

22장 패키지화 트리거

이 장에서 다룰 내용

1 패키지

2 DBMS_OUTPUT 패키지

3 트리거

4 예제를 통한 트리거의 적용

01. 패키지

- ❖ 패키지의 사전적인 의미는 꾸러미입니다.
- ❖ 관련 있는 프로시저를 보다 효율적으로 관리하기 위해서 패키지 단위로 배포할 때 유용하게 사용됩니다.
- ❖ 패키지는 패키지 선언(명세부)과 패키지 몸체 선언(몸체부) 두 가지 모두를 정의해야 합니다.

01. 패키지

```
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;
/
```

01. PL/SQL 구조

- ❖ 몸체부내에는 여러 가지의 프로시저나 함수를 정의하고 있습니다.
- ❖ 몸체부에 정의한 프로시저나 함수는 앞장에서 배운 저장 프로시저와 저장 함수와 동일한 문법구조를 갖습니다.
- ❖ 명세부에는 몸체부에 정의한 함수들을 선언해 놓습니다.
- ❖ 패키지 내의 정의된 프로시저나 함수를 호출하는 방식은 다음과 같습니다.

EXECUTE [패키지명].[프로시저명]

〈실습하기〉 패키지 작성하기

앞장에서 작성했던 저장 프로시서와 저장 함수로 구성된 패키지를 생성해 봅시다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:PACK01.SQL)

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

〈실습하기〉 패키지 작성하기

```
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;
```

〈실습하기〉 패키지 작성하기

```
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;
END;
/
```


〈실습하기〉 패키지 작성하기

2. 입력을 마쳤으면 저장하고 SQL 파일을 실행시키기 위해서 @PACK01를 입력합니다. SQL 프롬프트에서 패키지내의 프로시저나 함수를 직접 호출합니다.

```
VARIABLE VAR_RES NUMBER;  
EXECUTE :VAR_RES := EXAM_PACK.CAL_BONUS(7788);  
PRINT VAR_RES;  
EXECUTE EXAM_PACK.CURSOR_SAMPLE02;
```

02. DBMS_OUTPUT 패키지

- ❖ PL/SQL에서 조회한 결과값을 출력하기 위해서 DBMS_OUTPUT에 대한 자세한 설명 없이 DBMS_OUTPUT를 사용해 왔습니다. 패키지에 대한 개념을 학습하였기 때문에 이제 DBMS_OUTPUT에 대해서 살펴보도록 합시다.
- ❖ 오라클을 설치하게 되면 특정 목적을 위해서 사용할 수 있도록 오라클 사에서 제공해 주는 패키지들이 설치됩니다. 이러한 패키지에는 오라클에서 제공되는 프로시저와 함수들을 관련된 것끼리 묶어서 집합으로 제공합니다. 패키지에 대한 정보를 출력해 보도록 합시다.

```
CONN system/manager  
DESC DBA_OBJECTS
```

02. DBMS_OUTPUT 패키지

- ❖ 패키지들은 목적에 따라 관련된 프로시저와 함수들을 관리합니다.

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

03. 트리거

❖ 다음은 트리거(trigger)의 사전적인 의미입니다.

- 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 변경되도록 하기 위해서 사용합니다.

❖ 트리거는 특정 동작을 이벤트로 그로 인해서만 실행되는 프로시저의 일종입니다.

03. 트리거

- ❖ 트리거를 만들기 위한 CREATE TRIGGER 문의 형식은 다음과 같습니다.

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

03 트리거

❖ 트리거의 타이밍

- [BEFORE] 타이밍은 어떤 테이블에 INSERT, UPDATE, DELETE 문이 실행될 때 해당 문장이 실행되기 전에 트리거가 가지고 있는 BEGIN ~ END 사이의 문장을 실행합니다.
- [AFTER] 타이밍은 INSERT, UPDATE, DELETE 문이 실행되고 난 후에 트리거가 가지고 있는 BEGIN ~ END 사이의 문장을 실행합니다.

❖ 트리거의 이벤트

- 사용자가 어떤 DML(INSERT, UPDATE, DELETE)문을 실행했을 때 트리거를 발생시킬 것인지를 결정합니다.

❖ 트리거의 몸체

- 해당 타이밍에 해당 이벤트가 발생하게 되면 실행될 기본 로직이 포함되는 부분으로 BEGIN ~ END에 기술합니다.

03 트리거

❖ 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 합니다.
- 문장 레벨 트리거는 어떤 사용자가 트리거가 설정되어 있는 테이블에 대해 DML(INSERT, UPDATE, DELETE)문을 실행할 때 단 한번만 트리거를 발생시킬 때 사용합니다.
- 행 레벨 트리거는 DML(INSERT, UPDATE, DELETE)문에 의해서 여러 개의 행이 변경된다면 각 행이 변경될 때마다 트리거를 발생시키는 방법입니다. 만약 5개의 행이 변경되면 5번 트리거가 발생합니다.

03 트리거

❖ 트리거 조건

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

〈실습하기〉 단순 메시지 출력하는 트리거 작성하기

사원 테이블에 새로운 데이터가 들어오면 '신입사원이 입사했습니다.'란 메시지를 출력도록 문장 레벨 트리거로 작성해봅시다.

1. 신입 사원의 정보를 추가할 사원 테이블을 새롭게 만들어 놓읍시다.
2. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:TRIG01.SQL)

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

〈실습하기〉 단순 메시지 출력하는 트리거 작성하기

3. 사원 테이블에 로우를 추가해 봅시다.

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

'전원지' 사원이 추가되자 '신입사원이 입사했습니다.'란 메시지가 출력되는 것을 보면 TRG_01 트리거가 수행되었음을 확인할 수 있습니다.

〈실습하기〉 급여 정보를 자동 추가하는 트리거 작성하기

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 합니다.)

1. 급여를 저장할 테이블을 생성하자.

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

〈실습하기〉 급여 정보를 자동 추가하는 트리거 작성하기

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

```
CREATE SEQUENCE SAL01_SALNO_SEQ;
```

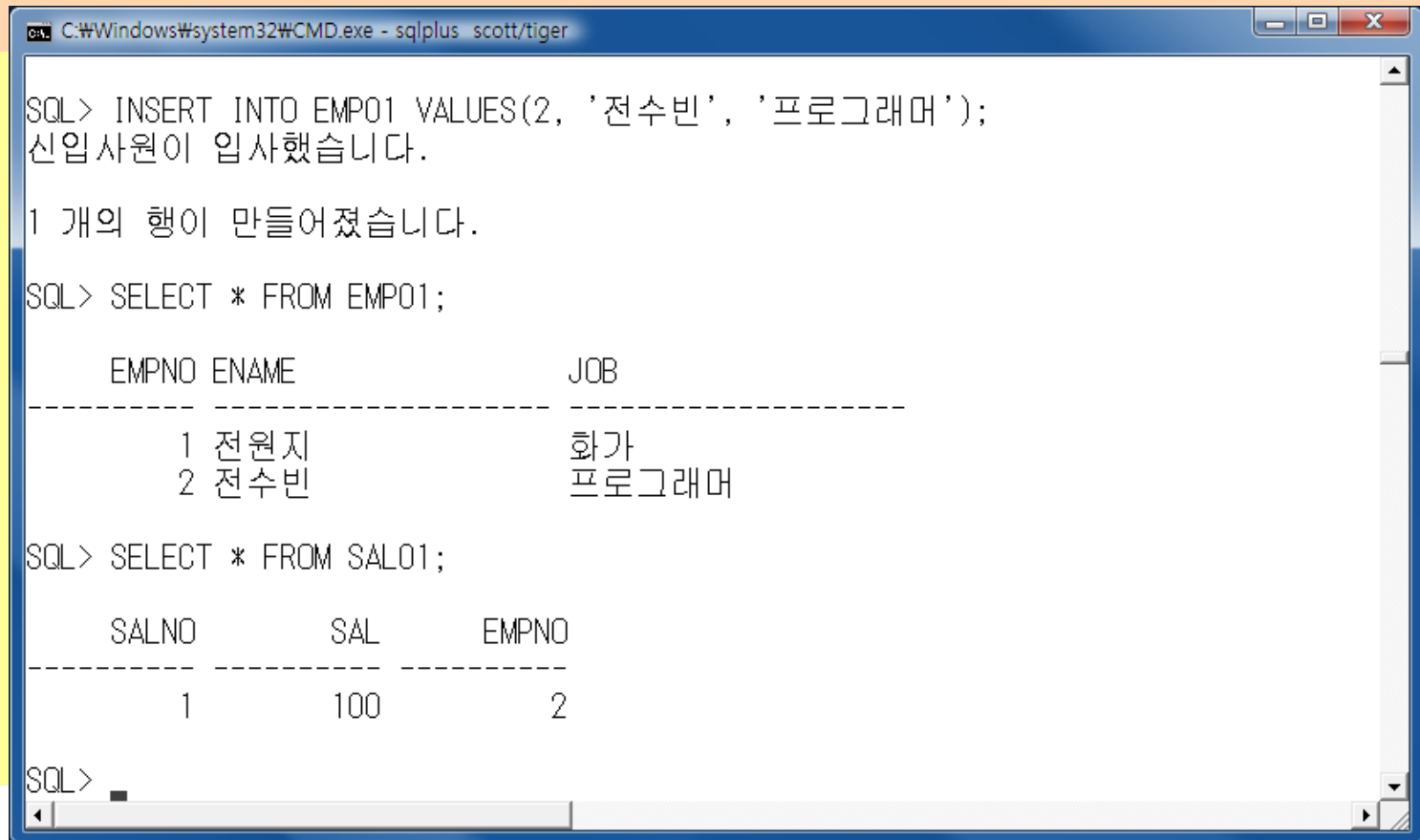
3. TRIG02.SQL에 다음과 같이 입력하시오.

```
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;  
/
```

<실습하기> 급여 정보를 자동 추가하는 트리거 작성하기

4. 사원 테이블에 로우를 추가합니다.

```
INSERT INTO EMP01 VALUES(2, '전수빈', '프로그래머');  
SELECT * FROM EMP01;  
SELECT * FROM SAL01;
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\CMD.exe - sqlplus scott/tiger". The user has entered the following SQL commands:

```
SQL> INSERT INTO EMP01 VALUES(2, '전수빈', '프로그래머');  
신입사원이 입사했습니다.  
  
1 개의 행이 만들어졌습니다.  
  
SQL> SELECT * FROM EMP01;  
  
EMPNO ENAME JOB  
-----  
1 전원지 화가  
2 전수빈 프로그래머  
  
SQL> SELECT * FROM SAL01;  
  
SALNO SAL EMPNO  
-----  
1 100 2  
  
SQL>
```

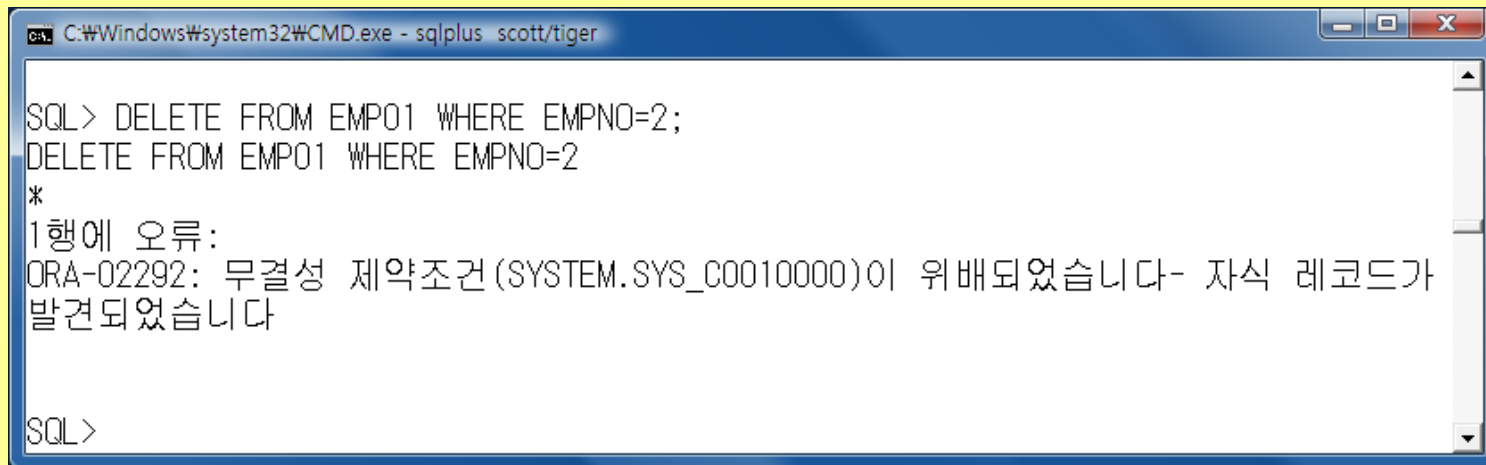
The output shows that the INSERT command was successful, adding a new employee (EMPNO 2, ENAME '전수빈', JOB '프로그래머'). The SELECT * FROM EMP01 command displays the contents of the EMP01 table, showing two rows: (1, '전원지', '화가') and (2, '전수빈', '프로그래머'). The SELECT * FROM SAL01 command displays the contents of the SAL01 table, showing one row: (1, 100, 2).

<실습하기> 급여 정보를 자동 삭제하는 트리거 작성하기

사원이 삭제되면 그 사원의 급여 정보도 자동 삭제되는 트리거를 작성해 보도록 합시다.

1. 이번에는 사원 테이블의 로우를 삭제해보자.

```
DELETE FROM EMP01 WHERE EMPNO=2;
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\CMD.exe - sqlplus scott/tiger". The user has entered the SQL command "DELETE FROM EMP01 WHERE EMPNO=2;". The prompt shows the command being entered twice. The first time, it shows "SQL> DELETE FROM EMP01 WHERE EMPNO=2;". The second time, it shows "DELETE FROM EMP01 WHERE EMPNO=2". Below the command, there is an asterisk "*" and the message "1행에 오류:". This is followed by the error message "ORA-02292: 무결성 제약조건 (SYSTEM.SYS_C0010000)이 위반되었습니다- 자식 레코드가 발견되었습니다". The prompt ends with "SQL>".

```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger

SQL> DELETE FROM EMP01 WHERE EMPNO=2;
DELETE FROM EMP01 WHERE EMPNO=2
*
1행에 오류:
ORA-02292: 무결성 제약조건 (SYSTEM.SYS_C0010000)이 위반되었습니다- 자식 레코드가
발견되었습니다

SQL>
```

〈실습하기〉 급여 정보를 자동 삭제하는 트리거 작성하기

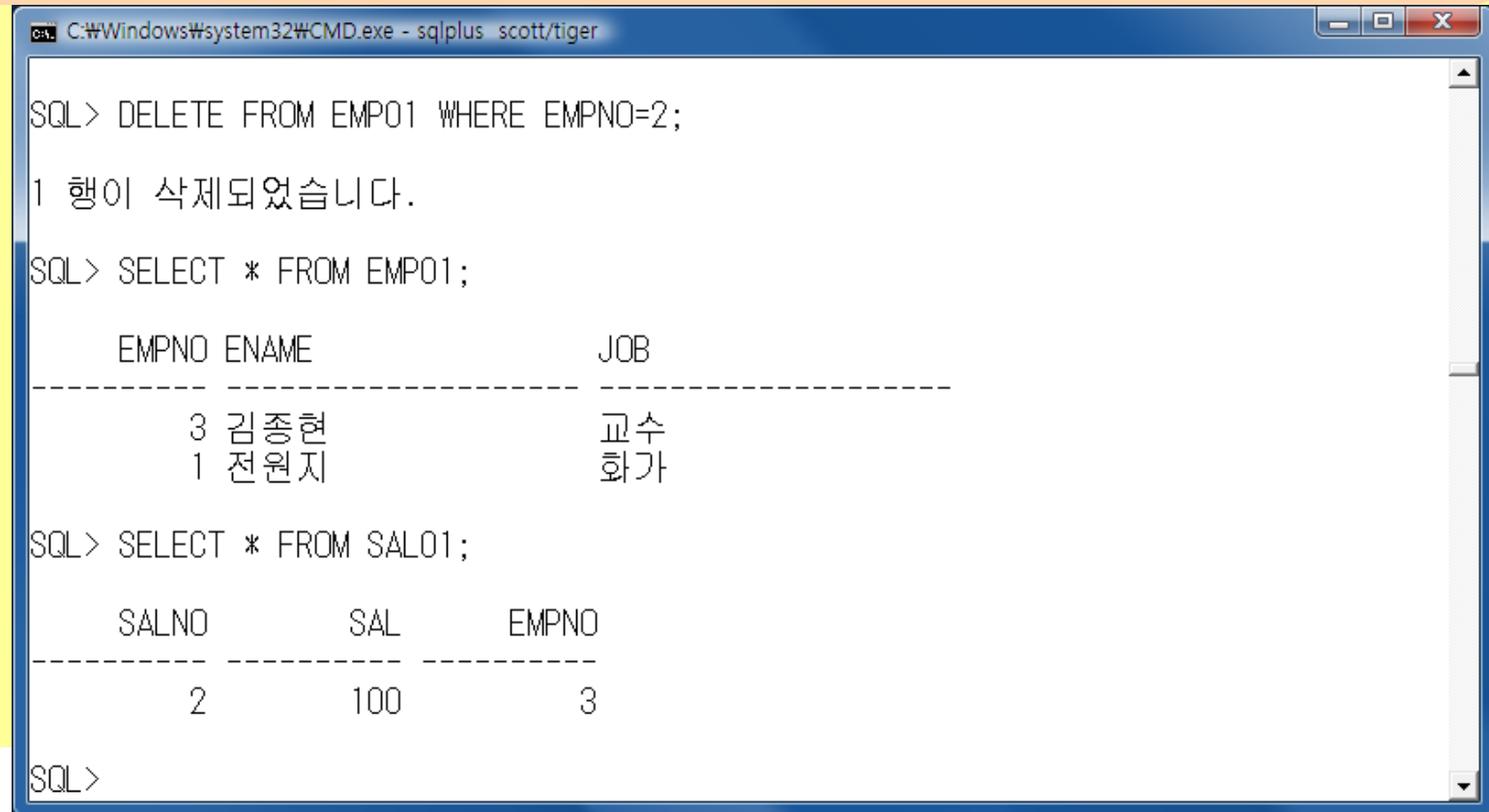
2. 사원번호 2를 급여 테이블에서 참조하고 있기 때문에 삭제가 불가능하다. 사원이 삭제되려면 그 사원의 급여 정보도 급여 테이블에서 삭제되어야 합니다. 사원의 정보가 제거 될 때 그 사원의 급여 정보도 함께 삭제하는 내용을 트리거로 작성하도록 합시다. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하시오.(파일이름:TRIG03.SQL)

```
CREATE OR REPLACE TRIGGER TRG_03
AFTER DELETE ON EMP01
FOR EACH ROW
BEGIN
DELETE FROM SAL01 WHERE EMPNO=:old.EMPNO;
END;
/
```

<실습하기> 급여 정보를 자동 삭제하는 트리거 작성하기

3. 사원 테이블에 로우를 삭제해 봅시다.

```
DELETE FROM EMP01 WHERE EMPNO=2;  
SELECT * FROM EMP01;  
SELECT * FROM SAL01;
```



```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger  
  
SQL> DELETE FROM EMP01 WHERE EMPNO=2;  
  
1 행이 삭제되었습니다.  
  
SQL> SELECT * FROM EMP01;  
  
      EMPNO ENAME      JOB  
-----  
        3 김종현      교수  
        1 전원지      화가  
  
SQL> SELECT * FROM SAL01;  
  
      SALNO      SAL      EMPNO  
-----  
        2      100        3  
  
SQL>
```

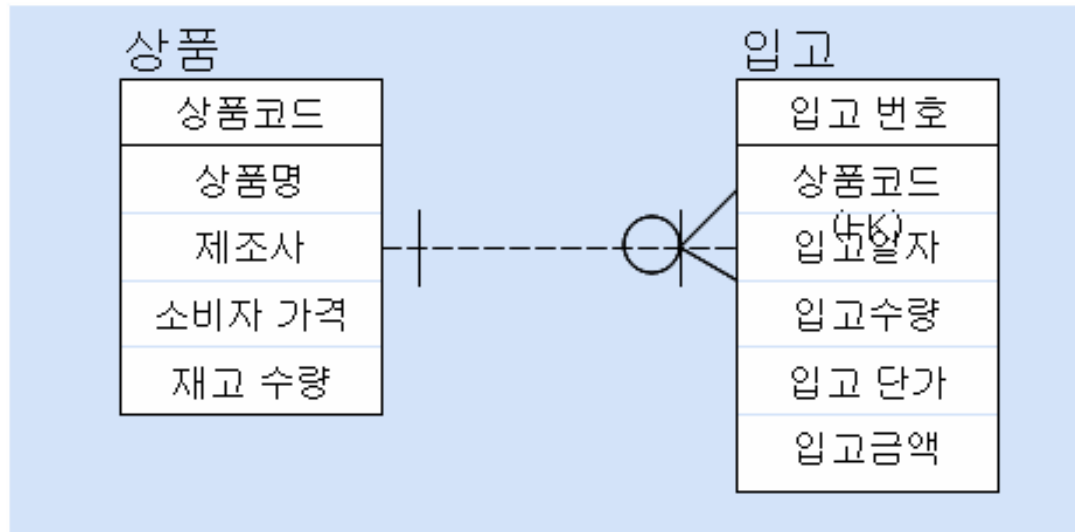

3.1 트리거 삭제

❖ **DROP TIGGER** 다음에 삭제할 트리거 명을 기술합니다.

```
DROP TRIGGER TRG_02;
```

04. 예제를 통한 트리거의 적용

- ❖ 상품 테이블의 예제를 통해서 실질적인 트리거의 적용 예를 살펴보도록 합시다.



〈실습하기〉 입고 트리거 작성하기

입고 테이블에 상품이 입력되면 입고 수량을 상품 테이블의 재고 수량에 추가하는 트리거 작성해 봅시다.

1. 상품 테이블을 생성합니다.

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

〈실습하기〉 입고 트리거 작성하기

2. 입고 테이블을 생성합니다.

```
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 실질적인 트리거의 적용 예를 살펴보고도
 록 하겠습니다. 우선 상품 테이블에 다음과 같은 샘플 데이터를 입력해봅시다.

```
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);
```

〈실습하기〉 입고 트리거 작성하기

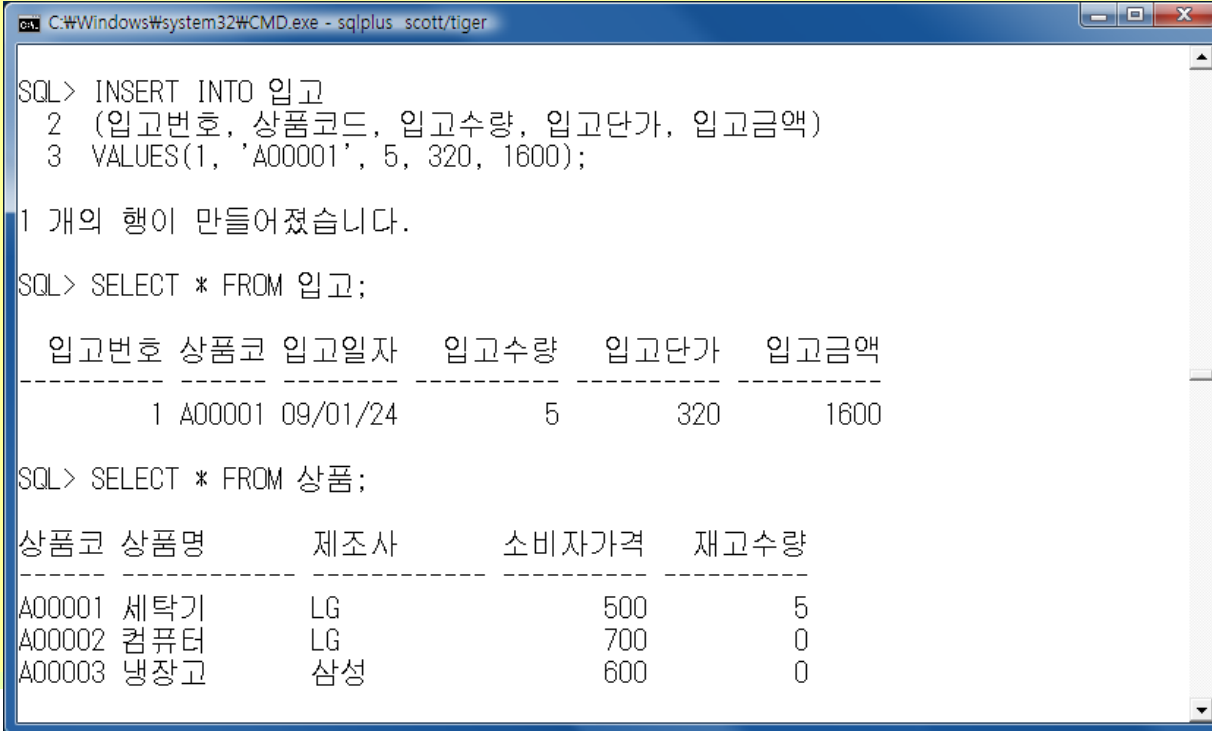
4. 입고 테이블에 상품이 입력되면 입고 수량을 상품 테이블의 재고 수량에 추가하는 트리거 작성을 위해 ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하시오.(파일이름:TRIG04.SQL)

```
-- 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;
/
```

〈실습하기〉 입고 트리거 작성하기

5. 트리거를 실행시킨 후 입고 테이블에 행을 추가합니다. 입고 테이블에는 물론 상품 테이블의 재고 수량이 변경됨을 확인할 수 있습니다.

```
INSERT INTO 입고(입고번호, 상품코드, 입고수량, 입고단가, 입고금액)
VALUES(1, 'A00001', 5, 320, 1600);
SELECT * FROM 입고;
SELECT * FROM 상품;
```



The screenshot shows a SQL command window titled "C:\Windows\system32\CMD.exe - sqlplus scott/tiger". The commands and their outputs are as follows:

```
SQL> INSERT INTO 입고
2  (입고번호, 상품코드, 입고수량, 입고단가, 입고금액)
3  VALUES(1, 'A00001', 5, 320, 1600);

1 개의 행이 만들어졌습니다.

SQL> SELECT * FROM 입고;
```

입고번호	상품코	입고일자	입고수량	입고단가	입고금액
1	A00001	09/01/24	5	320	1600

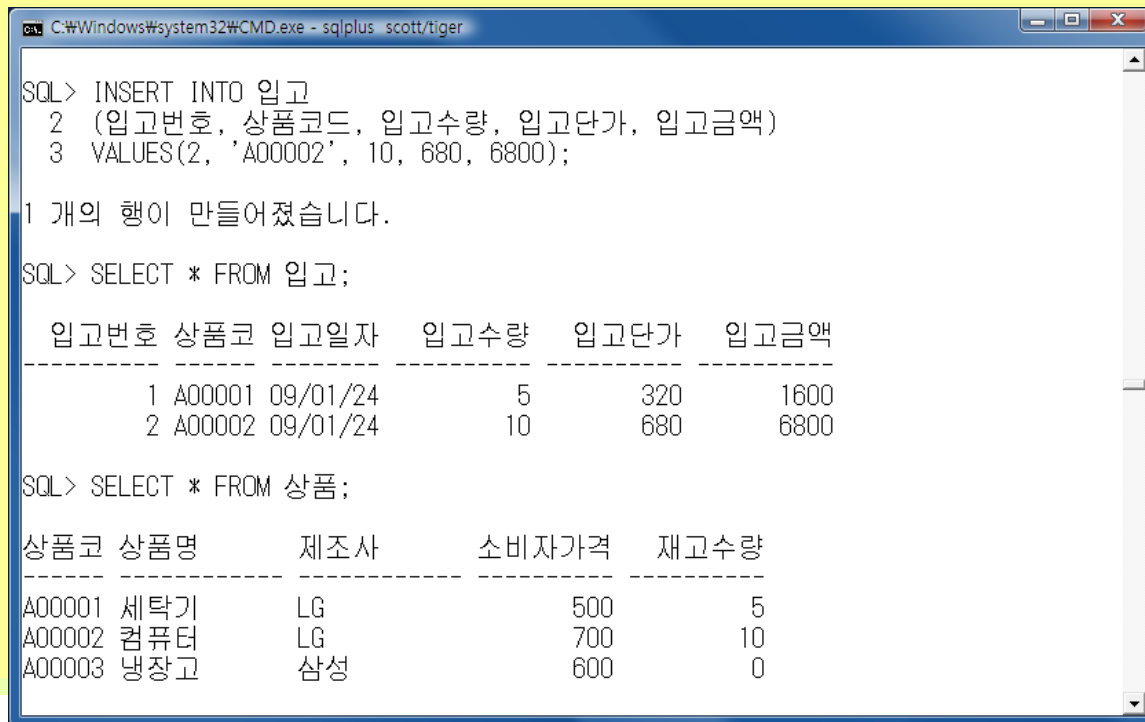
```
SQL> SELECT * FROM 상품;
```

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

〈실습하기〉 입고 트리거 작성하기

6. 입고 테이블에 상품이 입력되면 자동으로 상품 테이블의 재고 수량이 증가하게 됩니다. 입고 테이블에 또 다른 상품을 입력합니다.

```
INSERT INTO 입고(입고번호, 상품코드, 입고수량, 입고단가, 입고금액)
VALUES(2, 'A00002', 10, 680, 6800);
SELECT * FROM 입고;
SELECT * FROM 상품;
```



```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger

SQL> INSERT INTO 입고
  2 (입고번호, 상품코드, 입고수량, 입고단가, 입고금액)
  3 VALUES(2, 'A00002', 10, 680, 6800);

1 개의 행이 만들어졌습니다.

SQL> SELECT * FROM 입고;

입고번호 상품코 입고일자   입고수량   입고단가   입고금액
-----
      1 A00001 09/01/24         5       320      1600
      2 A00002 09/01/24        10       680      6800

SQL> SELECT * FROM 상품;

상품코 상품명   제조사   소비자가격   재고수량
-----
A00001 세탁기     LG        500           5
A00002 컴퓨터     LG        700          10
A00003 냉장고     삼성      600           0
```

〈실습하기〉 갱신 트리거 작성하기

이미 입고된 상품에 대해서 입고 수량이 변경되면 상품 테이블의 재고수량 역시 변경되어야 합니다. 이를 위한 갱신 트리거 작성해 봅시다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:TRIG05.SQL)

-- 갱신 트리거

CREATE OR REPLACE TRIGGER TRG03

AFTER UPDATE ON 입고

FOR EACH ROW

BEGIN

UPDATE 상품

SET 재고수량 = 재고수량 + (-:old.입고수량+:new.입고수량)

WHERE 상품코드 = :new.상품코드;

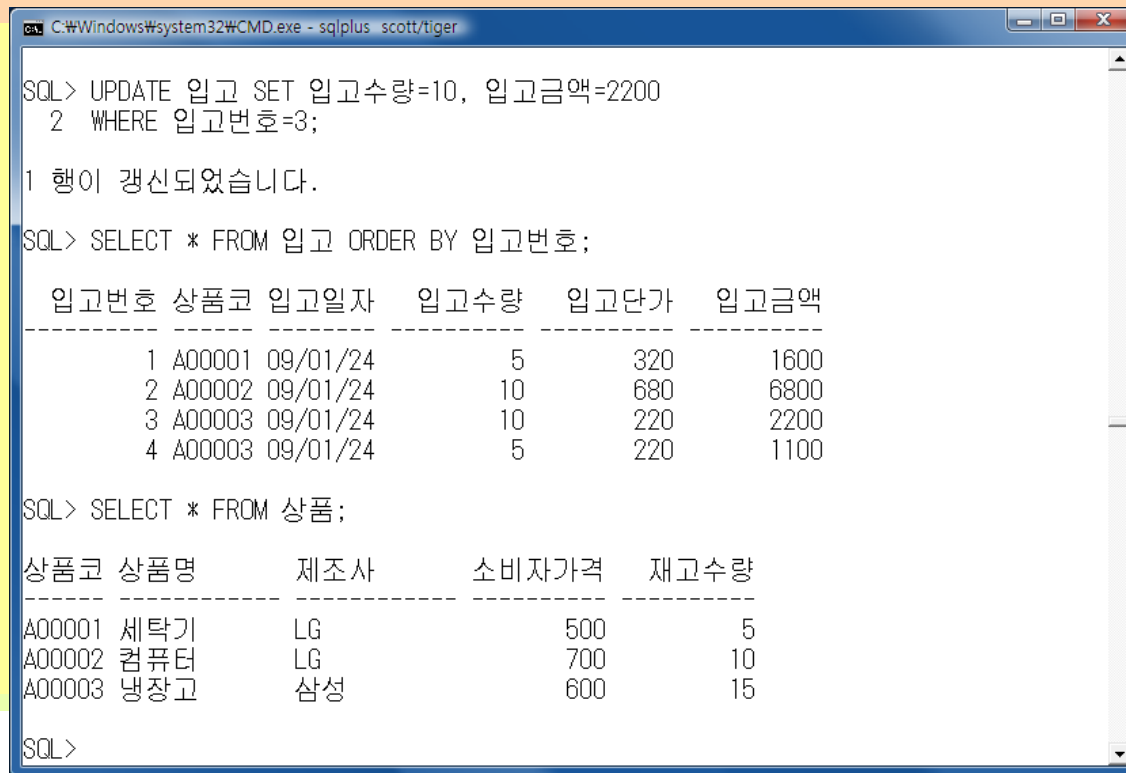
END;

/

〈실습하기〉 갱신 트리거 작성하기

2. 입고 번호 3번은 냉장고가 입고된 정보를 기록한 것으로서 입고 번호 3번의 입고 수량을 10으로 변경하였더니 냉장고의 재고 수량 역시 15로 변경되었습니다.

```
UPDATE 입고 SET 입고수량=10, 입고금액=2200
WHERE 입고번호=3;
SELECT * FROM 입고 ORDER BY 입고번호;
SELECT * FROM 상품;
```



```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger

SQL> UPDATE 입고 SET 입고수량=10, 입고금액=2200
2  WHERE 입고번호=3;

1 행이 갱신되었습니다.

SQL> SELECT * FROM 입고 ORDER BY 입고번호;

  입고번호  상품코드  입고일자  입고수량  입고단가  입고금액
-----
1  A00001  09/01/24         5      320      1600
2  A00002  09/01/24        10      680     6800
3  A00003  09/01/24        10      220     2200
4  A00003  09/01/24         5      220     1100

SQL> SELECT * FROM 상품;

상품코드  상품명      제조사      소비자가격  재고수량
-----
A00001   세탁기      LG           500          5
A00002   컴퓨터      LG           700         10
A00003   냉장고      삼성         600         15

SQL>
```

〈실습하기〉 삭제 트리거 작성하기

입고 테이블에서 입고되었던 상황이 삭제되면 상품 테이블에 재고수량에서 삭제된 입고수량 만큼을 빼는 삭제 트리거 작성해 봅시다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하시오.
(파일이름:TRIG06.SQL)

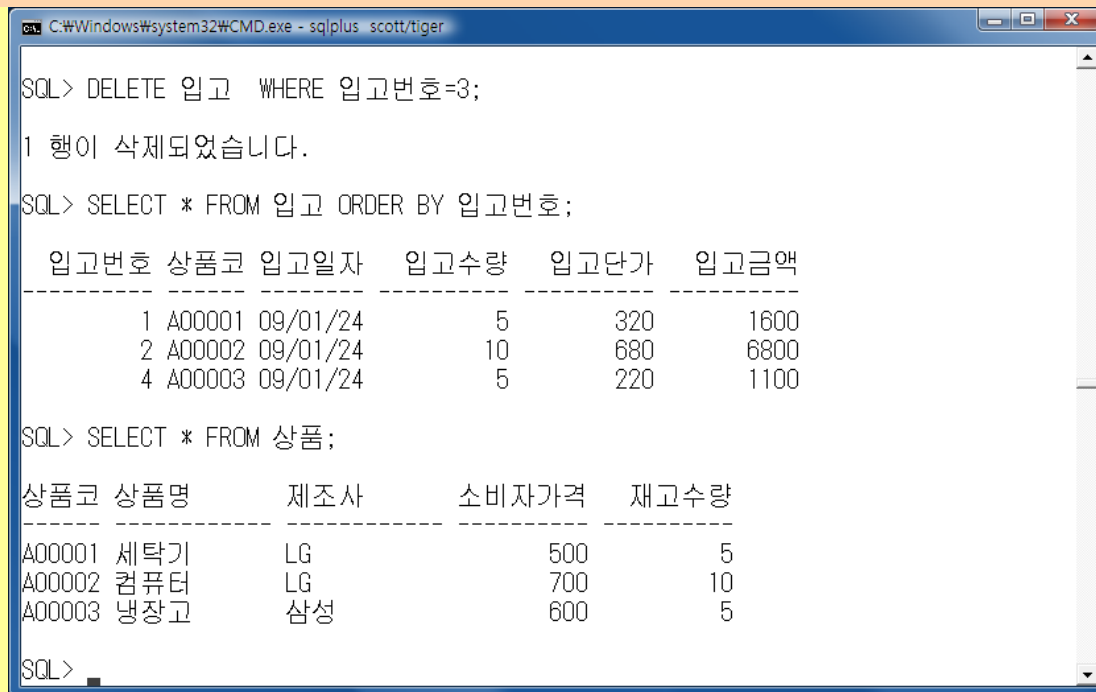
--삭제 트리거

```
CREATE OR REPLACE TRIGGER TRG04
AFTER DELETE ON 입고
FOR EACH ROW
BEGIN
UPDATE 상품
SET 재고수량 = 재고수량 - :old.입고수량
WHERE 상품코드 = :old.상품코드;
END;
/
```

〈실습하기〉 삭제 트리거 작성하기

2. 입고 번호 3번은 냉장고가 입고된 정보를 기록한 것으로서 입고 번호가 3번인 행을 삭제하였더니 냉장고의 재고 수량 역시 5로 변경되었습니다.

```
DELETE 입고 WHERE 입고번호=3;  
SELECT * FROM 입고 ORDER BY 입고번호;  
SELECT * FROM 상품;
```



```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger  
  
SQL> DELETE 입고 WHERE 입고번호=3;  
1 행이 삭제되었습니다.  
  
SQL> SELECT * FROM 입고 ORDER BY 입고번호;  
  
   입고번호  상품코드  입고일자   입고수량   입고단가   입고금액  
-----  
          1  A00001  09/01/24         5        320       1600  
          2  A00002  09/01/24        10        680       6800  
          4  A00003  09/01/24         5        220       1100  
  
SQL> SELECT * FROM 상품;  
  
  상품코드  상품명   제조사   소비자가격   재고수량  
-----  
A00001   세탁기    LG        500           5  
A00002   컴퓨터    LG        700          10  
A00003   냉장고    삼성      600           5  
  
SQL>
```

Thank You !