20.패키지&트리거

목차

1. 패키지(1)

- 패키지의 사전적인 의미는 꾸러미이다.

- 관련 있는 프로시저를 보다 효율적으로 관리하기 위해서 패키지 단위로 배포할 때 유용하게 사용된다.

- 패키지는 패키지 선언(명세부)과 패키지 몸체 선언(몸체부) 두 가지 모두를 정의해야한다.

1. 패키지(2)

형식

```
CREATE [ OR REPLACE ] PACKAGE package_name
IS
PROCEDURE procedure_name1;
PROCEDURE procedure_name2;
END;
/


CREATE [ OR REPLACE ] PACKAGE BODY package_name
IS
PROCEDURE procedure_name1
IS
....
END;
END;
/
```

1.1 PL/SQL 구조

- 몸체부내에는 여러 가지의 프로시저나 함수를 정의하고 있다.
- 몸체부에 정의한 프로시저나 함수는 앞장에서 배운 저장 프로시저와 저장 함수와 동일한 문법구조를 갖는다.
- 명세부에는 몸체부에 정의한 함수들을 선언해 놓는다.
- 패키지 내의 정의된 프로시저나 함수를 호출하는 방식은 다음과 같다.

형식	**EXECUTE [패키지명].[프로시저명]**

1.2 패키지 작성하기(1)

- ex) 앞장에서 작성했던 저장 프로시저와 저장 함수로 구성된 패키지를 생성해보자
- 1. 다음과 같이 입력하시오.

예

```
CREATE OR REPLACE PACKAGE EXAM_PACK IS
FUNCTION CAL_BONUS(VEMPNO IN EMP.EMPNO%TYPE)
RETURN NUMBER;
PROCEDURE CURSOR_SAMPLE02;
END EXAM_PACK;
/
```

1.2 패키지 작성하기(2)

예

```
CREATE OR REPLACE PACKAGE BODY EXAM_PACK IS
    FUNCTION CAL_BONUS(VEMPNO IN EMP.EMPNO%TYPE )
        RETURN NUMBER
        IS
        VSAL NUMBER(7, 2);
        BEGIN
        SELECT SAL INTO VSAL
        FROM EMP
        WHERE EMPNO = VEMPNO;
        RETURN (VSAL * 200);
    END CAL_BONUS;
```

1.2 패키지 작성하기(3)

예

```
PROCEDURE CURSOR_SAMPLE02
        IS
        VDEPT DEPT%ROWTYPE;
        CURSOR C1
        IS
        SELECT * FROM DEPT;
        BEGIN
        DBMS_OUTPUT.PUT_LINE('부서번호 / 부서명 / 지역명');
        DBMS_OUTPUT.PUT_LINE('--------------------------');
        FOR VDEPT IN C1 LOOP
            EXIT WHEN C1%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(VDEPT.DEPTNO||
            ' '||VDEPT.DNAME||' '||VDEPT.LOC);
        END LOOP;
    END CURSOR_SAMPLE02;
END EXAM_PACK;
```

1.2 패키지 작성하기(4)

❖ 2. 실행하여 결과를 확인한다.

예

```
SET SERVEROUTPUT ON;

--함수 확인
VAR VAR_RES NUMBER;
EXEC :VAR_RES := EXAM_PACK.CAL_BONUS(1001);
PRINT VAR_RES;

--프로시저 확인
EXEC EXAM_PACK.CURSOR_SAMPLE02;
```

```
  VAR_RES
----------
     60000
```

```
부서번호 / 부서명 / 지역명
----------------------------
10 경리부 서울
20 인사부 인천
30 영업부 용인
40 전산부 수원
```

2. DBMS_OUTPUT 패키지(1)

- PL/SQL에서 조회한 결과값을 출력하기 위해서 DBMS_OUTPUT에 대한 자세한 설명 없이 DBMS_OUTPUT를 사용해 왔다. 패키지에 대한 개념을 학습하였기 때문에 이제 DBMS_OUTPUT에 대해서 살펴보도록 하자.

- 오라클을 설치하게 되면 특정 목적을 위해서 사용할 수 있도록 오라클 사에서 제공해 주는 패키지들이 설치된다. 이러한 패키지에는 오라클에서 제공되는 프로시저와 함수들을 관련된 것끼리 묶어서 집합으로 제공한다. 패키지에 대한 정보를 출력해 보도록 하자.

예

```
CONN system/password;
DESC DBA_OBJECTS;
```

2. DBMS_OUTPUT 패키지(2)

- 패키지들은 목적에 따라 관련된 프로시저와 함수들을 관리한다.

예

```
SELECT OBJECT_NAME FROM DBA_OBJECTS
WHERE OBJECT_TYPE='PACKAGE'
AND OBJECT_NAME LIKE 'DBMS_%'
ORDER BY OBJECT_NAME;
```

3. 트리거(1)

- 다음은 트리거(trigger)의 사전적인 의미이다.

① (총의) 방아쇠; =HAIR TRIGGER.
② 제동기, 제륜(制輪) 장치.
③ (연쇄 반응. 생리 현상. 일련의 사건 등을 유발하는) 계기,유인,자극.

- 오라클에서의 트리거 역시 해당 단어의 의미처럼 어떤 이벤트가 발생하면 자동적으로 방아쇠가 당겨져 총알이 발사되듯이 특정 테이블이 변경되면 이를 이벤트로 다른 테이블이 자동으로(연쇄적으로) 변경되도록 하기 위해서 사용한다.

- 트리거는 특정 동작을 이벤트로 그로인해서만 실행되는 프로시저의 일종이다.

3. 트리거(2)

❖ 트리거를 만들기 위한 CREATE TRIGGER 문의 형식은 다음과 같다.

형식

```
CREATE TRIGGER trigger_name
timing[BEFORE|AFTER] event[INSERT|UPDATE|DELETE]
ON table_name
[FOR EACH ROW]
[WHEN conditions]

BEGIN
statement
END
```

3. 트리거(3)

- 트리거의 타이밍
  - [BEFORE] 타이밍은 어떤 테이블에 INSERT, UPDATE, DELETE 문이 실행될 때 해당 문장이 실행되기 전에 트리거가 가지고 있는 BEGIN ~ END 사이의 문장을 실행한다.
  - [AFTER] 타이밍은 INSERT, UPDATE, DELETE 문이 실행되고 난 후에 트리거가 가지고 있는 BEGIN ~ END 사이의 문장을 실행한다.
- 트리거의 이벤트
  - 사용자가 어떤 DML(INSERT, UPDATE, DELETE)문을 실행했을 때 트리거를 발생시킬 것인지를 결정한다.
- 트리거의 몸체
  - 해당 타이밍에 해당 이벤트가 발생하게 되면 실행될 기본 로직이 포함되는 부분으로 BEGIN ~ END에 기술한다.

3.1 트리거의 유형

- 트리거의 유형은 FOR EACH ROW에 의해 문장 레벨 트리거와 행 레벨 트리거로 나눈다

- FOR EACH ROW가 생략되면 문장 레벨 트리거이고 행 레벨 트리거를 정의하고자 할 때에는 반드시 FOR EACH ROW를 기술해야만 한다.

- 문장 레벨 트리거는 어떤 사용자가 트리거가 설정되어 있는 테이블에 대해 DML(INSERT, UPDATE, DELETE)문을 실행할 때 단 한번만 트리거를 발생시킬 때 사용한다.

- 행 레벨 트리거는 DML(INSERT, UPDATE, DELETE)문에 의해서 여러 개의 행이 변경된다면 각 행이 변경될 때마다 트리거를 발생시키는 방법이다. 만약 5개의 행이 변경되면 5번 트리거가 발생된다.

3.2 트리거의 조건

- 트리거 조건은 행 레벨 트리거에서만 설정할 수 있으며 트리거 이벤트에 정의된 테이블에 이벤트가 발생할 때 보다 구체적인 데이터 검색 조건을 부여할 때 사용된다.

3.3 단순 메시지 트리거 작성하기(1)

- ex) 사원 테이블에 새로운 데이터가 들어오면 '신입사원이 입사했습니다.' 란 메시지를 출력되도록 문장 레벨 트리거로 작성해보자.
- 1. 신입 사원의 정보를 추가할 사원 테이블을 새롭게 만들어 놓는다.
- 2. 다음과 같이 입력하시오.

예

```
CREATE OR REPLACE TRIGGER TRG_01
AFTER INSERT
ON EMP01
BEGIN
DBMS_OUTPUT.PUT_LINE('신입사원이 입사했습니다.');
END;
/
```

- 3. 트리거 확인

예

```
SELECT TRIGGER_NAME FROM USER_TRIGGERS;
```

3.3 단순 메시지 트리거 작성하기(2)

- 3. 트리거 확인

  예

```
SELECT TRIGGER_NAME FROM USER_TRIGGER;
```

- 4. 사원 테이블에 행을 추가해보자.

  예

```
SET SERVEROUTPUT ON
INSERT INTO EMP01 VALUES(1, '김홍도', '화가');
```

- ' 김홍도' 사원이 추가되자 '신입사원이 입사했습니다.' 란 메시지가 출력되는 것을 보면 TRG_01 트리거가 수행되었음을 확인할 수 있다.

3.4 급여 메시지 트리거 작성하기(1)

- ex) 사원 테이블에 새로운 데이터가 들어오면(즉, 신입 사원이 들어오면) 급여 테이블에 새로운 데이터(즉 신입 사원의 급여 정보)를 자동으로 생성하도록 하기 위해서 사원 테이블에 트리거를 작성해 봅시다. (신입사원의 급여는 일괄적으로 100으로 한다.)
- 1. 급여를 저장할 테이블을 생성하자.

예

```
CREATE TABLE SAL01(
  SALNO NUMBER(4) PRIMARY KEY,
  SAL NUMBER(7,2),
  EMPNO NUMBER(4) REFERENCES EMP01(EMPNO)
)
```

3.4 급여 메시지 트리거 작성하기(2)

- 2. 급여번호를 자동 생성하는 시퀀스를 정의하고 이 시퀀스로부터 일련번호를 얻어 급여번호에 부여한다.

예

```
CREATE SEQUENCE SAL01_SALNO_SEQ;
```

- 3. 다음과 같이 입력하시오.

예

```
CREATE OR REPLACE TRIGGER TRG_02
AFTER INSERT
ON EMP01
FOR EACH ROW
BEGIN
INSERT INTO SAL01 VALUES(
SAL01_SALNO_SEQ.NEXTVAL, 100, :NEW.EMPNO);
END;
/
```

3.4 급여 메시지 트리거 작성하기(3)

❖ 4. 사원 테이블에 행을 추가한다.

예

```
INSERT INTO EMP01 VALUES(2, '임요환', '프로게이머');
SELECT * FROM EMP01;
SELECT * FROM SAL01;
```

EMPNO	ENAME	JOB
1	김홍도	화가
2	임요환	프로그래머

SALNO	SAL	EMPNO
1	100	2

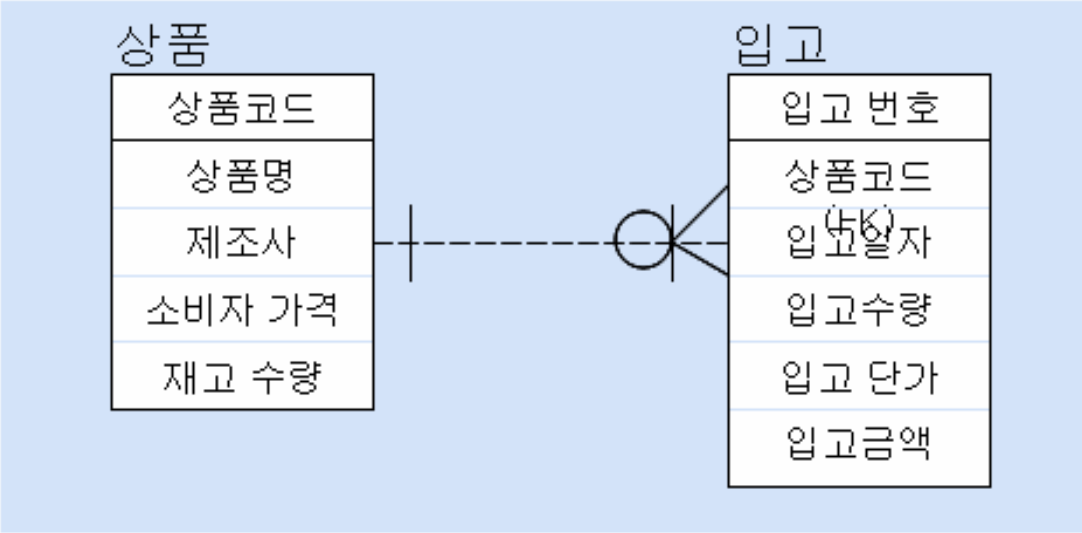
3.5 트리거 삭제

- DROP TRIGGER 다음에 삭제할 트리거 명을 기술한다.

예 **DROP TRIGGER TRG_02;**

4. 트리거 적용

- 상품 테이블의 예제를 통해서 실질적인 트리거의 적용 예를 살펴보도록 하자.



4.1 입고 트리거 작성(1)

- ex) 입고 테이블에 상품이 입력되면 입고 수량을 상품 테이블의 재고 수량에 추가하는 트리거 작성해보자.
- 1. 상품 테이블을 생성한다.

예

```
CREATE TABLE 상품(
상품코드 CHAR(6) PRIMARY KEY,
상품명 VARCHAR2(12) NOT NULL,
제조사 VARCHAR(12),
소비자가격 NUMBER(8),
재고수량 NUMBER DEFAULT 0
);
```

4.1 입고 트리거 작성(2)

- 2. 입고 테이블을 생성한다.

예

```
CREATE TABLE 입고(
입고번호 NUMBER(6) PRIMARY KEY,
상품코드 CHAR(6) REFERENCES 상품(상품코드),
입고일자 DATE DEFAULT SYSDATE,
입고수량 NUMBER(6),
입고단가 NUMBER(8),
입고금액 NUMBER(8)
);
```

- 3. 상품테이블의 재고수량 컬럼을 통해서 트리거의 적용 예를 살펴보도록 하자. 우선 상품 테이블에 다음과 같은 샘플 데이터를 입력해본다.

예

```
INSERT INTO 상품(상품코드, 상품명, 제조사, 소비자가격)
VALUES('A00001','세탁기', 'LG', 500);
INSERT INTO 상품(상품코드, 상품명, 제조사, 소비자가격)
VALUES('A00002','컴퓨터', 'LG', 700);
INSERT INTO 상품(상품코드, 상품명, 제조사, 소비자가격)
VALUES('A00003','냉장고', '삼성', 600);
```

4.1 입고 트리거 작성(3)

❖ 4. 입고 테이블에 상품이 입력되면 입고 수량을 상품 테이블의 재고 수량에 추가하는 트리거 작성을 위해 다음과 같이 입력하시오.

예

```
-- 입고 트리거
CREATE OR REPLACE TRIGGER TRG_04
AFTER INSERT ON 입고
FOR EACH ROW
BEGIN
UPDATE 상품
SET 재고수량 = 재고수량 + :NEW.입고수량
WHERE 상품코드 = :NEW.상품코드;
END;
/
```

4.1 입고 트리거 작성(4)

- 5. 트리거를 실행시킨 후 입고 테이블에 행을 추가한다. 입고 테이블에는 물론 상품 테이블의 재고 수량이 변경됨을 확인할 수 있다.

예

```
INSERT INTO 입고(입고번호, 상품코드, 입고수량, 입고단가, 입고금액) VALUES(1, 'A00001', 10, 320, 3200);
SELECT * FROM 입고;
SELECT * FROM 상품;
```

입고번호	상품코드	입고일자	입고수량	입고단가	입고금액
1	A00001	21/01/15	10	320	3200

상품코드	상품명	제조사	소비자가격	재고수량
A00001	세탁기	LG	500	10
A00002	컴퓨터	LG	700	0
A00003	냉장고	삼성	600	0

4.1 입고 트리거 작성(5)

- 6. 제품이 판매가 되어 입고 수량이 줄어들게 되는 경우 자동으로 재고수량이 변경됨을 확인할 수 있다.

예

```
INSERT INTO 입고(입고번호, 상품코드, 입고수량, 입고단가, 입고금액) VALUES(2, 'A00001', -5, 320, -1600);
SELECT * FROM 입고;
SELECT * FROM 상품;
```

입고번호	상품코드	입고일자	입고수량	입고단가	입고금액
1	A00001	21/01/15	10	320	3200
2	A00001	21/01/15	-5	320	-1600

상품코드	상품명	제조사	소비자가격	재고수량
A00001	세탁기	LG	500	5
A00002	컴퓨터	LG	700	0
A00003	냉장고	삼성	600	0