|  |  |
| --- | --- |
| A picture containing logo  Description automatically generated | **DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING**  **MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO**  **Database Management Systems (4th Semester) 18CS**  **Lab Experiment 15** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll No:** |  | **Date of Conduct:** |  |
| **SubmissionDate:** |  | **Grade Obtained:** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Recognition (0.3)** | **Completeness & accuracy (0.4)** | **Timeliness (0.3)** | **Score (1.0)** |
|  |  |  |  |

**Objective:** **To understand the use of Triggers in PL/SQL.**

**Tools:** **MySql/oracle**.

**Introduction:**

**Triggers:** Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

* A **database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
* A **database definition (DDL)** statement (CREATE, ALTER, or DROP).
* A **database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP,

Or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

**Benefits of Triggers**

Triggers can be written for the following purposes −

* Generating some derived column values automatically
* Enforcing referential integrity
* Event logging and storing information on table access
* Auditing
* Synchronous replication of tables
* Imposing security authorizations
* Preventing invalid transactions

**Creating Triggers**

**The syntax for creating a trigger is −**

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

ON table\_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

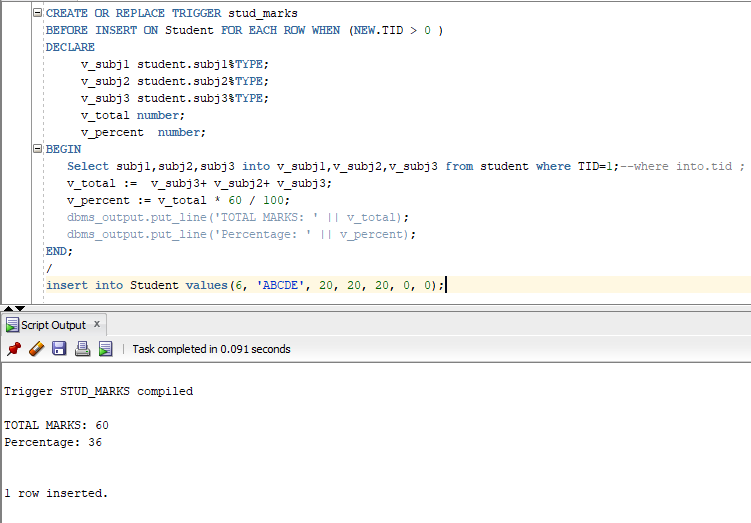
END;

Where,

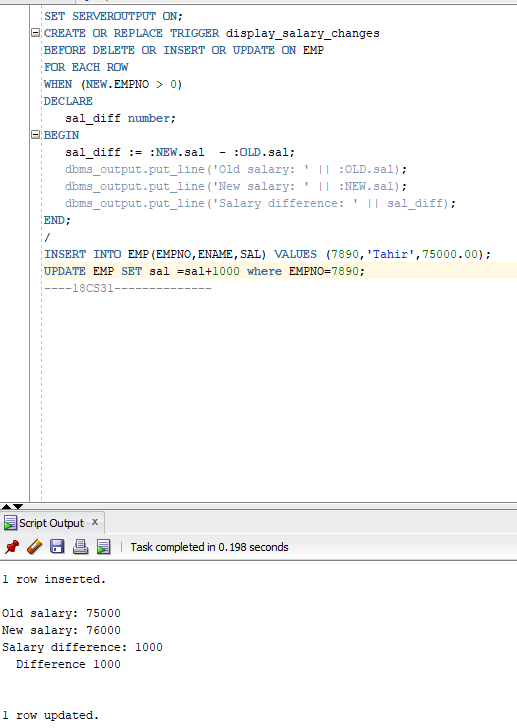
* CREATE [OR REPLACE] TRIGGER trigger\_name − Creates or replaces an existing trigger with the *trigger\_name*.
* {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
* {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
* [OF col\_name] − This specifies the column name that will be updated.
* [ON table\_name] − This specifies the name of the table associated with the trigger.
* [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
* [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
* WHEN (condition) − this provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

**LAB TASK**

**1. Consider Student table, in which student’s marks assessment is recorded. In such schema, create a trigger so that the total and percentage of specified marks is automatically inserted whenever a record is inserted.**

**Task:**

**2. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger should display the salary difference between the old values and new values.**

**Task:**