트리거

- 이벤트처리 (~했을때 수반되는 처리)
- 특정테이블에 이벤트 (insert, delete, update)가 발생했을 시
  다른 테이블에 연관된 내용을 변경하도록 하는일.

형식)

```
CREATE [OR REPLACE] TRIGGER 트리거명
(BEFORE|AFTER) (INSERT|DELETE|UPDATE)  -- 이벤트
ON 테이블명  -- 이벤트가 발생하는 테이블
[FOR EACH ROW] -- 실행될 문장 행에 각각 적용.
[WHEN 조건식]
BEGIN
   이벤트 발생시 실행할 문장(주로 DML 문장); -- 이벤트 핸들러
END;
```

문제) 사원테이블에 사원정보가 새로 입력될 때 마다
입사환영메시지를 출력하시오.

```
?

1  DROPTABLEEMP02;
2
3  CREATETABLEEMP02
4  AS(
5      SELECTEMPNO
6           , ENAME
7           , DEPTNO
8        FROMEMP
9       WHERE1=0
10);
11
12CREATEORREPLACETRIGGERTR_WELCOME
```

```
13AFTERINSERTONEMP02
14BEGIN
15     DBMS_OUTPUT.PUT_LINE('WELCOME KOSTA');
16END;
17/
18
19INSERTINTOEMP02 VALUES(7002, '길라임',20);
20WELCOME KOSTA
21
221 개의 행이 만들어졌습니다.
23INSERTINTOEMP02 VALUES(7004, '김주원', 40);
24WELCOME KOSTA
25
261 개의 행이 만들어졌습니다.
```

문제) 사원테이블에 사원정보가(EMPNO, ENAME, SAL) 입력되었을때
급여테이블(NO, EMPNO, SAL)에 그 사원에 대한 급여 정보를 자동으로
입력하도록 트리거를 설계하시오.

```
?

1  CREATETABLEEMP03
2  (
3          EMPNO NUMBER(4) PRIMARYKEY
4        , ENAME VARCHAR2(15)
5        , SAL NUMBER(7,2)
6  );
7
8  CREATETABLESALARY
9  (
10         NONUMBER PRIMARYKEY
11       , EMPNO NUMBER(4)
12       , SAL NUMBER(7,2)
13);
```

```
14
15CREATESEQUENCESALARY_SEQ
16        START WITH1
17        INCREMENT BY1
18        NOCYCLE
19        NOCACHE;
20
21---> INSERT INTO EMP03 VALUES (8000, '홍길동', 3000);
22---> INSERT INTO SALARY VALUES (1, 8000, 3000);
23
24INSERTINTOEMP03 VALUES(8000, '홍길동', 3000);
25
26CREATEORREPLACETRIGGERTR_SALARY
27AFTERINSERTONEMP03
28FOREACH ROW
29BEGIN
30    INSERTINTOSALARY
31VALUES(SALARY_SEQ.NEXTVAL, :NEW.EMPNO, :NEW.SAL);
32END;
33/
34
35SELECT* FROMEMP03;
36
37      EMPNO ENAME                    SAL
38---------- -------------- ----------
39       8000 홍길동                  3000
40
41SELECT* FROMSALARY;
42
43         NO       EMPNO         SAL
44---------- ---------- ----------
             1         8000         3000
```

<바인드변수> 2 가지

- FOR EACH ROW 를 사용해야 함

:NEW - 새로 입력된(INSERT, UPDATE) 데이터

:OLD - 기존 데이터

사용법) :NEW.EMPNO

문제) 테이블 만들기

1. <상품테이블>
2. <입고테이블>
3. 제품이 입고되면 상품재고에 자동으로 입력되게 함.
4. 입력, 수정, 삭제되면 재고에 변화를 줘야함.

1. 입력트리거 (입고테이블에 상품이 입력되었을때 재고수량 증가)

예) 입고테이블에 모니터가 10 개 입고 되었을때 자동으로 상품의 재고가 변경.

2. 수정트리거 (입고테이블에 상품의 재고정보가 변경되었을때 재고수량 변경)

예) 두번째 입고된 15 를 10 으로 변경하면 상품테이블의 재고수량 25 가 20 으로 변경

3. 삭제트리거(입고테이블에 입력된 행이 삭제되었을때 그 수만큼 재고수량 감소)

예) 첫번째 입고된 행(수량 10 입력)이 삭제되면 삼풍테이블의 재고수량이 20 에서 10 으로 변경

```
?

1  CREATETABLE 상품
2  (
3          상품코드  CHAR(4) CONSTRAINT 상품_PK PRIMARYKEY--a001
4        , 상품명  VARCHAR(15) NOTNULL
```

```
5       , 제조사 VARCHAR(15)
6       , 소비자가격 NUMBER
7       , 재고수량 NUMBER DEFAULT0
8  );
9
10--상품등록
11INSERTINTO 상품(상품코드, 상품명, 제조사, 소비자가격)
12VALUES('a001', '마우스', 'LC', 1000);
13
14INSERTINTO 상품 VALUES('a002', '키보드', '삼송', 2000, 0);
15
16INSERTINTO 상품 VALUES('a003', '모니터', 'HB', 10000, 0);
```

?

```
1  CREATETABLE 입고
2  (
3         입고번호 NUMBER PRIMARYKEY
4       , 상품코드 CHAR(4) REFERENCES 상품(상품코드)
5       , 입고일자 DATE
6       , 입고수량 NUMBER
7       , 입고단가 NUMBER
8       , 입고금액 NUMBER
9  );
10
11CREATESEQUENCE 입고_SEQ
12        START WITH1
13        INCREMENT BY1
14        NOCYCLE
15        NOCACHE;
16
17CREATEORREPLACEPROCEDURESP_PRO_INSERT(
18        CODE CHAR
19      , SU NUMBER
```

```
20      , WON NUMBER
21)
22IS
23BEGIN
24    INSERTINTO 입고
25    VALUES(
26             입고_SEQ.NEXTVAL
27           , CODE
28           , SYSDATE
29           , SU
30           , WON
31           , SU*WON
32     );
33END;
34/
```

```
?

1  -- TRIGGER INSERT
2  CREATEORREPLACETRIGGERTR_PRODUCT_INSERT
3  AFTERINSERTON 입고
4  FOREACH ROW
5  BEGIN
6       UPDATE 상품  SET
7              재고수량  =  재고수량  + :NEW.입고수량
8        WHERE 상품코드  = :NEW.상품코드;
9  END;
10/
11/*
12UPDATE 입고  SET
13         입고금액  =  입고수량  *  입고단가;
```

```
14ORA-04091: SCOTT.입고 테이블이 변경되어 트리거/함수가 볼 수 없습니다.
15ORA-06512: "SCOTT.TR_PRODUCT_INSERT", 6 행
16ORA-04088: 트리거 'SCOTT.TR_PRODUCT_INSERT'의 수행시 오류
17*/
18----> 프로시저로 INSERT 함으로 해결
19
20-- TRIGGER UPDATE
21CREATEORREPLACETRIGGERTR_PRODUCT_UPDATE
22AFTERUPDATEON 입고
23FOREACH ROW
24BEGIN
25    UPDATE 상품 SET
26           재고수량 = 재고수량 - :OLD.입고수량 + :NEW.입고수량
27     WHERE 상품코드 = :OLD.상품코드;
28END;
29/
30-- TRIGGER DELETE
31CREATEORREPLACETRIGGERTR_PRODUCT_DELETE
32AFTERDELETEON 입고
33FOREACH ROW
34BEGIN
35    UPDATE 상품 SET
36           재고수량 = 재고수량 - :OLD.입고수량
37     WHERE 상품코드 = :NEW.상품코드;
38END;
39/
40
41EXECSP_PRO_INSERT('a002', 20, 3000);
42
43EXECSP_PRO_INSERT('a002', 10, 3000);
44
45UPDATE 입고 SET 입고수량 = 15 WHERE 입고번호 = 7;
```

```
46
47DELETEFROM 입고  WHERE 입고번호  = 6;
48
49SELECT* FROM 상품;
50
51SELECT* FROM 입고;
```

```
?

1  상품  상품명          제조사          소비자가격    재고수량
2  ---- --------------- --------------- ---------- ----------
3  a001  마우스          LC                 1000          0
4  a002  키보드          삼송               2000         30
5  a003  모니터          HB                10000         30
6
7
8    입고번호  상품  입고일자    입고수량    입고단가    입고금액
9  ---------- ---- -------- ---------- ---------- ----------
10         13 a002 15/05/15         20       3000       60000
11          7 a003 15/05/15         15       5000       50000
12         14 a002 15/05/15         10       3000       30000
13          8 a003 15/05/15         15       5000       75000
```

```
1  create or replace function fn_salary_ename(
2  v_ename in employee.ename%type)
3  return number
4  is
```

```
  5   v_salary number(7,2);
  6   begin
  7   select salary into v_Salary
  8   from employee
  9   where ename = v_ename;
 10   return v_salary;
 11* end;
```

함수가 생성되었습니다.

```
SQL> variable v_salary number;
SQL> execute :v_salary := fn_salary_ename('SCOTT');
```

PL/SQL 처리가 정상적으로 완료되었습니다.

```
SQL> print v_salary;

  V_SALARY
----------
        3000
```

[출처]orcle pl/sql 함수(function)|**작성자** seo2918

DML trigger

after trigger

table 에 insert, update, delete 등의 작업이 일어난 후에 작동

before trigger

table 이나 view 에 이벤트가 작동하기 전에 작동한다.

table, view 둥다 작동하는데 주로 뷰가 업데이트 가능하도록 사용한다

insert, update, delete 세가지 이벤트로 작동

트리거는 몸체의 내용이 몇번이나 실행되는지에 따라서 문장레벨 트리거, 행레벨 트리거로 나뉜다

문장레벨 트리거

어떤 table 에 대해 DML 문을 실행할 때 단 한번만 트리거를 발생시키는 것

행 레벨 트리거

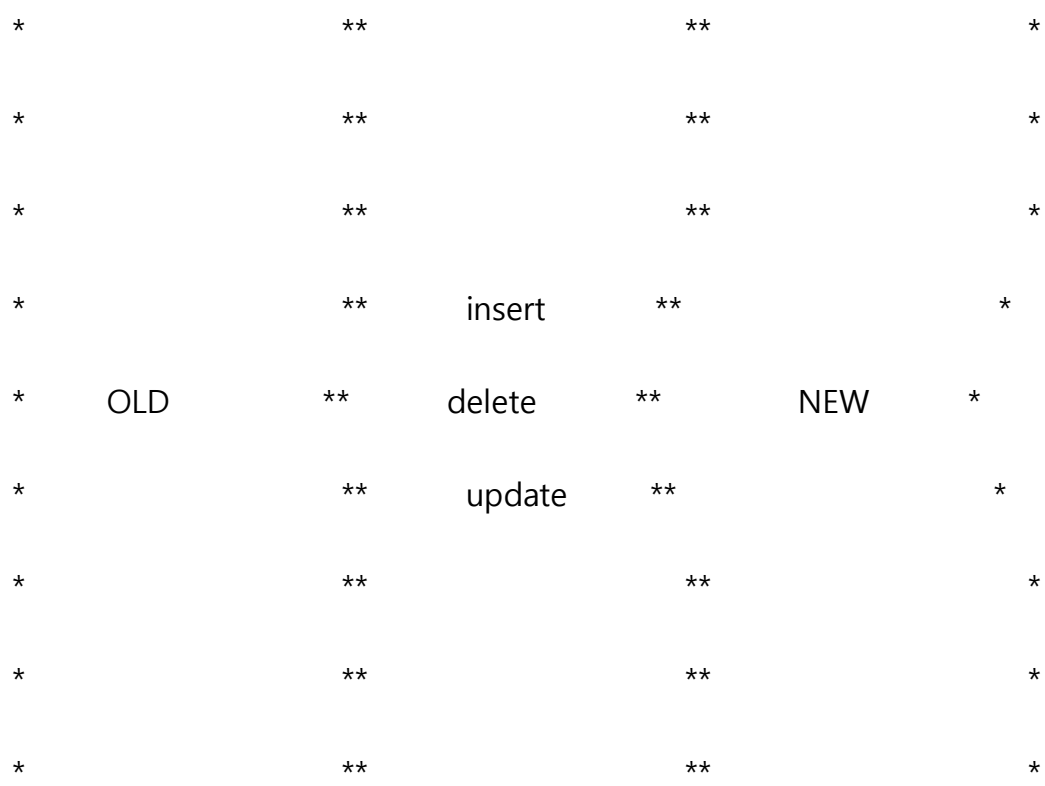
어떤 DML 에 대해 table 의 행이 변경된 수에 따라 트리거가 발생되는 것

즉, 10 개의 행이 바뀐다면 트리거는 10 번 실행된다.

for each row 를 사용해서 문장레벨 트리거와 행 레벨 트리거를 나눈다.

행 레벨 트리거는 실제 데이터의 값을 취하기 위해서 OLD.salary, NEW.salary 와 같은 키워드를 사용한다.

이렇게 생각하면 된다



OLD 와 NEW 는 둘다 임시 테이블이다. 그러니까 insert, delete, update 문이 실행되면 oracle 에서는 DML 이 실행되면 최신화 된 테이블은 NEW 라는 테이블에 저장되고, DML 이 실행되기 이전의 테이블은 잠시동안 OLD 라는 테이블에 임시로 저장되어진다.

////////////////////////////insert trigger////////////////////////////

```
create or replace trigger trigger_sampel1
        after insert
```

```
        on dept_original
        for each row
begin
        if inserting then
        dbms_output.put_line('insert trigger 발생');
        insert into dept_copy
        values(:old.dno, :old.dname, :old.loc);
        end if;
end;
////////////////////////////insert trigger////////////////////////////


////////////////////////////delete trigger////////////////////////////
create or replace trigger trigger_sampel1
        delete insert
        on dept_original
        for each row
begin
        if inserting then
        dbms_output.put_line('delete trigger 발생');
        delete from dept_copy
```

```
        where dept_copy.dno = :old.dno;

        end if;

end;

////////////////////////////delete trigger////////////////////////////
```