www.itsci.mju.ac.th/sayan

LEC 11: DATABASE TRIGGER

SAYAN UNANKARD

1/2558

WHAT IS DATABASE TRIGGER ?

- Database Trigger หมายถึงโปรแกรม PL/SQL ที่อยู่ในลักษณะของ Named block ที่ถูกสร้างและถูกจัดเก็บอยู่ในระบบฐานข้อมูล Oracle โดยการทำงานของ Database Trigger นั้นจะทำงานเองเมื่อมี Trigger Event เกิดขึ้น
- Trigger Event หมายถึง การที่มีคำสั่ง DML คือ INSERT UPDATE หรือ DELETE กระทำกับตารางบนระบบฐานข้อมูล โดยที่ตารางดังกล่าวเป็นตารางที่เราสร้าง Database Trigger ไว้นั่นเอง

DATABASE TRIGGER SYNTAX

Trigger จะประกอบด้วยคำสั่ง SQL statements โดยจะรันเมื่อมีการกำหนด timing ตามเงื่อนไขที่กำหนดไว้ เมื่อมีเหตุการณ์ event1 หรือ event2 หรือ event3 ในตาราง table_name ที่กำหนด

```
CREATE [OR REPLACE] TRIGGER trigger_name
timing
event1 [OR event2 OR event3]
ON object_name
[[REFERENCING OLD AS old | NEW AS new]
FOR EACH ROW
[WHEN (condition)]]
BEGIN
  SQL statements;
END
```

BEFORE TRIGGER

การทำงานของ Trigger จะประมวลผลก่อนที่คำสั่ง DML จะทำงาน เช่น หากต้องการให้แสดงชื่อผู้ใช้ที่เข้ามาทำคำสั่ง DML ก่อนการ Insert และ Update

```
CREATE OR REPLACE TRIGGER show_user_trg
BEFORE INSERT OR UPDATE
ON EMPLOYEES
DECLARE
USER_NAME VARCHAR2(30);
BEGIN
   SELECT USER INTO USER_NAME FROM DUAL;
   DBMS_OUTPUT.PUT_LINE(USER_NAME);
END;
```

BEFORE TRIGGER

หากมีการทำงานคำสั่ง INSERT หรือ UPDATE โดยที่ Trigger จะทำงานก่อนที่จะทำคำสั่งดังกล่าว (ผู้ใช้ที่เข้าใช้งานคือ HR)

```
INSERT INTO EMPLOYEES VALUES(1234,'Sayan', 20000);

ผลลัพธ์ที่ได้คือ
HR
1 row created.
```

```
UPDATE EMPLOYEES SET SALARY = 25000 WHERE EMPID = 1234;

ผลลัพธ์ที่ได้คือ
HR
1 row updated.
```

AFTER TRIGGER

การทำงานของ Trigger จะประมวลผลก่อนที่คำสั่ง DML จะทำงาน เช่น หากต้องการให้แสดงวันและเวลา ผู้ใช้ที่เข้ามาทำคำสั่ง DML หลังจากการ Insert และ Update

```
CREATE OR REPLACE TRIGGER show_user_trg
AFTER INSERT OR UPDATE
ON EMPLOYEES
DECLARE
TIME_DML VARCHAR2(50);
BEGIN
   SELECT TO_CHAR(SYSDATE,'DD-MM-YY HH:MM:SS') INTO
   TIME_DML FROM DUAL;
   DBMS_OUTPUT.PUT_LINE(TIME_DML);
END;
```

AFTER TRIGGER

หากมีการทำงานคำสั่ง INSERT หรือ UPDATE โดยที่ Trigger จะทำงานหลังจากที่จะทำคำสั่งดังกล่าว (ผู้ใช้ที่เข้าใช้งานคือ HR)

```
INSERT INTO EMPLOYEES VALUES(1234,'Sayan', 20000);

ผลลัพธ์ที่ได้คือ
HR
15-05-08 12:35:45
1 row created.
```

```
UPDATE EMPLOYEES SET SALARY = 25000 WHERE EMPID = 1234;

ผลลัพธ์ที่ได้คือ
HR
15-05-08 12:37:25
1 row updated.
```

CONDITIONAL PREDICATES WITH DATABASE TRIGGER

ในกรณีที่สร้าง Trigger สำหรับรองรับการทำงานหลาย ๆ กรณีของคำสั่ง DML เช่น INSERT หรือ UPDATE หรือ DELETE ในกรณีที่ต้องการทราบว่าแต่ละครั้งที่ Trigger ทำงานนั้น ๆ เป็นคำสั่ง INSERT หรือ UPDATE หรือ DELETE กันแน่

```
IF INSERTING THEN
    DBMS_OUTPUT.PUT_LINE('you are Insert data...');
END IF;
IF UPDATING THEN
    DBMS_OUTPUT.PUT_LINE('you are Update data...');
END IF;
IF DELETING THEN
    DBMS_OUTPUT.PUT_LINE('you are Delete data...');
END IF;
```

CONDITIONAL PREDICATES WITH DATABASE TRIGGER EXAMPLE

ต้องการให้เก็บประวัติการทำงานลงในตารางที่ชื่อว่า

```
CREATE OR REPLACE TRIGGER AUDITS_TRG
AFTER INSERT OR UPDATE OR DELETE
ON JOBS
DECLARE
    T_USER_NAME VARCHAR2(50);
    T_DML_DATE DATE;
    T_DML_TYPE VARCHAR2(50);
BEGIN
    SELECT USER INTO T_USER_NAME FROM DUAL;
    SELECT SYSDATE INTO T_DML_DATE FROM DUAL;
    IF INSERTING THEN
          T_DML_TYPE := 'Insert';
    ELSIF UPDATING THEN
          T_DML_TYPE := 'Update';
    ELSE  T_DML_TYPE := 'Delete';
    END IF;
    INSERT INTO AUDITS VALUES(T_USER_NAME,T_DML_DATE,T_DML_TYPE);
END;
```

STATEMENT LEVEL TRIGGER

หมายถึงการที่เรามี Database Trigger อยู่ไม่ว่าจะเป็นรูปแบบ Before หรือ After ก็ตาม เมื่อมีการใช้คำสั่ง DML หนึ่ง ๆ ที่ทำให้ Database Trigger หนึ่ง ๆ ทำงาน Trigger ดังกล่าวจะทำงานเพียงแค่หนึ่งหนเท่านั้น โดยทั่วไป Database Trigger ที่เราสร้างขึ้นนั้นจะถูกกำหนดให้เป็นแบบ Statement Level Trigger

ROW LEVEL TRIGGER

Database Trigger แบบนี้จะมีจำนวนครั้งของการทำงานเท่ากับจำนวนของเรคคอร์ด ที่มีผลจากการกระทำของคำสั่ง DML หนึ่ง ๆ ที่ทำให้ Database Trigger ทำงาน

การสร้าง Row Level Trigger จะมีคำสั่งเพิ่มเติมคือคำสั่ง FOR EACH ROW

เช่น ทุก ๆ ครั้งที่มีการแก้ไขข้อมูลด้วยคำสั่ง UPDATE ซึ่งใน 1 คำสั่งนั้น ๆ อาจจะมีผลกับการแก้ไขข้อมูลหลาย ๆ แถว Trigger ก็จะมีการทำงานตามจำนวนแถวที่มีการแก้ไขข้อมูล

```
UPDATE EMPLOYEES SET SALARY = SALARY + 100 WHERE SALARY < 20000
```

ROW LEVEL TRIGGER EXAMPLE

ต้องการให้เก็บประวัติการทำงานลงในตารางที่ชื่อว่า

EMP_HISTORY(EMP_ID, EMP_NAME, SALARY, UPDATE_DATE)

```
CREATE OR REPLACE TRIGGER EMPHISTORY_TRG
AFTER UPDATE OR DELETE
ON EMPLOYEES
FOR EACH ROW
DECLARE
    T_UPDATE_DATE DATE;
BEGIN
    SELECT SYSDATE INTO T_UPDATE_DATE FROM DUAL;
    INSERT INTO EMP_HISTORY VALUES(:OLD.EMP_ID,
   :OLD.EMP_NAME,:OLD.SALARY,T_UPDATE_DATE);
END;
```

UPDATE EMPLOYEES SET SALARY = SALARY + 100 WHERE SALARY < 20000

UPDATE OF COLUMN

ในคำสั่งการสร้าง Database Trigger ตรงส่วนของการระบุถึง event ที่ทำให้เกิดการทำงานของ Database Trigger นั่นคำสั่ง ที่เราระบุนั้นอาจเป็น INSERT หรือ DELETE หรือ UPDATE

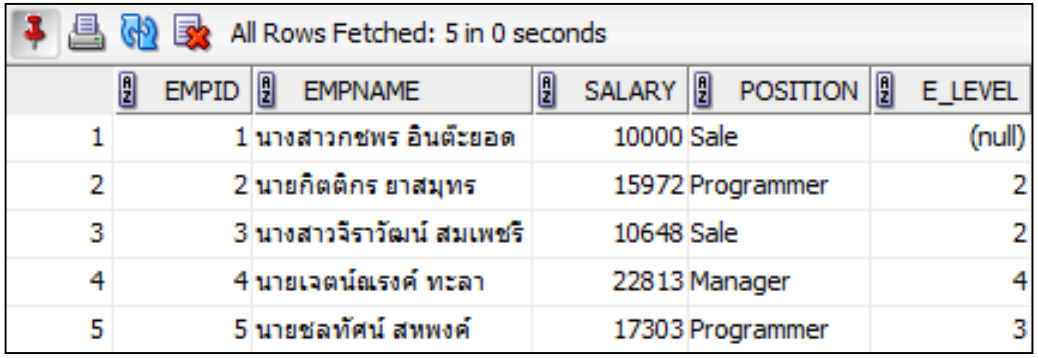
หลังคำสั่ง UPDATE คือชื่อของคอลัมน์ที่เมื่อถูกเปลี่ยนแปลงข้อมูลด้วยคำสั่ง UPDATE แล้วทำให้ Database Trigger เกิดการทำงานขึ้น

```
CREATE OR REPLACE TRIGGER UPDATE_COL_TRG
BEFORE INSERT OR UPDATE OF SALARY
ON EMPLOYEES
FOR EACH ROW
BEGIN
    IF :NEW.SALARY < :OLD.SALARY THEN
       :NEW.SALARY := :OLD.SALARY;
    END IF;
END;
```

UPDATE EMPLOYEES SET SALARY = 18000 WHERE SALARY < 20000

UPDATE OF COLUMN

```
UPDATE EMPLOYEE SET SALARY = 100 WHERE EMPID =1;
```



All Rows Fetched: 5 in 0 seconds

	EMPID	EMPNAME	SALARY	POSITION	E_LEVEL
1	1	นางสาวกชพร อินต๊ะยอด	10000	Sale	(null)
2	2	นายกิตติกร ยาสมุทร	15972	Programmer	2
3	3	นางสาวจิราวัฒน์ สมเพชร์	10648	Sale	2
4	4	นายเจตน์ณเรงค์ ทะลา	22813	Manager	4
5	5	นายชลทัศน์ สหพงค์	17303	Programmer	3

```
UPDATE EMPLOYEE SET SALARY = 20000 WHERE EMPID =1;
```

	EMPID	EMPNAME	SALARY	POSITION	E_LEVEL
1	1	นางสาวกชพร อินต๊ะยอด	20000	Sale	(null)
2	2	นายกิตติกร ยาสมุทร	15972	Programmer	2
3	3	นางสาวจิราวัฒน์ สมเพชร์	10648	Sale	2
4	4	นายเจตน์ณเรงค์ ทะลา	22813	Manager	4
5	5	นายชลทัศน์ สหพงค์	17303	Programmer	3

WHEN IN DATABASE TRIGGER

ในคำสั่งการสร้าง Database Trigger ยังสามารถใช้คำสั่ง WHEN ช่วยในการทำงานของ Trigger ให้มีประสิทธิภาพสูงขึ้น ลดการเขียนโปรแกรมที่อาจจะมีความยาวลงได้ รูปแบบของการใช้คำสั่ง WHEN คือ WHEN [condition]

```
CREATE OR REPLACE TRIGGER WHEN_COND_TRG
BEFORE INSERT OR UPDATE OF SALARY
ON EMPLOYEES
FOR EACH ROW
WHEN (:NEW.SALARY < :OLD.SALARY)
BEGIN
    :NEW.SALARY := :OLD.SALARY;
END;
```

SYSTEM TRIGGERS

นอกจาก Database Trigger ที่สร้างขึ้นจะทำงานเนื่องมาจากคำสั่ง DML แล้วนั้น คำสั่งประเภท DDL และ เหตุการณ์ที่เกิดขึ้นจากการกระทำของตัวระบบฐานข้อมูลเอง (Database Events) ก็สามารถทำให้ Database Trigger ทำงานได้เช่นกัน ซึ่งเรียกว่า System Triggers

คำสั่งประเภท DDL ได้แก่ CREATE ALTER DROP หรือ GRANT เป็นต้น

Database Events เช่น LOGON, LOGOUT, STARTUP หรือ SHUTDOWN

```
CREATE TABLE LOG_AUDIT(
  USER_ID VARCHAR2(30),
  LOGINDATE DATE
);
CREATE OR REPLACE TRIGGER LOGON_TRG
AFTER LOGON --CREATE
ON SCHEMA
BEGIN
    INSERT INTO LOG_AUDIT VALUES(USER, SYSDATE);
END;
```

CONTROLLING TRIGGERS USING SQL

Disable หรือ Re-enable ส่วนของ database trigger

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

Disable หรือ Re-enable ส่วนของ triggers ทั้งหมดของตารางนั้น ๆ

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

การลบ trigger จากฐานข้อมูล

```
DROP TRIGGER trigger_name
```

VIEW DATABASE TRIGGERS

หากต้องการเรียกดู Database Trigger ที่สร้างขึ้นมาสามารถเรียกดูข้อมูลเหล่านี้ได้จากตาราง USER_TRIGGERS

```
SELECT TRIGGER_TYPE, TABLE_NAME, TRIGGERING_EVENT
FROM USER_TRIGGERS
WHERE TRIGGER_NAME = 'COND_TRG';
```

```
TRIGGER_TYPE     TABLE_NAME                     TRIGGERING_EVENT
---------------- ------------------------------ ------------------------------
BEFORE EACH ROW  EMPLOYEE                       INSERT OR UPDATE OR DELETE
```

```
SELECT TRIGGER_BODY
FROM USER_TRIGGERS
WHERE TRIGGER_NAME = 'COND_TRG';
```

```
TRIGGER_BODY
--------------------------------------------------------------------
BEGIN
  IF :NEW.SALARY < :OLD.SALARY THEN
       :NEW.SALARY := :OLD.SALARY;
  END IF;
END;
```