Trigger

Tổng quan về trigger

- Trigger là một đơn vị chương trình lưu trữ trong database và thực thi (fire) để đáp ứng một sự kiện nào đó.
- Sự kiện này được kết hợp với một table, view, schema, hoặc database, và là một trong những sự kiện sau:
  - Một câu lệnh DML (DELETE, INSERT, hoặc UPDATE).
  - Một câu lệnh DDL (CREATE, ALTER, DROP)
  - Một tác vụ trên database (SERVERERROR, LOGON, LOGOFF, STARTUP, hoặc SHUTDOWN).

Sự khác biệt giữa trigger và constraint (ràng buộc)

- Cả trigger và contraint đều ràng buộc dữ liệu đầu vào, nhưng chúng có những điểm khác biệt đáng kể:
  - Trigger chỉ áp dụng cho dữ liệu mới.
  - Constraint có thể áp dụng cho dữ liệu mới hoặc cả dữ liệu cũ và dữ liệu mới
  - Trigger có thể tuân theo những quy tắc phức tạp mà constraint không thể.

DML Trigger

- DML trigger được tạo trên table hoặc view và nó bắt các sự kiện INSERT, DELETE, UPDATE.
- DML trigger có thể là trigger đơn (simple) hoặc trigger phức hợp (compound).

Simple DML trigger

- Simple DML trigger kích hoạt duy nhất ở một trong những thời điểm sau:
  - Trước khi câu lệnh thực thi
    (BEFORE statement trigger hay statement-level BEFORE trigger)
  - Sau khi câu lệnh thực thi
    (AFTER statement trigger hay statement-level AFTER trigger)
  - Trước mỗi dòng (row) mà câu lệnh tác động
    (BEFORE each row trigger hay row-level BEFORE trigger).
  - Sau mỗi dòng (row) mà câu lệnh tác động
    (AFTER each row trigger hay row-level AFTER trigger).

Compound DML trigger

- Compound DML trigger có thể kích hoạt tại một thời điểm, một vài thời điểm hoặc tất cả các thời điểm.
- Compound trigger giúp ích cách tiếp cận trong việc chia sẻ dữ liệu ở các thời điểm khác nhau (timing point).

Thứ tự kích hoạt trigger (DML trigger)

- Thứ tự kích hoạt trigger:
  - Trước khi câu lệnh DML thực thi
  - Đối với mỗi dòng (row):
    - a. trước mỗi row mà câu lệnh tác động.
    - b. thực thi Insert, Update, Delete
    - c. sau mỗi row mà câu lệnh tác động
  - Sau khi câu lệnh DML thực thi.

Thứ tự kích hoạt trigger (ví dụ)

Giả sử ta có 4 SIMPLE DML trigger được tạo trên table NHANVIEN (với phần thực thi là null)

BEFORE statement trigger
AFTER statement trigger
BEFORE each row trigger
AFTER each row trigger

Thực thi câu lệnh:

UPDATE nhanvien SET luong = luong + 10 WHERE diachi = 'Q11';

NHANVIEN

MANV	HOTEN	DIACHI	LUONG
1	NGUYEN VAN A	Q11	20 30
2	NGUYEN VAN B	Q11	50 60
3	NGUYEN VAN C	Q10	30
4	NGUYEN VAN D	Q11	100 110

BEFORE statement trigger

← BEFORE each row
thực thi Update
AFTER each row trigger

← BEFORE each row
thực thi Update
AFTER each row trigger

← BEFORE each row
thực thi Update
AFTER each row trigger

AFTER statement trigger

Tạo trigger

```
CREATE [OR RELACE] TRIGGER triggername
        {BEFORE | AFTER}
        {DELETE, INSERT, UPDATE [OF columnname….]}
    ON tablename
        [REFERENCING {OLD AS old, NEW AS new}]
        [FOR EACH ROW [WHEN condition]]
        DECLARE
                Variable declaration;
                Constant declaration;
BEGIN
         PL/SQL subprogram body;
        [EXCEPTION
                exception PL/SQL block;
END;
```

Tạo trigger
Ví dụ

```
CREATE TRIGGER Print_trigger_type _biu_s
  BEFORE INSERT OR UPDATE ON emp
  BEGIN
    dbms_output.put_line('call before statement');
  END;
```

```
CREATE TRIGGER Print_trigger_type _aiu
 AFTER INSERT OR UPDATE ON emp
 FOR EACH ROW
 BEGIN
    dbms_output.put_line('call after each row');
 END;
```

```
CREATE TRIGGER Print_trigger_type _biu
BEFORE INSERT OR UPDATE ON emp
  FOR EACH ROW
  BEGIN
    dbms_output.put_line('call before each row');
  END;
```

```
CREATE TRIGGER Print_trigger_type _aiu_s
  AFTER INSERT OR UPDATE ON emp
BEGIN
   dbms_output.put_line('call after statement');
END;
```

Vị từ điều kiện trong trigger

Vị từ điều kiện	TRUE nếu và chỉ nếu:
INSERTING	Một câu lệnh INSERT kích hoạt trigger
UPDATING	Một câu lệnh UPDATE kích hoạt trigger
UPDATING ('column')	Một câu lệnh UPDATE tác động trên một column cụ thể nào đó kích hoạt trigger.
DELETING	Một câu lệnh DELETE kích hoạt trigger

Vị từ điều kiện trong trigger
Ví dụ

```
CREATE OR REPLACE TRIGGER     t
  BEFORE
    INSERT OR
    UPDATE OF salary, department_id OR
    DELETE
  ON employees
BEGIN
    CASE
          WHEN INSERTING THEN
                DBMS_OUTPUT.PUT_LINE('Inserting');
          WHEN UPDATING('salary') THEN
                DBMS_OUTPUT.PUT_LINE('Updating salary');
          WHEN UPDATING('department_id') THEN
                DBMS_OUTPUT.PUT_LINE('Updating department ID');
          WHEN DELETING THEN
                DBMS_OUTPUT.PUT_LINE('Deleting');
    END CASE;
END;
```

Old và New

- Khi **row-trigger** kích hoạt, có 2 dữ liệu ảo được tạo, gọi là new và old.
  - new *table_name*%ROWTYPE;
  - old *table_name*%ROWTYPE;
- Old và new có kiểu dữ liệu ROWTYPE từ table bị tác động. Sử dụng dấu chấm (.) để tham chiếu đến column từ old và new.

Triggering Statement	OLD.field Value	NEW.field Value
INSERT	NULL	Post-insert value
UPDATE	Pre-update	Post-update value
DELETE	Pre-delete value	NULL

Có thể gán lại giá trị cho NEW đối với BEFORE EACH ROW TRIGGER
Không thể gán giá trị cho NEW đối với AFTER EACH ROW TRIGGER
Không thể gán lại giá trị cho OLD

Old và New (ví dụ)

Giả sử ta có một trigger BEFORE each row được tạo trên table NHANVIEN (với phần thực thi là null)

Thực thi câu lệnh:

UPDATE nhanvien SET luong = luong + 10 WHERE diachi = 'Q11';

NHANVIEN

MANV	HOTEN	DIACHI	LUONG
1	NGUYEN VAN A	Q11	20
2	NGUYEN VAN B	Q8	50
3	NGUYEN VAN C	Q10	30
4	NGUYEN VAN D	Q11	100

BEFORE each row ⇕ :old.luong = 20, :new.luong = 30

30 ← thực thi Update set luong = :NEW.luong

BEFORE each row ⇕ :old.luong = 100, :new.luong = 110

110 ← thực thi Update set luong = :NEW.luong

Ví dụ

- Viết trigger mỗi khi insert, update, delete thì in ra thông tin lương của nhân viên bao gồm lương cũ, lương mới và hiệu số giữa lương cũ và lương mới

```
CREATE OR REPLACE TRIGGER Print_salary_changes
   BEFORE  INSERT  OR UPDATE OR DELETE ON emp
   FOR EACH ROW
DECLARE
        sal_diff          NUMBER;
BEGIN
       sal_diff  :=      NVL( :NEW.luong,0)       -       NVL(:OLD.luong,0);
       dbms_output.put('Old salary: ' ||       :OLD. luong );
       dbms_output.put(' New salary: ' ||         :NEW. luong );
       dbms_output.put_line(' Difference ' ||   sal_diff);
END;
```

Thực thi câu lệnh:

UPDATE nhanvien SET salary = luong + 10 WHERE diachi = ‘Q11’;

NHANVIEN

MANV	HOTEN	DIACHI	LUONG
1	A	Q11	20
2	B	Q8	50
3	C	Q11	75

BEFORE each row ⇕ :old.luong = 20, :new.luong = 30

```
sal_diff := :NEW.luong - :OLD.luong;
dbms_output.put('Old salary: ' || OLD. luong );
dbms_output.put(' New salary: ' || :NEW. luong );
dbms_output.put_line(' Difference ' || sal_diff);
```

30 ← thực thi Update set luong = :NEW.luong

BEFORE each row ⇕ :old.luong = 75, :new.luong = 85

```
sal_diff := :NEW.luong - :OLD.luong;
dbms_output.put('Old salary: ' || OLD. luong );
dbms_output.put(' New salary: ' || :NEW. luong );
dbms_output.put_line(' Difference ' || sal_diff);
```

85 ← thực thi Update set luong = :NEW.luong

Ví dụ

- Viết trigger mỗi khi insert, update, delete thì in ra thông tin lương của nhân viên bao gồm lương cũ, lương mới và hiệu số giữa lương cũ và lương mới

```
CREATE OR REPLACE TRIGGER Print_salary_changes
   AFTER   INSERT OR UPDATE ON Nhanvien
   FOR EACH ROW
DECLARE
        sal_diff          NUMBER;
BEGIN
       sal_diff  :=       :NEW.luong       -       :OLD.luong;
       dbms_output.put('Old salary: ' ||       :OLD. luong );
       dbms_output.put(' New salary: ' ||         :NEW. luong );
       dbms_output.put_line(' Difference ' ||   sal_diff);
END;
```

Thực thi câu lệnh:

UPDATE nhanvien SET luong = luong + 10 WHERE diachi = ‘Q11’;

NHANVIEN

MANV	HOTEN	DIACHI	LUONG
1	A	Q11	20
2	B	Q8	50
3	C	Q11	75

30 ← thực thi Update set luong := 30

:old.luong = 20

:new.luong = 30

AFTER each row

```
sal_diff :=    :NEW.luong    -    :OLD.luong;
dbms_output.put('Old salary: ' ||   OLD. luong );
dbms_output.put(' New salary: ' || :NEW. luong );
dbms_output.put_line(' Difference ' ||   sal_diff);
```

85 ← thực thi Update set luong = 85

:old.luong = 75

:new.luong = 85

AFTER each row

```
sal_diff :=    :NEW.luong    -    :OLD.luong;
dbms_output.put('Old salary: ' ||   OLD. luong );
dbms_output.put(' New salary: ' || :NEW. luong );
dbms_output.put_line(' Difference ' ||   sal_diff);
```

Old và New (ví dụ)

- Viết trigger đảm bảo rằng khi thêm mới một hóa đơn (hoadon) thì trị giá (trigia) của hóa đơn đó luôn bằng 0.

INSERT INTO hoadon VALUES ('HD03', '16/06/2000', 700)

hoadon	Sohd	Nghd	Trigia
	HD01	15/06/2000	30000
	HD02	15/06/2000	200
	HD03	**16/06/2000**	**0**

```
CREATE OR REPLACE TRIGGER insert_hoadon
   BEFORE  INSERT ON hoadon
   FOR EACH ROW
BEGIN
        :NEW.trigia := 0;
END;
```

Ví dụ

- Viết trigger mỗi khi thêm hoặc sửa dữ liệu thì HOTEN của nhân viên phải được chuyển thành chữ in hoa.

NHANVIEN	MANV	HOTEN	DIACHI	LUONG
	1	NGUYEN VAN A	Q11	20
	2	NGUYEN VAN B	Q8	50
	3	NGUYEN VAN C	Q10	30
	4	NGUYEN VAN D	Q11	100

```
CREATE TRIGGER hoten_upper
   BEFORE INSERT OR UPDATE ON nhanvien
        FOR EACH ROW
BEGIN
            :NEW.hoten := UPPER (:NEW.hoten);
END;
```

Ví dụ

- Viết trigger đảm bảo mỗi khi tăng lương của nhân viên nếu lương mới thấp hơn lương cũ thì lương vẫn được giữ nguyên bằng lương cũ.

```
UPDATE nhanvien SET luong =(luong+40)/2
```

NHANVIEN	MANV	HOTEN	DIACHI	LUONG	
	1	NGUYEN VAN A	Q11	20	30
	2	NGUYEN VAN B	Q8	50	50
	3	NGUYEN VAN C	Q10	30	35

```
CREATE OR REPLACE TRIGGER nhanvien_tangluong
   BEFORE  UPDATE OF luong ON nhanvien
        FOR EACH ROW
BEGIN
       IF (:NEW.luong < :OLD.luong) THEN
              :NEW.luong    :=    :OLD.luong;
       END IF;
END;
```

Ví dụ

- Mỗi khi có một user nào đó tác động vào bảng nhanvien thì thông tin về username, ngày cập nhật được lưu trữ trong table emp_log.

```
CREATE TABLE Emp_log (
        Username VARCHAR2(50),
        Log_date DATE
);
```

```
CREATE OR REPLACE TRIGGER Log_emp_update
  AFTER INSERT OR UPDATE OR DELETE ON nhanvien
BEGIN
            INSERT INTO Emp_log   VALUES (USER, SYSDATE);
END;
```

Aborting Triggers with Error
Ví dụ

- Viết trigger đảm bảo mỗi khi tăng lương của nhân viên thì lương mới không được thấp hơn lương cũ.Nếu tồn tại một trường hợp nào đó mà lương mới thấp hơn lương cũ thì báo lỗi.

```
CREATE OR REPLACE TRIGGER nhanvien_tangluong
   BEFORE  UPDATE OF luong ON nhanvien
        FOR EACH ROW
BEGIN
      IF (:NEW.luong < :OLD.luong) THEN
            RAISE_APPLICATION_ERROR(-20000, 'luong moi khong duoc thap
                                                              hon luong cu');
      END IF;
END;
```

Aborting Triggers with Error
Ví dụ

- Viết trigger đảm bảo tuổi vào làm của nhân viên không được nhỏ hơn 18

```
CREATE OR REPLACE TRIGGER PersonCheckAge
AFTER INSERT OR UPDATE OF Ngaysinh, ngayvl  ON NHANVIEN
FOR EACH ROW
BEGIN
   IF (extract(year from :NEW.ngayvl) - extract(year from :NEW.ngaysinh)<18)
THEN
      RAISE_APPLICATION_ERROR(-20000, ‘Tuoi khong nho hon 18');
   END IF;
END;
```

Ví dụ: trị giá hóa đơn bằng tổng tien (tiền) của các CTHD thuộc hóa đơn đó

SOHD	NGHD	TRIGIA
HD01	25/12/2008	30000
HD02	12/05/2009	23000
HD03	12/05/2009	0

SOHD	MASP	TIEN
HD01	BB01	15000
HD01	BB02	10000
HD01	BB03	5000
HD02	BB01	20000
HD02	BB04	3000

```
CREATE TRIGGER cthd_ins_ai
AFTER INSERT  ON   CTHD
FOR EACH ROW
BEGIN
      UPDATE  hoadon SET   trigia = NVL(trigia, 0)+ :NEW.tien
      WHERE  sohd = :NEW.sohd;
END;
```

```
CREATE TRIGGER cthd_del_ad
AFTER DELETE  ON   CTHD
FOR EACH ROW
BEGIN
      UPDATE  hoadon SET   trigia = NVL(trigia, 0) -  :OLD.tien
      WHERE  sohd = :OLD.sohd;
END;
```

```
CREATE TRIGGER cthd_upd_au
AFTER UPDATE OF sohd, tien  ON   CTHD
FOR EACH ROW
BEGIN
      UPDATE  hoadon SET   trigia = NVL(trigia, 0) + :NEW.tien
      WHERE  sohd = :NEW.sohd;
      UPDATE  hoadon SET   trigia = NVL(trigia, 0) -  :OLD.tien
      WHERE  sohd = :OLD.sohd;
END;
```

Cách khác

```
CREATE TRIGGER cthd_aiud_fer
AFTER INSERT OR DELETE OR UPDATE OF sohd, tien ON cthd
FOR EACH ROW
BEGIN
  IF ( INSERTING OR UPDATING )  THEN
    UPDATE hoadon  SET trigia= NVL (trigia,0)+:NEW.tien
    WHERE sohd = :NEW.sohd;
  END IF;

  IF ( UPDATING OR DELETING)   THEN
    UPDATE hoadon
    SET trigia = NVL(trigia,0)-:OLD.tien
    WHERE sohd = :OLD.sohd;
  END IF;
END;
```

Ví dụ: trị giá hóa đơn bằng tổng sl*gia của các CTHD thuộc hóa đơn đó

SOHD	NGHD	TRIGIA
HD01	25/12/2008	30000
HD02	12/05/2009	23000
HD03	12/05/2009	0

SOHD	MASP	SL	GIA
HD01	BB01	15	1000
HD01	BB02	10	1000
HD01	BB03	1	5000
HD02	BB01	20	1000
HD02	BB04	2	1500

```
CREATE TRIGGER cthd_ins_ai
AFTER INSERT  ON   CTHD
FOR EACH ROW
BEGIN
      UPDATE  hoadon SET   trigia = trigia + :NEW.sl*  :NEW.gia
      WHERE  sohd = :NEW.sohd;
END;
```

```
CREATE TRIGGER cthd_del_ad
AFTER DELETE  ON   CTHD
FOR EACH ROW
BEGIN
      UPDATE  hoadon SET   trigia = trigia -  :OLD.sl*  :OLD.gia
      WHERE  sohd = :OLD.sohd;
END;
```

```
CREATE TRIGGER cthd_upd_au
AFTER UPDATE  ON   CTHD
FOR EACH ROW
BEGIN
      UPDATE  hoadon SET   trigia = trigia + :NEW.sl*  :NEW.gia
      WHERE  sohd = :NEW.sohd;
      UPDATE  hoadon SET   trigia = trigia -  :OLD.sl*  :OLD.gia
      WHERE  sohd = :OLD.sohd;
END;
```

Table mutating

- Error xảy ra khi một row-level trigger truy cập đến cùng table mà trigger đang dựa lên trong lúc thực thi. Table lúc này gọi là mutating.

OPERATION	TYPE	MUTATING?
insert	before/statement-level	No
insert	after/statement-level	No
update	before/statement-level	No
update	after/statement-level	No
delete	before/statement-level	No
delete	after/statement-level	No
insert	before/row-level	Single row NO,Multi-row yes
insert	after/row-level	Yes
update	before/row-level	Yes
update	after/row-level	Yes
delete	before/row-level	Yes
delete	after/row-level	Yes

Table mutating

employee	id	firstname	salary	deptid
	1	helen	100	fin
	2	john	50	fin

```
CREATE TRIGGER employee_raisesal
BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW
        WHEN (NEW.id != 1)
DECLARE
                sal1 NUMBER;
BEGIN
        SELECT salary INTO sal1 FROM  employee;
        -- more processing...
END;
```

```
update employee set salary=50 where id=2;
```

MUTATING ERROR