



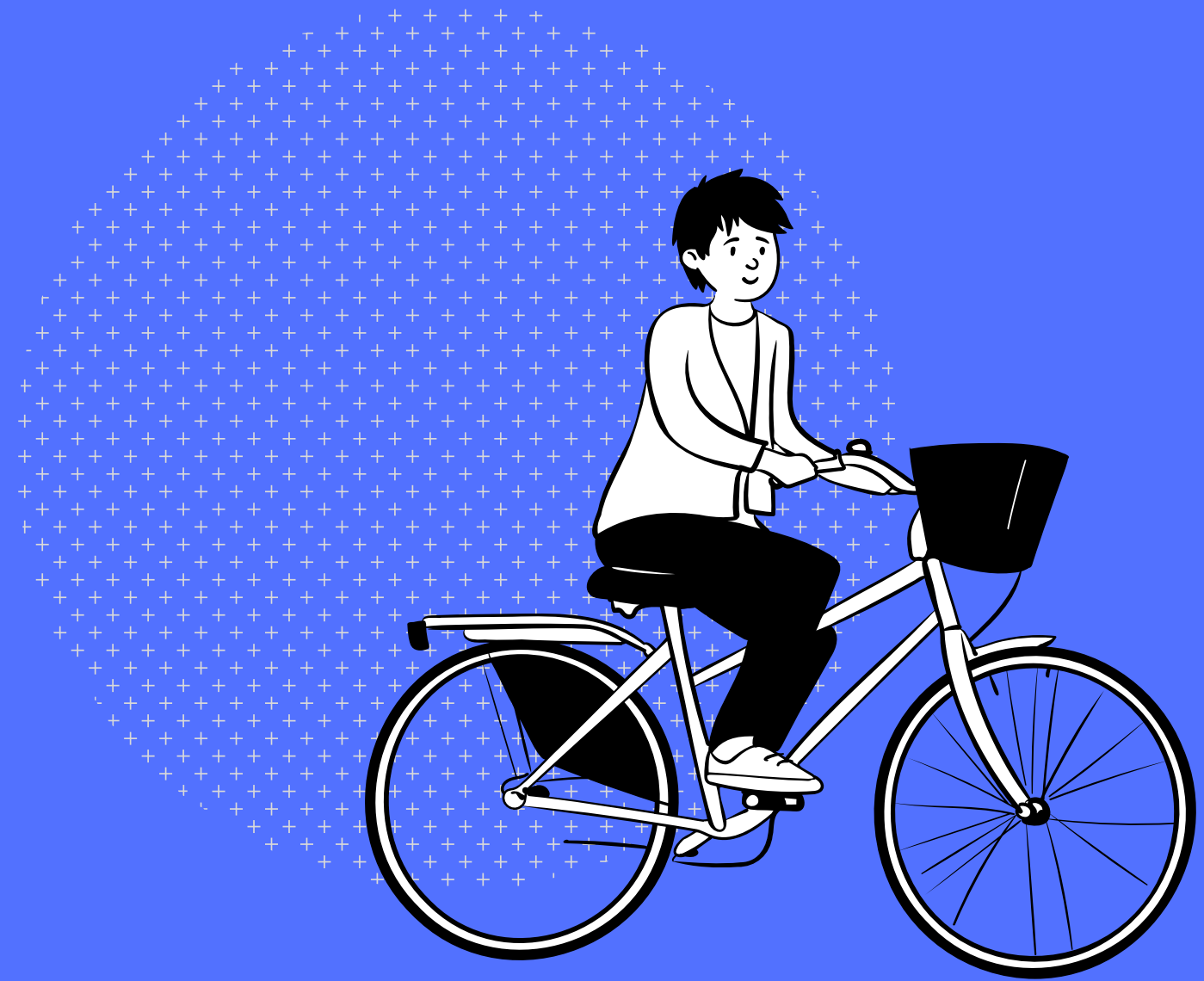
PL/SQL

TRIGGER

Presentation by



CONTENT INDEX



- ทริกเกอร์คืออะไร
- Triggering event
- ทริกเกอร์ใช้ทำอะไร
- Creating Triggers
- OLD and NEW Pseudorecords

- Example 1
- Example 2
- Challenge



ทริกเกอร์คืออะไร

Trigger หมายถึง โปรแกรม PL/SQL ที่อยู่ใน
ลักษณะของ Name Block ที่ถูกสร้างและจัดเก็บไว้
ในระบบฐานข้อมูล โดยการทำงานของ Database
Trigger นั้นจะทำงานเองโดยอัตโนมัติเมื่อมี
Trigger Events เกิดขึ้น





Triggering event

หมายถึง

การที่มีเหตุการณ์บางอย่างเกิดขึ้นกับ
table, schema, database หรือ system

เช่น มีการใช้คำสั่ง DML มากระทำต่อ table ที่เราได้เขียน
database trigger ผูกเอาไว้ ก็จะทำให้ database trigger นั้น
เกิดการทำงานโดยอัตโนมัติตามคำสั่ง PL/SQL ที่เราได้เขียนไว้
นั่นเอง

- **INSERT**

- **UPDATE**

- **DELETE**

ทริกเตอร์ใช้ทำอะไร

ทริกเตอร์ใช้ทำอะไร

ทริกเตอร์ใช้ทำอะไร



ทริคเตอร์ใช้ทำอะไร



Automatically

ช่วยสร้างค่าในคอลัมน์บางส่วนโดย
อัตโนมัติ



Event logging and storing information on table access

ใช้บันทึกเหตุการณ์ที่เกิดขึ้นใน
ตารางที่ได้รับการเข้าถึง



ทริคเตอร์ใช้ทำอะไร



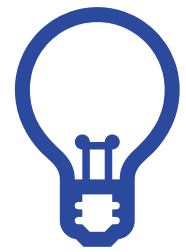
Preventing invalid transactions

ใช้ป้องกันการทำการที่ไม่ถูกต้อง



Auditing

ใช้ตรวจสอบข้อมูล



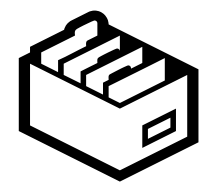
Creating Triggers

syntax

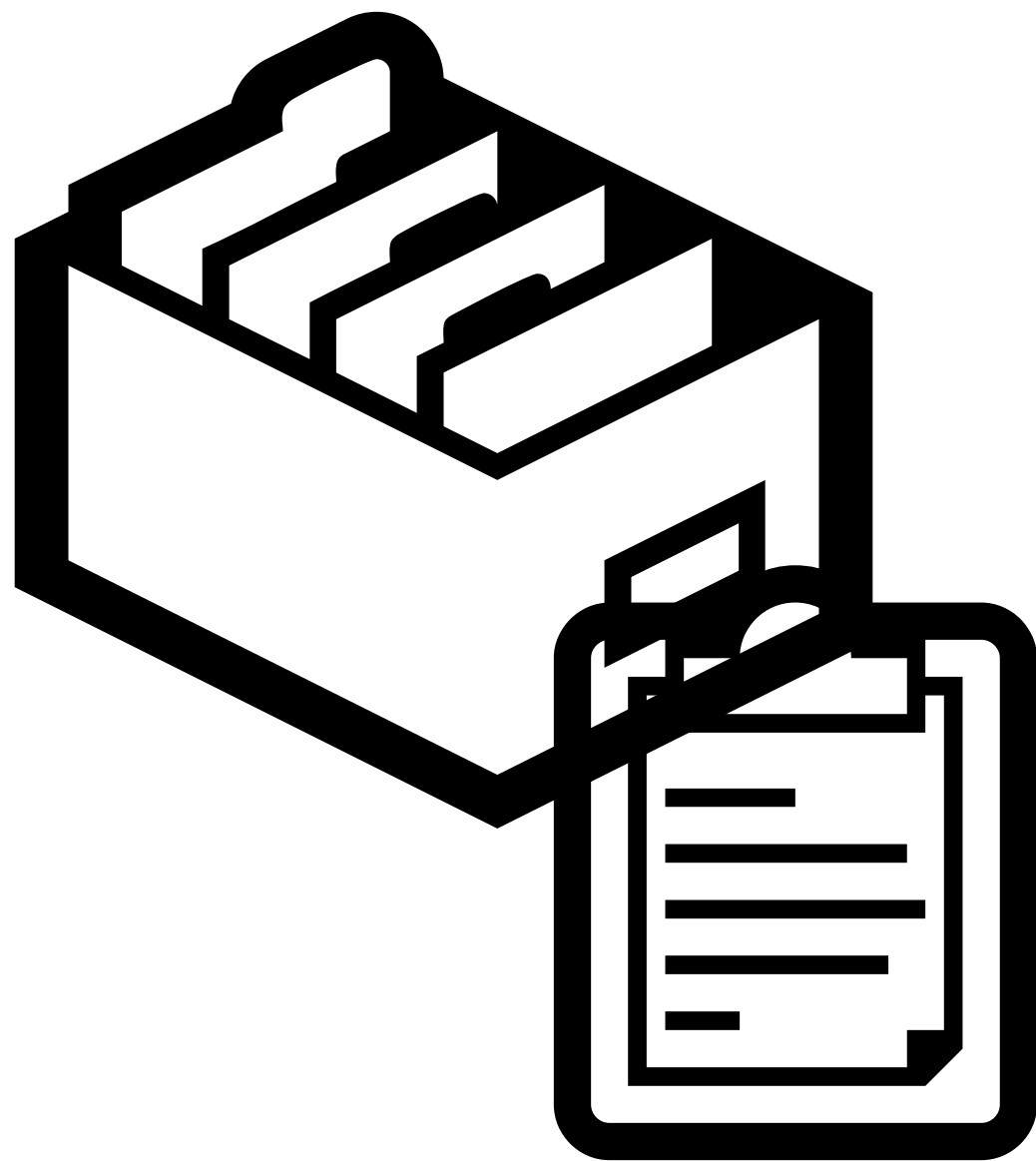
```
CREATE [OR REPLACE ] TRIGGER (trigger_name)
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
    Declaration-statements
BEGIN
    Executable-statements
EXCEPTION
    Exception-handling-statements
END;
```


Where,

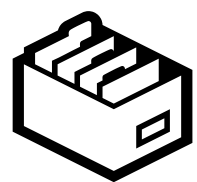
- **CREATE [OR REPLACE] TRIGGER (trigger_name)** – สร้างหรือแทนที่ทริกเกอร์ที่มีอยู่ด้วย (*trigger_name*)
- **{BEFORE | AFTER | INSTEAD OF}** – This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- **{INSERT [OR] | UPDATE [OR] | DELETE}** – 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.



OLD and NEW Pseudorecords



เมื่อทริกเกอร์เริ่มทำงาน
PL/SQL จะสร้างและอัปเดตบันทึกค่าสอง
รายการคือ **OLD** และ **NEW**
เรียกว่า **pseudorecords**



OLD and NEW Pseudorecords

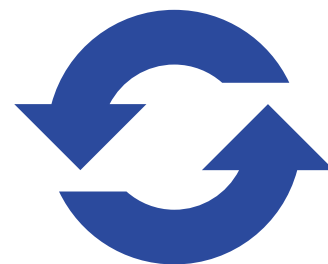


สำหรับทริกเกอร์ INSERT

OLD ไม่มีค่า

และ

NEW มีค่าใหม่

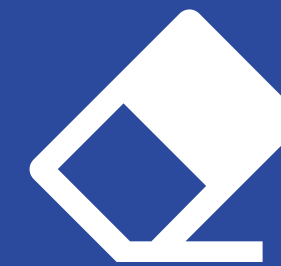


สำหรับทริกเกอร์ UPDATE

OLD จะเป็นค่าเก่า

และ

NEW จะเป็นค่าใหม่



สำหรับทริกเกอร์ DELETE

OLD ไม่มีค่า

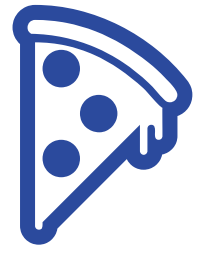
และ

NEW มีค่าใหม่

 **Download Source code**

<https://github.com/Mickey4527/Trigger-and-Package>

ข้าก็ขอเขียนโหลดเลย !

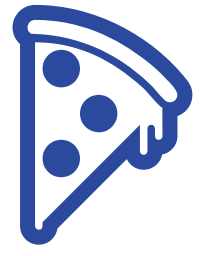


Example 1

สมมุติว่า ...

นายมิกกี้ ต้องการทราบว่า เงินเดือนของพนักงานในตาราง Employees เมื่อมีการเปลี่ยนแปลงแล้ว (ไม่ว่าจะเป็น INSERT หรือ UPDATE หรือ DELETE) ให้ show output เงินเดือนเก่าเป็นเท่าไร เงินเดือนใหม่เป็นเท่าไร และความต่างของเงินเก่ากับใหม่เป็นเท่าไร

🤔 นายมิกกี้ควรทำอะไร ?



Example 1

สมมุติว่า ...

นายมิกกี้ ต้องการทราบว่า เงินเดือนของพนักงานในตาราง Employees เมื่อมีการเปลี่ยนแปลงแล้ว (ไม่ว่าจะเป็น INSERT หรือ UPDATE หรือ DELETE) ให้ show output เงินเดือนเก่าเป็นเท่าไร เงินเดือนใหม่เป็นเท่าไร และความต่างของเงินเก่ากับใหม่เป็นเท่าไร

🤖 นายมิกกี้ควรทำอะไร ?

Step 1



```
1  CREATE OR REPLACE TRIGGER display_salary_changes
2  BEFORE DELETE OR INSERT OR UPDATE ON Employees
3  FOR EACH ROW
4  WHEN (NEW.Employee_ID > 0)
5  DECLARE
6      sal_diff number;
7  BEGIN
8      sal_diff := :NEW.salary - :OLD.salary;
9      dbms_output.put_line('Old salary: ' || :OLD.salary);
10     dbms_output.put_line('NEW salary: ' || :NEW.salary);
11     dbms_output.put_line('Salary difference: ' || sal_diff);
12 END;
13 /
14
```

Step 2



```
1  SET SERVEROUTPUT ON;
```

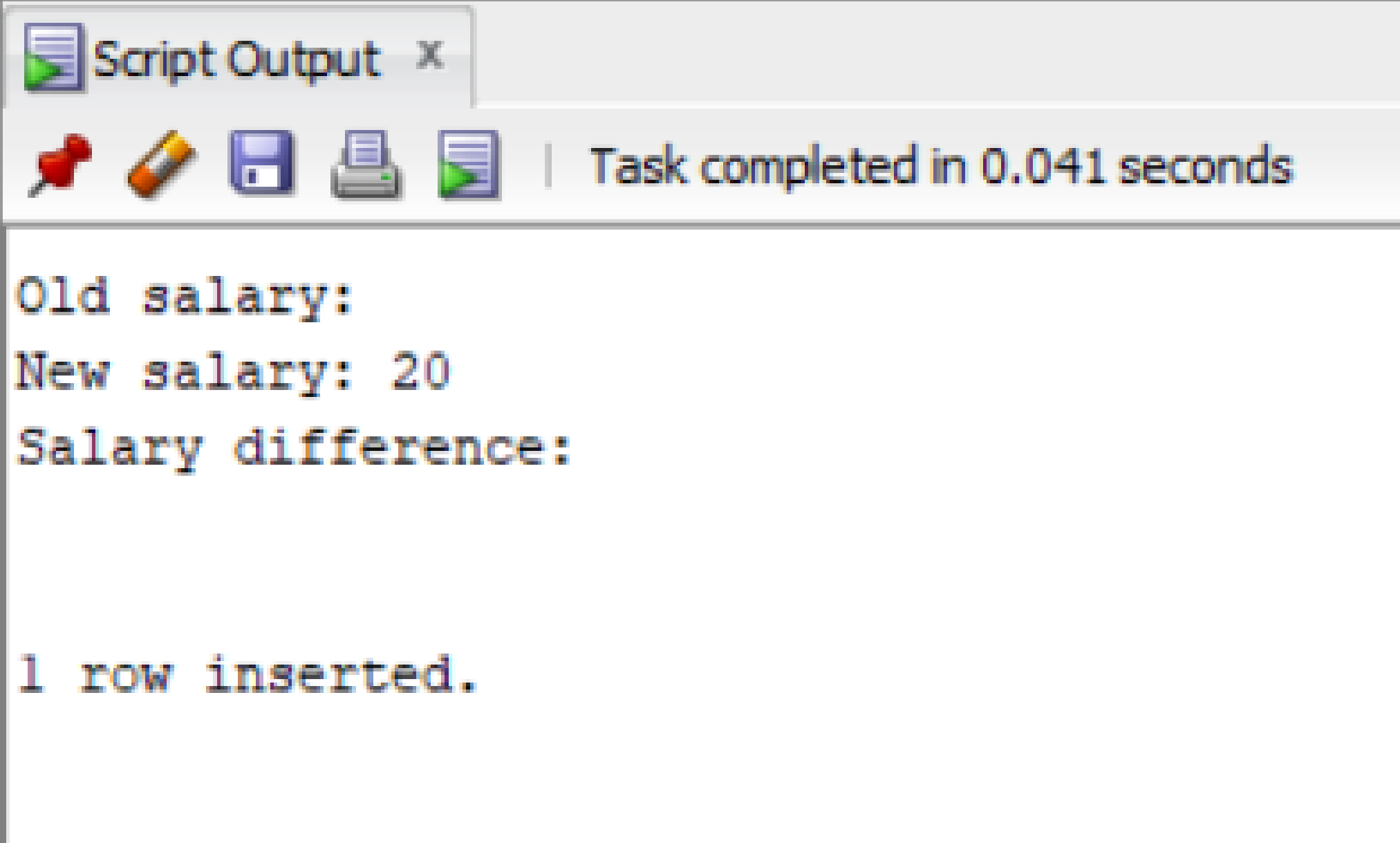

Try it!



```
1  INSERT INTO employees VALUES (employees_seq.nextval, 'Aman', 'Pooja', '123@gmail0com', '0001112222', sysdate, 'AD_PRES', '20', '', '100', '30');
```

ANSWER?

ANSWER

A screenshot of a 'Script Output' window. The window has a title bar with a green play button icon, the text 'Script Output', and a close button 'x'. Below the title bar is a toolbar with icons for a red pushpin, an eraser, a save icon, a printer, and a document with a green play button. To the right of the toolbar, it says 'Task completed in 0.041 seconds'. The main area of the window contains the following text:

```
Old salary:  
New salary: 20  
Salary difference:  
  
1 row inserted.
```

```
Script Output x  
Task completed in 0.041 seconds  
Old salary:  
New salary: 20  
Salary difference:  
  
1 row inserted.
```

ແລ້ວຕໍ່າ UPDATE ລ່ະ ?

TRY IT!



```
1  UPDATE employees SET salary = 2000
2  WHERE first_name = 'Aman';
```

ANSWER

```
Old salary: 20
```

```
New salary: 2000
```

```
Salary difference: 1980
```

```
1 row updated.
```

ແລ້ວຕໍ່າ DELETE ລະ ?

TRY IT!



```
1  DELETE FROM employees
2  WHERE first_name = 'Aman';
```


ANSWER

```
1 row deleted.
```

WHY?



```
1  CREATE OR REPLACE TRIGGER display_salary_changes
2  BEFORE DELETE OR INSERT OR UPDATE ON Employees
3  FOR EACH ROW
4  WHEN (NEW.Employee_ID > 0)
5  DECLARE
6      sal_diff number;
7  BEGIN
8      sal_diff := :NEW.salary - :OLD.salary;
9      dbms_output.put_line('Old salary: ' || :OLD.salary);
10     dbms_output.put_line('NEW salary: ' || :NEW.salary);
11     dbms_output.put_line('Salary difference: ' || sal_diff);
12 END;
13 /
14
```



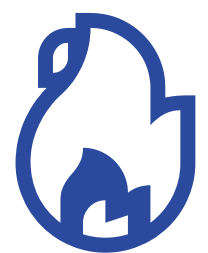
```
1  CREATE OR REPLACE TRIGGER display_salary_changes
2  BEFORE DELETE OR INSERT OR UPDATE ON Employees
3  FOR EACH ROW
4  WHEN (NEW.Employee_ID > 0)
5  DECLARE
6      sal_diff number;
7  BEGIN
8      sal_diff := :NEW.salary - :OLD.salary;
9      dbms_output.put_line('Old salary: ' || :OLD.salary);
10     dbms_output.put_line('NEW salary: ' || :NEW.salary);
11     dbms_output.put_line('Salary difference: ' || sal_diff);
12 END;
13 /
14
```

A red arrow points from the word `NEW` on line 4 to two large red question marks (`??`).



```
1 CREATE OR REPLACE TRIGGER display_salary_changes
2 BEFORE DELETE OR INSERT OR UPDATE ON Employees
3 FOR EACH ROW
4 WHEN (NEW.Employee_ID > 0)
5 DECLARE
6     sal_diff number;
7 BEGIN
8     sal_diff := :NEW.salary - :OLD.salary;
9     dbms_output.put_line('Old salary: ' || :OLD.salary);
10    dbms_output.put_line('NEW salary: ' || :NEW.salary);
11    dbms_output.put_line('Salary difference: ' || sal_diff);
12 END;
13 /
14
```

เมื่อลบค่าใหม่จึงเป็น **NEW= 0**



Example 2

สมมุติว่า ...

นายเจ ต้องการทำ Trigger ที่สามารถทราบว่าในแต่ละวันมีการเปลี่ยนแปลงอะไรบ้างในตาราง Employees แล้วเปลี่ยน วันไหน, ใครเปลี่ยน



นายเจควรทำอย่างไร ?

Step 1

Create table



```
1 CREATE TABLE EMPLOYEES_LOG (LOG_DATE DATE , action VARCHAR2(50) , BY_USER VARCHAR2(20));
```

Step 2

Create Trigger



```
1  CREATE OR REPLACE TRIGGER EMP_CHANGE_TRIGGER
2  AFTER INSERT OR UPDATE OR DELETE ON EMPLOYEES
3  DECLARE
4      log_action EMPLOYEES_LOG.action%TYPE;
5  BEGIN
6      IF INSERTING THEN
7          log_action := 'Insert';
8      ELSIF UPDATING THEN
9          log_action := 'Update';
10     ELSIF DELETING THEN
11         log_action := 'Delete';
12     ELSE
13         DBMS_OUTPUT.PUT_LINE('This code is not reachable.');
```


TRY IT!



```
1  INSERT INTO employees VALUES (employees_seq.nextval, 'Aman', 'Pooja', '123@gmail0com', '0001112222', sysdate, 'AD_PRES', '20', '', '100', '30');
```

<div> <div> Welcome Page </div> <div> saran </div> <div> EMPLOYEES_LOG </div> </div>				
<div> <div>Columns</div> <div>Data</div> <div>Model</div> <div>Constraints</div> <div>Grants</div> <div>Statistics</div> <div>Triggers</div> <div>Fla</div> </div>				
<div> <div>Sort..</div> <div>Filter:</div> </div>				
	LOG_DATE	ACTION	BY_USER	
1	26-SEP-23	Insert	HR	

Challenge

นาย ก. ตูณพูดว่า “นักศึกษามีกิน มีใช้ มีเกียรติ มีศักดิ์ศรี” แต่นาย ก. ตูณไม่สามารถทำได้
อย่างที่พูดไว้ ทำให้นักศึกษาแหว่ไป comment ในระบบ ว่า “Trabatsat” (ตระบัดสัตย์)
ฯลฯ นาย ก. ตูณไม่พอใจที่นักศึกษากล่าวว่าตน เลยบ่นให้กับนักศึกษาเจ (ผู้ซึ่งไม่รู้
อีโหน่อีเหน่) ว่า “ โทๆเป็นผู้ใหญ่กันแล้ว อย่าใช้วาทกรรมด้อยค่ากัน” เมื่อนายตูณบ่น
นักศึกษาเจเสร็จแล้ว นายตูณเลยคิดว่า... ต้องการสร้าง Trigger แทนที่คำว่า “Trabatsat”
(ตระบัดสัตย์) เป็นคำว่า “ManKhong ManKhang Yangyuen” (มันคง มันคง ยั้งยั้ง) ,
คำว่า “Nai kor Som Lon” และ “Nai kor Pita” เป็น “Nai kor Thanawat” เมื่อมีการ
INSERT หรือ UPDATE ลงในตาราง COMMENTS

Challenge

Keyword

เมื่อมีการ INSERT หรือ UPDATE หากมีคำว่า “Trabatsat” (ตระกูลสัตย์)
แทนที่เป็นคำว่า “ManKhong ManKhang Yangyuen” และ หากมีคำว่า
“Nai kor Som Lon” และ “Nai kor Pita” เป็น “Nai kor Thanawat”

Step 1

Create table



```
1 CREATE TABLE COMMENTS(  
2     COMMENTS_TITLE VARCHAR2(256)  
3 );
```

INSERT



```
1 INSERT INTO COMMENTS VALUES ('Trabadsat');
```

Answer



```
1 CREATE OR REPLACE TRIGGER TRG_CONVERT_WORDS
2 BEFORE INSERT OR UPDATE ON COMMENTS
3 FOR EACH ROW
4 BEGIN
5     :NEW.COMMENTS_TITLE := REPLACE(UPPER(:NEW.COMMENTS_TITLE), 'TRABADSATY', 'ManKhong ManKhang Yangyuen');
6     :NEW.COMMENTS_TITLE := REPLACE(UPPER(:NEW.COMMENTS_TITLE), 'NAI KOR SOM LON', 'Nai Kor Thanawat');
7     :NEW.COMMENTS_TITLE := REPLACE(UPPER(:NEW.COMMENTS_TITLE), 'NAI KOR PITA', 'NaI Kor Thanawat');
8 END;
9 /
```



PL/SQL

Packages



Presentation by



CONTENT INDEX



- แฝกคืออะไร
- ส่วนประกอบของแฝก

- Example - 1
- Example - 2

แพคเกจคืออะไร

Package คือ ส่วนของ Schema Object ที่
รวบรวม **PROCEDURE** หรือ **FUNCTION** หลายๆ ตัวไว้
ด้วยกัน เพื่อง่ายต่อการควบคุมทั้งแบ่งกลุ่มของการทำงาน
การจัดการ การเข้าถึง





ซึ่งถ้าจะเข้าใจให้ง่าย ๆ มันคือ **Class** ที่ประกอบด้วยหลาย ๆ **Method**
และ Function โดยในรูปแบบของ Method ก็จะแยกย่อยการทำงานที่
แตกต่างกันไป



ส่วนประกอบแพ็คเกจ

ใน **Package** บน **Oracle Database** จะสามารถบรรจุรายการได้ 2 ประเภทคือ

PROCEDURE

- ไม่มีการคืนค่าตามสิ่งที่ทำให้ใช้สำหรับการประมวลผลหรือการดำเนินการที่ไม่ได้ส่งค่าออก
- PROCEDURE ใช้สำหรับการปรับปรุงข้อมูลในฐานข้อมูล

FUNCTION

- จะคืนค่าหลังจากการทำงาน ซึ่งค่าที่คืนมาสามารถนำไปใช้ในคำสั่ง SQL หรือโปรแกรม PL/SQL ได้
- ใช้สำหรับการคำนวณค่า, การสืบค้นข้อมูลและคืนค่า, หรือการประมวลผลที่จำเป็นต้องคืนค่าออกมา



ภายในหนึ่ง **Package** จะสามารถมีได้หลาย ๆ **PROCEDURE** รวมทั้งหลาย ๆ **FUNCTION**



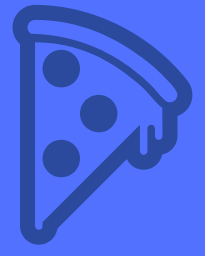
ส่วนประกอบแพ็คเกจ

การใช้งาน Package ในส่วนของ **PROCEDURE/FUNCTION** จะถูกแบ่งออกเป็น 2 ส่วน

- **Header**
- **Body**

Header คือการประกาศชื่อ **PROCEDURE** และ **FUNCTION** รวมทั้ง Argument ต่าง ๆ

Body คือส่วนที่จะสั่งให้ **PROCEDURE** และ **FUNCTION** นั้นทำงานต่าง ๆ



Example - 1

Package Header



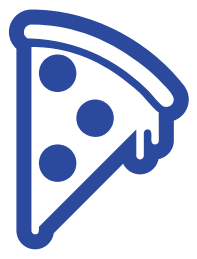
```
1  CREATE OR REPLACE PACKAGE my_package AS
2      var1 NUMBER;
3      var2 VARCHAR2(50);
4
5      PROCEDURE pro1;
6      FUNCTION fun1(param1 NUMBER);
7  END my_package;
8  /
```



Correct



```
1  CREATE OR REPLACE PACKAGE my_package AS
2      var1 NUMBER;
3      var2 VARCHAR2(50);
4
5      PROCEDURE pro1;
6      FUNCTION fun1(param1 NUMBER) RETURN NUMBER;
7  END my_package;
8  /
9
```



Example - 1

Body

```
1 CREATE OR REPLACE PACKAGE BODY my_package AS
2     var1 NUMBER := 0;
3     var2 VARCHAR2(50) := 'Hello';
4
5     PROCEDURE pro1 IS
6     BEGIN
7         DBMS_OUTPUT.PUT_LINE('Procedure 1 is called');
8     END;
9
10    FUNCTION fun1(param1 NUMBER) RETURN NUMBER IS
11    BEGIN
12        RETURN param1*10;
13    END;
14 END my_package;
15 /
```

วิธีเรียกใช้

{__(ชื่อ Package)__.__(ชื่อ FUNCTION หรือ PROCEDURE ตามด้วย())__};



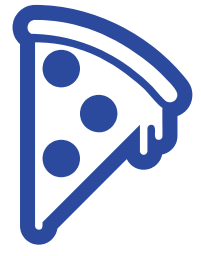
```
1 my_package.fun1(5);
```


ลองเรียกใช้



```
1 BEGIN
2     DBMS_OUTPUT.PUT_LINE('Function result: ' || my_package.fun1(5));
3 END;
4 /
```

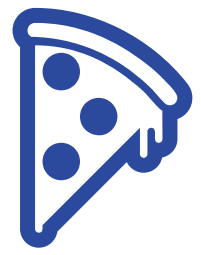
Function result: 50



Example - 2

ในตาราง Employees จะข้อมูลจำนวนหนึ่ง ซึ่งในตารางจะมีข้อมูลคือ Employee ID, Firstname, Lastname, Email, phone number, hire date, job id salary COMMISSION_PCT, MANAGER_ID และ DEPARTMENT_ID

สมมุติว่า ...



Example - 2

เราได้รับคำสั่งจากลูกค้าว่าให้ทำโค้ดที่สามารถเพิ่มข้อมูลพนักงานได้,
ลบข้อมูลพนักงานได้, อัปเดตเงินเดือนและอัปเดต commission โดยมีเงื่อนไขว่า

1. เมื่อเพิ่มข้อมูลพนักงาน employee_id ให้ใช้ sequence nextval() และวันที่เป็นวันที่เพิ่มข้อมูล (sysdate)
2. เมื่อลบข้อมูลจะใช้ข้อมูลแค่ Employee ID เท่านั้น
3. อัปเดตเงินเดือนจะใช้ข้อมูลแค่ Employee ID และ salary เท่านั้น
4. ต้องใช้เป็น Package เท่านั้น

🤖 เราในฐานะเด็กฝึกงานควรทำอย่างไร ?

Package Header

```
1  CREATE OR REPLACE PACKAGE employee_mana AS
2      FUNCTION add_employee(first_name VARCHAR2,
3                             last_name VARCHAR2,
4                             email VARCHAR2,
5                             PHONE_NUMBER VARCHAR2,
6                             JOB_ID VARCHAR2,
7                             salary NUMBER,
8                             COMMISSION_PCT NUMBER,
9                             MANAGER_ID NUMBER,
10                             DEPARTMENT_ID NUMBER)
11          RETURN NUMBER;
12  PROCEDURE remove_employee(employee_id NUMBER);
13  PROCEDURE update_employee_salary(employee_id NUMBER,
14                                   salary NUMBER);
15  PROCEDURE update_employee_commission_pct(employee_id NUMBER,
16                                             commission_pct NUMBER);
17  END employee_mana;
18  /
```

Package Body + Add Employee

```
1  CREATE OR REPLACE PACKAGE BODY employee_mana AS
2      /*add employee*/
3      FUNCTION add_employee(first_name VARCHAR2,
4                             last_name VARCHAR2,
5                             email VARCHAR2,
6                             PHONE_NUMBER VARCHAR2,
7                             JOB_ID VARCHAR2,
8                             salary NUMBER,
9                             COMMISSION_PCT NUMBER,
10                             MANAGER_ID NUMBER,
11                             DEPARTMENT_ID NUMBER)
12          RETURN NUMBER IS
13      new_emp_id NUMBER;
14      BEGIN
15          new_emp_id := employees_seq.NEXTVAL;
16          INSERT INTO employees
17          VALUES (new_emp_id,
18                  first_name,
19                  last_name,
20                  email,
21                  PHONE_NUMBER,
22                  SYSDATE,
23                  JOB_ID,
24                  salary,
25                  COMMISSION_PCT,
26                  MANAGER_ID,
27                  DEPARTMENT_ID);
28          dbms_output.put_line('New employee ID: ' || new_emp_id);
29          RETURN new_emp_id;
30      END add_employee;
```

Remove Employee



```
1  PROCEDURE remove_employee(employee_id NUMBER) IS
2      BEGIN
3          DELETE FROM employees
4          WHERE employee_id = remove_employee.employee_id;
5          dbms_output.put_line('Employee ID: ' || remove_employee.employee_id || ' has been removed');
6      END remove_employee;
```

Update Salary



```
1  PROCEDURE update_employee_salary(employee_id NUMBER,  
2                                     salary NUMBER) IS  
3      BEGIN  
4          UPDATE employees  
5          SET salary = update_employee_salary.salary  
6          WHERE employee_id = update_employee_salary.employee_id;  
7      END update_employee_salary;
```

Update commission



```
1  PROCEDURE update_employee_commission_pct(employee_id NUMBER,  
2                                          commission_pct NUMBER) IS  
3  BEGIN  
4      UPDATE employees  
5      SET commission_pct = update_employee_commission_pct.commission_pct  
6      WHERE employee_id = update_employee_commission_pct.employee_id;  
7  END update_employee_commission_pct;  
8  
9  END employee_mana;  
10 /
```


ทันงง! au TRIGGER หน่อย

DROP TRIGGER display_salary_changes;

Try it!



```
1  /*test*/
2  DECLARE
3      new_emp_id NUMBER(6);
4  BEGIN
5      new_emp_id := employee_mana.add_employee('John', 'Doe', 'John@mail.com', '123456789', 'IT_PROG', 10000, 0.1, 100, 90);
6      dbms_output.put_line('new_emp_id = ' || new_emp_id);
7      employee_mana.update_employee_salary(new_emp_id, 20000);
8      employee_mana.update_employee_commission_pct(new_emp_id, 0.2);
9      employee_mana.remove_employee(new_emp_id);
10 END;
11 /
```

Result

au trigger

```
New employee ID: 225  
new_enp_id = 225  
Employee ID: 225 has been removed  
  
PL/SQL procedure successfully completed.
```

Trigger

```
Old salary:  
New salary: 10000  
Salary difference:  
New employee ID: 224  
new_enp_id = 224  
Old salary: 10000  
New salary: 20000  
Salary difference: 10000  
Old salary: 20000  
New salary: 20000  
Salary difference: 0  
Employee ID: 224 has been removed  
  
PL/SQL procedure successfully completed.
```

Thank you