Oracle PL/SQL

PL/SQL Package and Trigger

Agenda

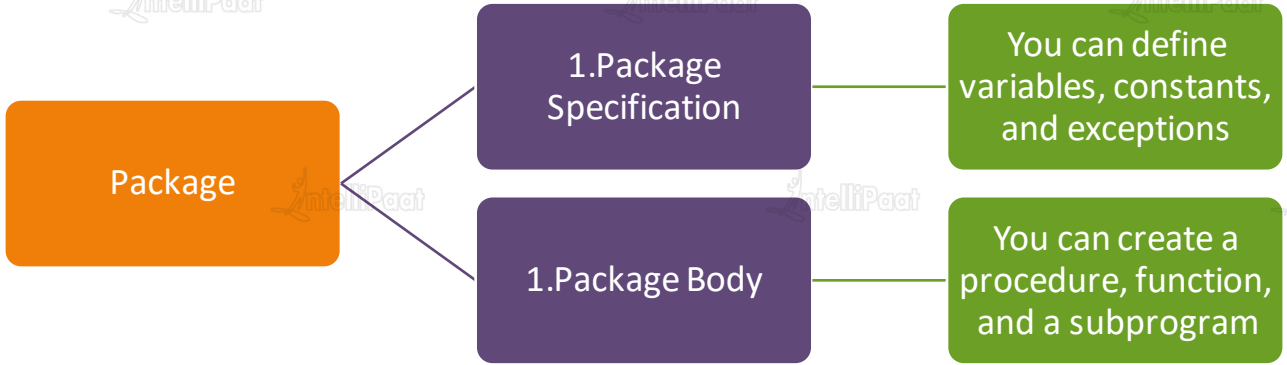
01 PL/SQL Package

02 PL/SQL Trigger

PL/SQL Package

PL/SQL Package

Package is a schema object and a collection of related PL/SQL types (variables, constants), cursors, procedures, and functions which are defined within a single context



PL/SQL Package: Advantages

You can create a package to store all related functions. Procedures are grouped together into a single unit called packages

Packages are reliable for granting privileges

All functions and procedures within a package can share variables among them

Packages support overloading of functions and procedures

Packages improve the performance of loading multiple objects into memory at once; therefore, subsequent calls to related programs do not require a physical I/O

PL/SQL Package: Syntax

Defining Package Specification Syntax

```
CREATE [OR REPLACE] PACKAGE package_name
   IS | AS
   [variable_declaration ...]
   [constant_declaration ...]
   [exception_declaration ...]
   [cursor_specification ...]
   [PROCEDURE [Schema..] procedure_name
      [ (parameter {IN,OUT,IN OUT} datatype [,parameter]) ]
   ]
   [FUNCTION [Schema..] function_name
      [ (parameter {IN,OUT,IN OUT} datatype [,parameter]) ]
      RETURN return_datatype
   ]
END [package_name];
```

PL/SQL Package: Syntax

Creating Package Body Syntax

```
CREATE [OR REPLACE] PACKAGE BODY package_name
  IS | AS
  [private_variable_declaration ...]
  [private_constant_declaration ...]
  BEGIN
    [initialization_statement]
    [PROCEDURE [Schema..] procedure_name
      [ (parameter [,parameter]) ]
      IS | AS
        variable declarations;
        constant declarations;
      BEGIN
        statement(s);
      EXCEPTION
        WHEN ...
      END
    ]
    [FUNCTION [Schema..] function_name
        [ (parameter [,parameter]) ]
        RETURN return_datatype
      IS | AS
        variable declarations;
        constant declarations;
      BEGIN
        statement(s);
      EXCEPTION
        WHEN ...
      END
    ]
  [EXCEPTION
    WHEN built-in_exception_name_1 THEN
      User defined statement (action) w ill be taken;
  ]
END;
/
```

PL/SQL Package: Example

EMP_NO	EMP_NAME	EMP_DEPT	EMP_SALARY
1	Forbs Ross	Web Developer	45k
2	Marks Jems	Program Developer	38k
3	Saulin	Program Developer	34k
4	Zenia Sroll	Web Developer	42k

PL/SQL Package: Example

Defining Package Specification

```
REATE or REPLACE PACKAGE pkg1
   IS | AS
   PROCEDURE pro1
      (no in number, name out varchar2);
   FUNCTION fun1
      (no in number)
      RETURN varchar2;
   END;
/
```

PL/SQL Package: Example

Defining Package Body

```
CREATE or REPLACE PACKAGE BODY pkg1
IS
   PROCEDURE pro1(no in number,info our varchar2)
      IS
      BEGIN
         SELECT * INTO temp FROM emp1 WHERE eno = no;
      END;

   FUNCTION fun1(no in number) return varchar2
      IS
      name varchar2(20);
      BEGIN
         SELECT ename INTO name FROM emp1 WHERE eno = no;
         RETURN name;
      END;
END;
/
```

PL/SQL Package: Example

Program Calling Package

```
DECLARE
   no number := &no;
   name varchar2(20);
BEGIN
   pkg1.pro1(no,info);
   dbms_output.put_line('Procedure Result');
   dbms_output.put_line(info.eno||'    '||
                        info.ename||'   '||
                        info.edept||'   '||
                        info.esalary||' '||);
   dbms_output.put_line('Function Result');
   name := pkg1.fun1(no);
   dbms_output.put_line(name);
END;
/
```

Result

```
SQL>@pkg_prg
no number &n=2
Procedure Result
2    marks jems    Program Developer    38K
Function Result
marks jems

PL/SQL procedure successfully completed.
```

PL/SQL Package: Alter

- **You can update the package code by just recompiling the package body.**

Syntax

```
ALTER PACKAGE
package_name COMPILE BODY;
```

Example

```
SQL>ALTER PACKAGE pkg1 COMPILE BODY;

Package body Altered
```

PL/SQL Package: Drop

- **You can drop a package using the DROP PACKAGE statement.**

Syntax

```
DROP PACKAGE package_name;
```

Example

```
SQL>DROP PACKAGE pkg1;

Package dropped
```

PL/SQL Triggers

PL/SQL Triggers

Oracle engine get invoked automatically whenever a specified event occurs.

Trigger is stored into the database and invoked repeatedly, when specific conditions match.

You can change the trigger mode to activate/deactivate, but you can't explicitly run

Trigger automatically associates with the DML statement; when the DML statement executes, trigger implicitly executes

You can create trigger using the CREATE TRIGGER statement. If trigger is activated, it fires the DML statement, and if trigger is deactivated it can't fire

Components of Triggers

A Triggering SQL Statement

SQL DML (INSERT, UPDATE, and DELETE) statements that implicitly call trigger to execute

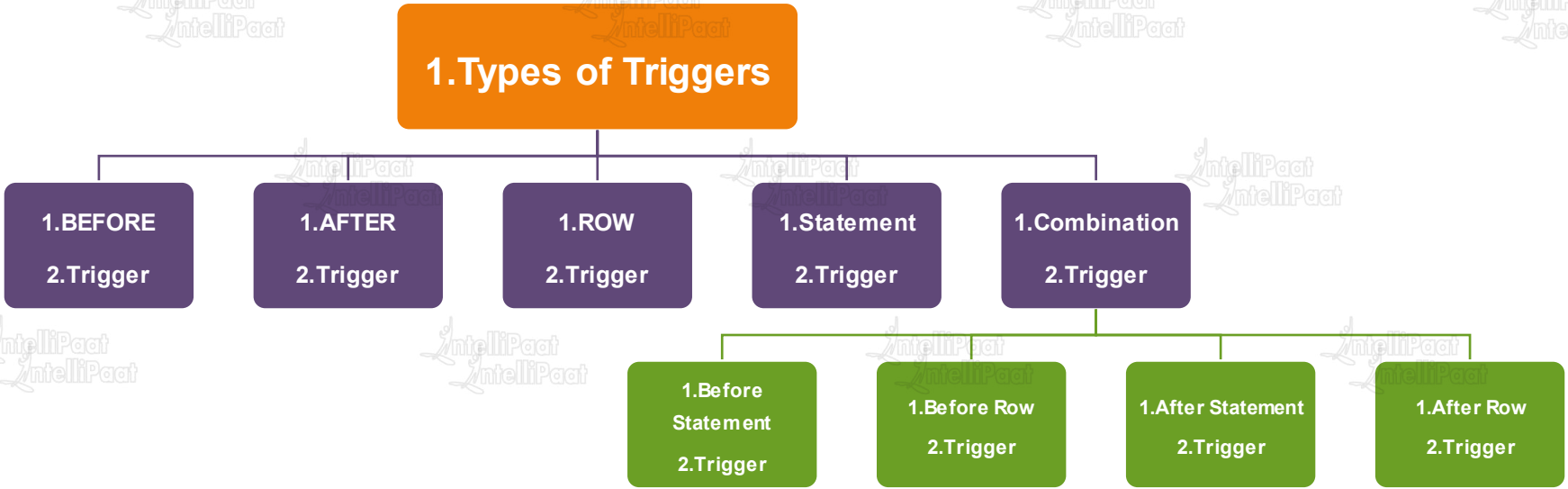
Trigger Action

When a triggering SQL statement is executed, trigger automatically calls and PL/SQL trigger blocks the execution

Trigger Restriction

We can specify the condition inside a trigger as when the trigger is to be fired

Types of Triggers



PL/SQL Triggers: Syntax

```
CREATE [OR REPLACE] TRIGGER trigger_name
   BEFORE | AFTER
   [INSERT, UPDATE, DELETE [COLUMN NAME..]
   ON table_name

   Referencing [ OLD AS OLD | NEW AS NEW ]
   FOR EACH ROW | FOR EACH STATEMENT [ WHEN
Condition ]

DECLARE
   [declaration_section
      variable declarations;
      constant declarations;
   ]

BEGIN
   [executable_section
      PL/SQL execute/subprogram body
   ]

EXCEPTION
   [exception_section
      PL/SQL Exception block
   ]

END;
```

Syntax Description

- **CREATE [OR REPLACE] TRIGGER trigger_name**
  - Creates a trigger with the given name. If it already exists, overwrite the existing trigger with the same defined name
- **BEFORE | AFTER**
  - Indicates when the trigger gets fired. BEFORE trigger executes before the statement executes. AFTER trigger executes after the statement executes
- **[INSERT, UPDATE, DELETE [COLUMN NAME]**
  - Determines the performing trigger event. You can define more than one triggering event separated by the OR keyword
- **ON table_name**
  - Assign a table name to the performing trigger event
- **Referencing [OLD AS OLD | NEW AS NEW]**
  - Give referencing to old and new values of data. :old means to use the existing row to perform the event and :new means to execute a new row to perform the event. You can set referencing names using old names or new user-defined names
    You can't reference old values when inserting a record or new values when deleting a record, because It does not exist
- **FOR EACH ROW | FOR EACH STATEMENT**
  - Trigger must be fired when each row gets affected (ROW Trigger) and be fired only once when the entire SQL statement is executed (STATEMENT Trigger)
- **WHEN Condition**
  - (Optional) Use only for row-level trigger. Trigger gets fired when a specified condition is satisfied.

PL/SQL Triggers: Example

Inserting a Trigger

```
CREATE or REPLACE TRIGGER trg1
   BEFORE
   INSERT ON emp1
   FOR EACH ROW
BEGIN
   :new.ename := upper(:new.ename);
END;
/
```

PL/SQL Triggers: Example

Restriction to Deleting a Trigger

```
CREATE or REPLACE TRIGGER trg1
   AFTER
   DELETE ON emp1
   FOR EACH ROW
BEGIN
   IF :old.eno = 1 THEN
      raise_application_error(-20015, 'You can't delete this row');
   END IF;
END;
/
```

Result

```
SQL>delete from emp1 where eno = 1;

Error Code: 20015

Error Name: You can't delete this row
```

Quiz

Quiz 1

What is the syntax for disabling a trigger?

A	DISABLE TRIGGER trigger_name;
B	ALTER TRIGGER trigger_name DISABLE;
C	ALTER TRIGGER DISABLE trigger_name;
D	Both a, b and c can be used

Quiz 1

What is the syntax for disabling a trigger?

A	DISABLE TRIGGER trigger_name;
B	ALTER TRIGGER trigger_name DISABLE;
C	ALTER TRIGGER DISABLE trigger_name;
D	Both a, b and c can be used

Quiz 2

Select the VALID trigger type(s)?

A	UPDATE row trigger
B	DELETE row trigger
C	INSERT row trigger
D	AFTER statement trigger

Answer 2

Select the VALID trigger type(s)?

A	UPDATE row trigger
B	DELETE row trigger
C	INSERT row trigger
D	AFTER statement trigger

Quiz 3

All objects placed in a package specification are called

A	Public objects
B	Private objects
C	None of the above
D	All of the above

Answer 3

All objects placed in a package specification are called

A	Public objects
B	Private objects
C	None of the above
D	All of the above

Quiz 4

Is it possible to open a cursor which is in a Package in another procedure ?

A	Yes
B	No

Answer 4

Is it possible to open a cursor which is in a Package in another procedure ?

A	Yes
B	No

Quiz 5

Is it possible to use Transactional control statements in Database Triggers ?

A	Yes
B	No

Answer 5

Is it possible to use Transactional control statements in Database Triggers ?

A	Yes
B	No

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor