a new row into the DEPARTMENTS table by using :NEW.DEPARTMENT_ID for the value of the new department's DEPARTMENT_ID. The trigger fires when the UPDATE statement modifies the DEPARTMENT_ID of employee 170 to 999. When the foreign key constraint is checked, it is successful because the trigger inserted the department 999 into the DEPARTMENTS table. Therefore, no exception occurs unless the department already exists when the trigger attempts to insert the new row. However, the EXCEPTION handler traps and masks the exception allowing the operation to succeed.

Note: Although the example shown in the slide is somewhat contrived due to the limited data in the HR schema, the point is that if you defer the constraint check until the commit, you then have the ability to engineer a trigger to detect that constraint failure and repair it prior to the commit action.

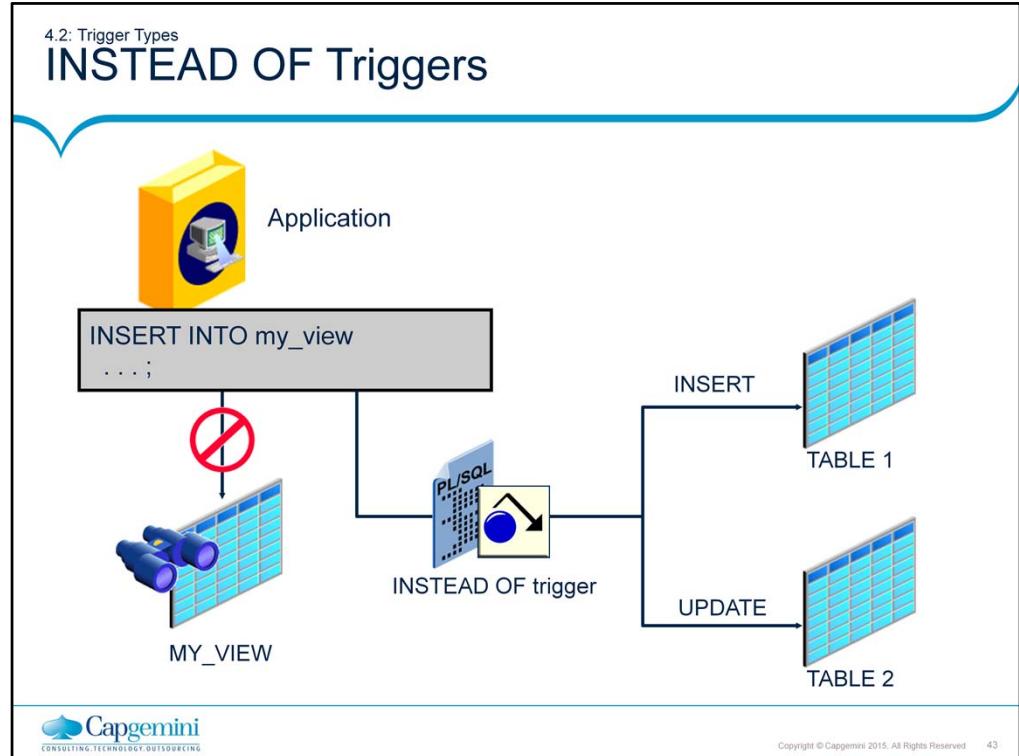
4.2: Trigger Types

Database trigger – Instead of

- Create or replace trigger <trigger_name>
- Instead of insert on <table_name/view_name>
- for each row
- <pl_sql>



Copyright © Capgemini 2015. All Rights Reserved 42



INSTEAD OF Triggers

Use INSTEAD OF triggers to modify data in which the DML statement has been issued against an inherently un-updatable view. These triggers are called INSTEAD OF triggers because, unlike other triggers, the Oracle server fires the trigger instead of executing the triggering statement. These triggers are used to perform INSERT, UPDATE, and DELETE operations directly on the underlying tables. You can write INSERT, UPDATE, and DELETE statements against a view, and the INSTEAD OF trigger works invisibly in the background to make the right actions take place. A view cannot be modified by normal DML statements if the view query contains set operators, group functions, clauses such as GROUP BY, CONNECT BY, START, the DISTINCT operator, or joins. For example, if a view consists of more than one table, an insert to the view may entail an insertion into one table and an update to another. So you write an INSTEAD OF trigger that fires when you write an insert against the view. Instead of the original insertion, the trigger body executes, which results in an insertion of data into one table and an update to another table.

Note: If a view is inherently updatable and has INSTEAD OF triggers, then the triggers take precedence. INSTEAD OF triggers are row triggers. The CHECK option for views is not enforced when insertions or updates to the view are performed by using INSTEAD OF triggers. The INSTEAD OF trigger body must enforce the check.

4.2: Trigger Types

Instead of Trigger

```
create or replace trigger emp_details_insert
Instead of insert on emp_details
for each row
begin

insert into emp(EMPNO,ENAME,JOB,MGR,HIREDATE,DEPTNO)
values(:new.empno,:new.ename,:new.job,:new.mgr,:new.hiredate,:new.deptn
o);

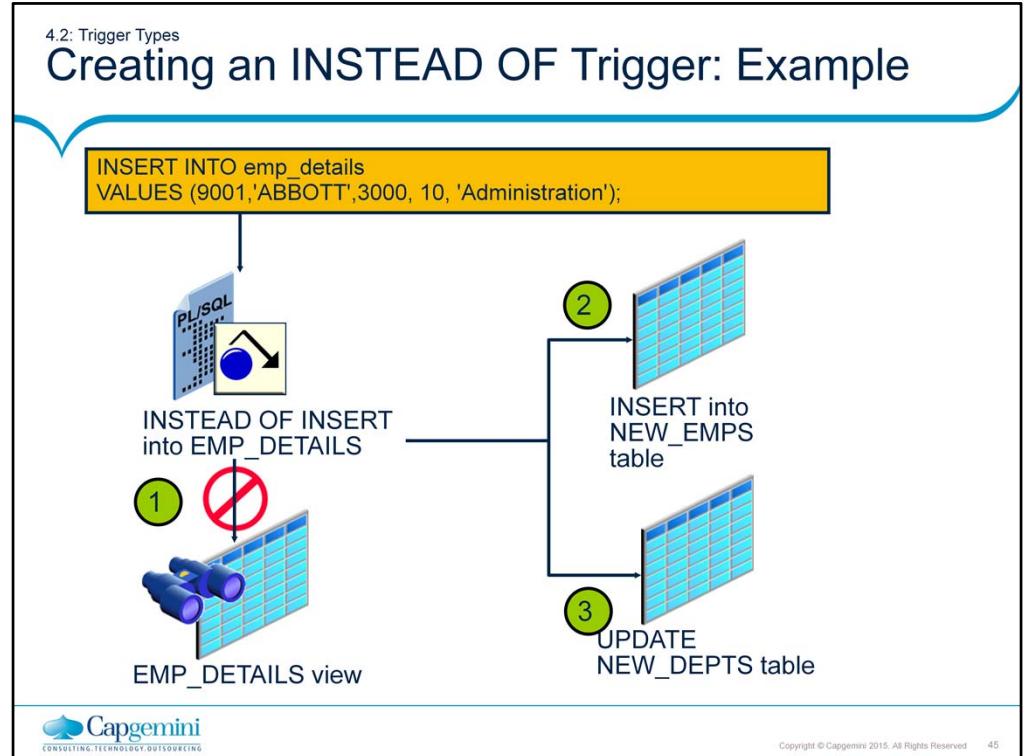
insert into emp_addr(EMPNO,ADDRESS,CONTACT)
values (:new.empno,:new.address,:new.contact);

end;
/
```



Copyright © Capgemini 2015. All Rights Reserved

44



Creating an INSTEAD OF Trigger

You can create an INSTEAD OF trigger in order to maintain the base tables on which a view is based.

The example in the slide illustrates an employee being inserted into the view EMP_DETAILS, whose query is based on the EMPLOYEES and DEPARTMENTS tables. The NEW_EMP_DEPT (INSTEAD OF) trigger executes in place of the INSERT operation that causes the trigger to fire. The INSTEAD OF trigger then issues the appropriate INSERT and UPDATE to the base tables used by the EMP_DETAILS view. Therefore, instead of inserting the new employee record into the EMPLOYEES table, the following actions take place:

1. The NEW_EMP_DEPT INSTEAD OF trigger fires.
2. A row is inserted into the NEW_EMPS table.
3. The DEPT_SAL column of the NEW_DEPTS table is updated. The salary value supplied for the new employee is added to the existing total salary of the department to which the new employee has been assigned.

Note: Before you run the example in the slide, you must create the required structures shown on the next two pages.

4.2: Trigger Types

Example of Instead of Trigger

- EMP table
- EMPNO
- ENAME
- JOB
- MGR
- HIREDATE
- SAL
- COMM
- DEPTNO

EMP_ADDR table

EMPNO
ADDRESS
CONTACT



Copyright © Capgemini 2015. All Rights Reserved 46

4.2: Trigger Types

Defining Complex View

- create view emp_details
- as
- select e.empno,ename,address,contact,job,mgr,hiredate,deptno
from emp e,emp_addr a
- where e.empno = a.empno



Copyright © Capgemini 2015. All Rights Reserved 47

4.2: Trigger Types

Creating an INSTEAD OF Trigger to Perform DML on Complex Views

```

CREATE TABLE new_emps AS
SELECT employee_id, last_name, salary, department_id
  FROM employees;

CREATE TABLE new_depts AS
SELECT d.department_id, d.department_name,
       sum(e.salary) dept_sal
  FROM employees e, departments d
 WHERE e.department_id = d.department_id;

CREATE VIEW emp_details AS
SELECT e.employee_id, e.last_name, e.salary,
       e.department_id, d.department_name
  FROM employees e, departments d
 WHERE e.department_id = d.department_id
 GROUP BY d.department_id, d.department_name;

```



Copyright © Capgemini 2015. All Rights Reserved. 48

Creating an INSTEAD OF Trigger (continued)

The example in the slide creates two new tables, NEW_EMPS and NEW_DEPTS, that are based on the EMPLOYEES and DEPARTMENTS tables, respectively. It also creates an EMP_DETAILS view from the EMPLOYEES and DEPARTMENTS tables.

If a view has a complex query structure, then it is not always possible to perform DML directly on the view to affect the underlying tables. The example requires creation of an INSTEAD OF trigger, called NEW_EMP_DEPT, shown on the next page. The NEW_EMP_DEPT trigger handles DML in the following way:

When a row is inserted into the EMP_DETAILS view, instead of inserting the row directly into the view, rows are added into the NEW_EMPS and NEW_DEPTS tables, using the data values supplied with the INSERT statement.

When a row is modified or deleted through the EMP_DETAILS view, corresponding rows in the NEW_EMPS and NEW_DEPTS tables are affected.

Note: INSTEAD OF triggers can be written only for views, and the BEFORE and AFTER timing options are not valid.

4.2: Trigger Types

The Status of a Trigger

- A trigger is in either of two distinct modes:
 - Enabled: The trigger runs its trigger action if a triggering statement is issued and the trigger restriction (if any) evaluates to true (default).
 - Disabled: The trigger does not run its trigger action, even if a triggering statement is issued and the trigger restriction (if any) would evaluate to true.



 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 49

4.2: Trigger Types

Creating a Disabled Trigger

- Before Oracle Database 11g, if you created a trigger whose body had a PL/SQL compilation error, then DML to the table failed.
- In Oracle Database 11g, you can create a disabled trigger and then enable it only when you know it will be compiled successfully.

```
CREATE OR REPLACE TRIGGER mytrg
  BEFORE INSERT ON mytable FOR EACH ROW
  DISABLE
BEGIN
  :New.ID := my_seq.Nextval;
  ...
END;
/
```



Copyright © Capgemini 2015. All Rights Reserved 50

Creating a Disabled Trigger

Before Oracle Database 11g, if you created a trigger whose body had a PL/SQL compilation error, then DML to the table failed. The following error message was displayed:

ORA-04098: trigger 'TRG' is invalid and failed re-validation

In Oracle Database 11g, you can create a disabled trigger, and then enable it only when you know it will be compiled successfully.

You can also temporarily disable a trigger in the following situations:

An object it references is not available.

You need to perform a large data load, and you want it to proceed quickly without firing triggers.

You are reloading data.

Note: The code example in the slide assumes that you have an existing sequence named `my_seq`.

4.2: Trigger Types

Managing Triggers Using the ALTER and DROP SQL Statements

-- Disable or reenable a database trigger:

```
ALTER TRIGGER trigger_name DISABLE | ENABLE;
```

-- Disable or reenable all triggers for a table:

```
ALTER TABLE table_name DISABLE | ENABLE ALL TRIGGERS;
```

-- Recompile a trigger for a table:

```
ALTER TRIGGER trigger_name COMPILE;
```

-- Remove a trigger from the database:

```
DROP TRIGGER trigger_name;
```

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 51

Managing Triggers

A trigger has two modes or states: ENABLED and DISABLED. When a trigger is first created, it is enabled by default. The Oracle server checks integrity constraints for enabled triggers and guarantees that triggers cannot compromise them. In addition, the Oracle server provides read-consistent views for queries and constraints, manages the dependencies, and provides a two-phase commit process if a trigger updates remote tables in a distributed database.

Disabling a Trigger

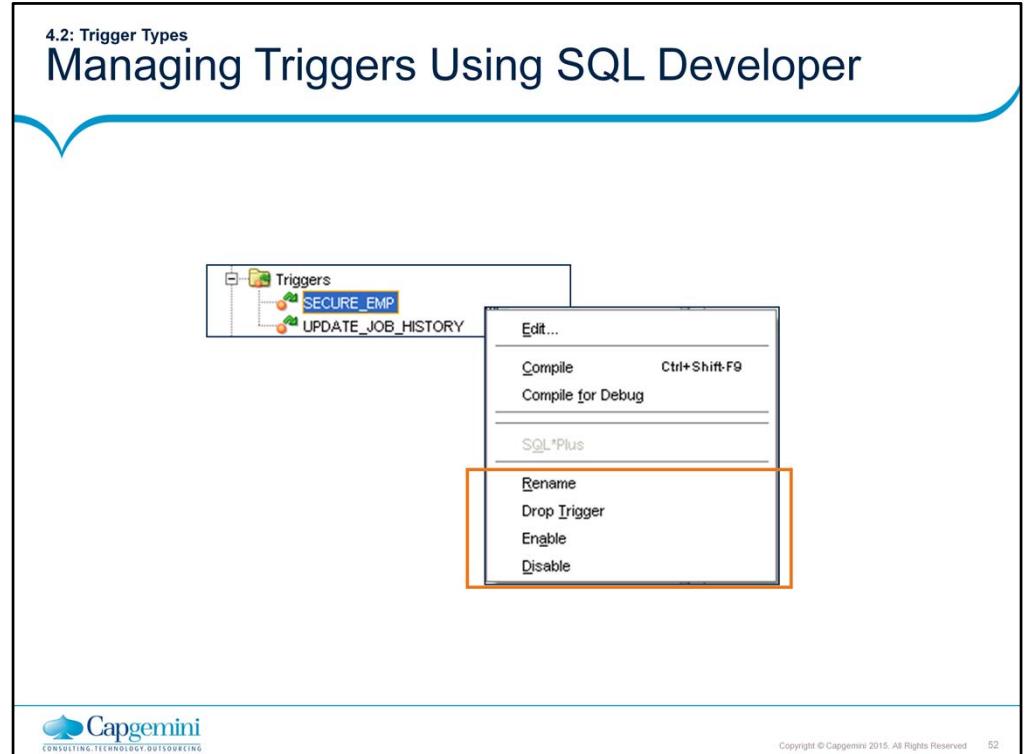
Use the ALTER TRIGGER command to disable a trigger. You can also disable all triggers on a table by using the ALTER TABLE command. You can disable triggers to improve performance or to avoid data integrity checks when loading massive amounts of data with utilities such as SQL*Loader. You might also disable a trigger when it references a database object that is currently unavailable, due to a failed network connection, disk crash, offline data file, or offline tablespace.

Recompiling a Trigger

Use the ALTER TRIGGER command to explicitly recompile a trigger that is invalid.

Removing Triggers

When a trigger is no longer required, use a SQL statement in SQL Developer or SQL*Plus to remove it. When you remove a table, all triggers on that table are also removed.



Managing Triggers Using SQL Developer

You can use the Triggers node in the Connections navigation tree to manage triggers. Right-click a trigger name, and then select one of the following options:

- Edit
- Compile
- Compile for Debug
- Rename
- Drop Trigger
- Enable
- Disable

4.2: Trigger Types

Testing Triggers

- Test each triggering data operation, as well as non-triggering data operations.
- Test each case of the WHEN clause.
- Cause the trigger to fire directly from a basic data operation, as well as indirectly from a procedure.
- Test the effect of the trigger on other triggers.
- Test the effect of other triggers on the trigger.

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 53

Testing Triggers

Testing code can be a time-consuming process. Do the following when testing triggers:

Ensure that the trigger works properly by testing a number of cases separately:

Test the most common success scenarios first.

Test the most common failure conditions to see that they are properly managed.

The more complex the trigger, the more detailed your testing is likely to be. For example, if you have a row trigger with a WHEN clause specified, then you should ensure that the trigger fires when the conditions are satisfied. Or, if you have cascading triggers, you need to test the effect of one trigger on the other and ensure that you end up with the desired results.

Use the DBMS_OUTPUT package to debug triggers.

4.2: Trigger Types

Viewing Trigger Information

- You can view the following trigger information:

Data Dictionary View	Description
USER_OBJECTS	Displays object information
USER/ALL/DBA_TRIGGERS	Displays trigger information
USER_ERRORS	Displays PL/SQL syntax errors for a trigger

 Capgemini
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 54

Viewing Trigger Information

The slide shows the data dictionary views that you can access to get information regarding the triggers.

The USER_OBJECTS view contains the name and status of the trigger and the date and time when the trigger was created.

The USER_ERRORS view contains the details about the compilation errors that occurred while a trigger was compiling. The contents of these views are similar to those for subprograms.

The USER_TRIGGERS view contains details such as name, type, triggering event, the table on which the trigger is created, and the body of the trigger. The SELECT Username FROM USER_USERS; statement gives the name of the owner of the trigger, not the name of the user who is updating the table.

4.2: Trigger Types

Using USER_TRIGGERS

DESCRIBE user_triggers

DESCRIBE user_triggers	Null	Type
TRIGGER_NAME		VARCHAR2(30)
TRIGGER_TYPE		VARCHAR2(16)
TRIGGERING_EVENT		VARCHAR2(27)
TABLE_OWNER		VARCHAR2(30)
BASE_OBJECT_TYPE		VARCHAR2(16)
TABLE_NAME		VARCHAR2(30)
COLUMN_NAME		VARCHAR2(4000)
REFERENCING_NAMES		VARCHAR2(128)
WHEN_CLAUSE		VARCHAR2(4000)
STATUS		VARCHAR2(8)
DESCRIPTION		VARCHAR2(4000)
ACTION_TYPE		VARCHAR2(11)
TRIGGER_BODY		LONG
CROSSEDITION		VARCHAR2(7)
BEFORE_STATEMENT		VARCHAR2(3)
BEFORE_ROW		VARCHAR2(3)
AFTER_ROW		VARCHAR2(3)
AFTER_STATEMENT		VARCHAR2(3)
INSTEAD_OF_ROW		VARCHAR2(3)
FIRE_ONCE		VARCHAR2(3)
APPLY_SERVER_ONLY		VARCHAR2(3)

21 rows selected

```
SELECT trigger_type, trigger_body
FROM user_triggers
WHERE trigger_name = 'SECURE_EMP';
```



Copyright © Capgemini 2015. All Rights Reserved 55

Using USER_TRIGGERS

If the source file is unavailable, then you can use the SQL Worksheet in SQL Developer or SQL*Plus to regenerate it from USER_TRIGGERS. You can also examine the ALL_TRIGGERS and DBA_TRIGGERS views, each of which contains the additional column OWNER, for the owner of the object. The result for the second example in the slide is as follows:

TRIGGER_TYPE	TRIGGER_BODY
BEFORE STATEMENT BEGIN	<pre> IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN '08' AND '18') THEN IF DELETING THEN RAISE_APPLICATION_ERROR(-20502, 'You may delete from EMPLOYEES table only during normal business hours.'); ELSIF INSERTING THEN RAISE_APPLICATION_ERROR(-20500, 'You may insert into EMPLOYEES table only during normal business hours.'); ELSIF UPDATING('SALARY') THEN RAISE_APPLICATION_ERROR(-20503, 'You may update SALARY only during normal business hours.'); ELSE RAISE_APPLICATION_ERROR(-20504, 'You may update EMPLOYEES table only during normal business hours.'); END IF; END IF; END;</pre>

4.2: Trigger Types

Managing Triggers

- Disable or re-enable a database trigger `Alter trigger trigger_name disable | enable`
- Disable or re-enable all triggers for a table `Alter table table_name disable | enable all triggers`
- Recompile a trigger `Alter trigger name compile`



Copyright © Capgemini 2015. All Rights Reserved

56

Summary

- Create database triggers that are invoked by DML operations
- Create statement and row trigger types
- Use database trigger-firing rules
- Enable, disable, and manage database triggers
- Develop a strategy for testing triggers
- Remove database triggers



Copyright © Capgemini 2015. All Rights Reserved 57

Add the notes here.

Review Question

- Question 1: Triggers should not issue Transaction Control Statements (TCL)
 - True / False
- Question 2: BEFORE DROP and AFTER DROP triggers are fired when a schema object is dropped
 - True / False
- Question 3: The :new and :old records must be used in WHEN clause with a colon
 - True / False



Copyright © Capgemini 2015. All Rights Reserved 58

Add the notes here.

Review Question

- Question 4: A _____ is a table that is currently being modified by a DML statement
- Question 5: A _____ is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects



Copyright © Capgemini 2015. All Rights Reserved 59

Add the notes here.