/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

트리거(Trigger)

- 데이터베스의 오브젝트에 이벤트가 발생하면 자동으로 실행되는 프로시저

- 이벤트: 트리거를 작동는 상황.

- DML 문 : insert, update, delete

- DDL 문 : create, alter, drop

- 데이터베이스 자체 이벤트 : 서버에러, 로그인, 데이터베이스 시작, 종료

- 트리거 타이밍: 트리거 실행 시점.

- BEFORE : 이벤트 발생 전

- AFTER : 이벤트 발생 후

- 트리거 종류

- 행 트리거

- DML의 영향을 받는 모든 행단위로 트리거 실행.

- 변경 레코드를 제어할 수 있다.

- FOR EACH ROW 옵션

- 명령문 트리거

- DML 문 단위로 트리거 실행

- 한 구문에서 몇개의 행이 갱신되어도 트리거는 한번만 호출 된다.

- 구문

CREATE [ OR REPLACE] TRIGGER 트리거이름

타이밍

이벤트 [ OR 이벤트2 OR 이벤트3 ..]

ON 테이블이름

[FOR EACH ROW]

[DECLARE

선언부]

BEGIN

실행부]

[EXCEPTION

예최처리]

END;

- 타이밍 : BEFORE, AFTER

- BEFORE에서 예외가 발생하면 event 쿼리도 실행 되지 않는다. after일 경우는 예외가 발생해도 event 쿼리는 실행된다.

- 이벤트

- 트리거를 발생시킬 SQL 구문

- update 의 경우 : update of 컬럼명[,컬럼명] 으로 컬럼을 지정할 수 있다.

- 테이블 : 트리거와 연결될 테이블 이름

- FOR EACH ROW : 넣으면 행 트리거, 생략하면 명령문 트리거

- DECLARE 절 : 실행문에서 사용할 변수를 선언 한다. 생략 가능

- BEGIN 절 : 실행부

- 실행부에서 commit/rollback을 할 수 없다.

- EXCEPTION : 실행부에서 발생한 예외처리. 생략 가능

- 실행부에서 이벤트 적용된 값 제어

- :old : 참조 전의 컬럼의 값 (update : 수정전 데이터, delete: 삭제 전의 데이터)

- :new : 참조 후의 컬럼의 값 (insert : 입력된 데이터, update: 수정한 데이터)

- :new.컬럼명, :old.컬럼명 으로 조회한다.

트리거 제거

- drop trigger 트리거 이름

- 연결된 테이블이 삭제되면 트리거도 자동 삭제된다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

set serveroutput on;

create or replace trigger ex\_trigger --0~n개의 이벤트가 발생할 때 실행됨

after delete on emp

--declare --is와 달리 생략가능

begin

dbms\_output.put\_line('emp에서 삭제가 발생');

end;

/

delete from emp;

rollback;

delete from emp where 1=0; --실제 0행이 삭제되지만 트리거는 동작함.

-- 행 트리거

DROP TABLE DEPT\_COPY;

CREATE TABLE DEPT\_COPY AS SELECT \* FROM DEPT;

create or replace trigger ex\_2\_trigger

before insert or delete or update of loc -- loc가 삽입, 삭제, 업데이트 될 때만 트리거 발생

on dept\_copy

for each row --행트리거

begin

dbms\_output.put\_line('변경전 loc : '|| :old.loc);

dbms\_output.put\_line('변경후 loc : '|| :new.loc);

dbms\_output.put\_line('변경전 dept\_name : '|| :old.dept\_name);

dbms\_output.put\_line('변경후 dept\_name : '|| :new.dept\_name);

dbms\_output.put\_line('------------------------------------');

end;

/

insert into dept\_copy values (3030,'새부서','인천');

delete from dept\_copy where loc = '인천';

update dept\_copy

set dept\_name=dept\_name, loc = '부산'

where loc = 'Seattle';

-- DEPT 테이블에 행이 insert 되면 같은 값을 DEPT\_COPY에 insert하는 트리거 작성.

create or replace trigger dept\_copy\_trigger

after insert on dept

for each row

declare

ex\_error exception;

begin

raise ex\_error; --중간에 예외가 발생할 경우 트리거가 정상적으로 처리되지 않는다.

insert into dept\_copy values(:new.dept\_id, :new.dept\_name, :new.loc);

--commit; 트리거 안에서는 트랜잭션 관련 작업 불가(commit, rollback)

end;

/

insert into dept values(1900, '영업기획부', '역삼동');

insert into dept values(1901, '영업기획부', '역삼동');

insert into dept values(1902, '영업기획부', '역삼동');

rollback;

select \* from dept where loc = '역삼동';

select \* from dept\_copy where loc = '역삼동';

-- 명령문 트리거

-- dept\_copy에 오전 12 ~ 07시 까지는 DML 구문을 실해하지 못하도록 하는 trigger 구현

create or replace trigger ban\_dml\_dept\_copy\_trigger

before insert or update or delete on dept\_copy

declare

ex\_error exception;

v\_hour varchar2(2);

begin

--시간 조회

v\_hour := to\_char(sysdate,'hh24');

if v\_hour between '00' and '07' then

raise ex\_error;

end if;

end;

/

update dept\_copy set loc = loc;

-- emp\_copy 테이블에서 DML(insert, delete, update)이 발생하면 그것을 log로 남기는 트리거를 생성

drop table emp\_copy ;

create table emp\_copy as select \* from emp where 1 =0;

drop table logs;

create table logs (

table\_name varchar2(100), -- 테이블이름

event\_time date default sysdate, --DML 실행시간

event varchar2(10) --DML 종류

);

create or replace trigger emp\_log\_trigger

after insert or delete or update

on emp\_copy

declare

v\_event varchar2(10); --어떤 DML이 실행됐는지 저장.

begin

if inserting then --inserting : event가 insert면 true, 아니면 false

v\_event := 'INSERT';

elsif updating then

v\_event := 'UPDATE';

elsif deleting then

v\_event := 'DELETE';

end if;

insert into logs values ('emp\_copy', sysdate, v\_event);

end;

/

insert into emp\_copy (emp\_name, hire\_date, salary) values ('홍길동', '2000-05-01', 10000);

update emp\_copy set emp\_name = emp\_name;

delete from emp\_copy;

desc emp\_copy;

select table\_name, to\_char(event\_time, 'hh24:mi:ss') 실행시간, event from logs;

-- 위의 코드를 dept\_copy\_log\_trigger, job\_copy\_log\_trigger 를 만들어 dept\_copy, job\_copy 테이블에 에 적용.

create or replace procedure logging\_sp(p\_table\_name varchar2)

is

v\_event varchar2(10); --어떤 DML이 실행됐는지 저장.

begin

if inserting then --inserting : event가 insert면 true, 아니면 false

v\_event := 'INSERT';

elsif updating then

v\_event := 'UPDATE';

elsif deleting then

v\_event := 'DELETE';

end if;

insert into logs values (p\_table\_name, sysdate, v\_event);

end;

/

create or replace trigger dept\_logging\_trigger

after insert or delete or update on dept

begin

logging\_sp('dept');

end;

/

create or replace trigger job\_logging\_trigger

after insert or delete or update on job

begin

logging\_sp('job');

end;

/

update job set job\_id = job\_id;

update dept set dept\_name = dept\_name;

select table\_name, to\_char(event\_time, 'hh24:mi:ss') 실행시간, event from logs;