〈10주차 실습〉

PL/SQL(2)

Database Programming

실습 순서

- 커서
- 프로시저
- 함수
- 트리거
- 실습 과제

묵시적 커서


```
SQL> DECLARE
  2         v_sal emp.sal%TYPE;
  3         v_update_row NUMBER;
  4         v_empno emp.empno%TYPE;
  5  BEGIN
  6         v_empno := 7900;
  7         SELECT sal
  8         INTO v_sal
  9         FROM emp
 10         WHERE empno = v_empno;
 11
 12         IF SQL%FOUND THEN
 13               DBMS_OUTPUT.PUT_LINE('데이터 존재 : ' || v_sal);
 14         END IF;
 15
 16         UPDATE emp
 17         SET sal = sal*1.1
 18         WHERE empno = v_empno;
 19
 20         v_update_row := SQL%ROWCOUNT;
 21         COMMIT;
 22
 23         DBMS_OUTPUT.PUT_LINE('급여인상 사원수 : ' || v_update_row);
 24  END;
 25  /
데이터 존재 : 950
급여인상 사원수 : 1

PL/SQL 처리가 정상적으로 완료되었습니다.
```

```
SQL> SELECT * FROM emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 80/12/17        800                    20
      7499 ALLEN      SALESMAN        7698 81/02/20       1600        300         30
      7521 WARD       SALESMAN        7698 81/02/22       1250        500         30
      7566 JONES      MANAGER         7839 81/04/02       2975                    20
      7654 MARTIN     SALESMAN        7698 81/09/28       1250       1400         30
      7698 BLAKE      MANAGER         7839 81/05/01       2850                    30
      7782 CLARK      MANAGER         7839 81/06/09       2450                    10
      7788 SCOTT      ANALYST         7566 87/01/13       3000                    20
      7839 KING       PRESIDENT            81/11/17       5000                    10
      7844 TURNER     SALESMAN        7698 81/09/08       1500          0         30
      7876 ADAMS      CLERK           7788 87/10/23       1100                    20
      7900 JAMES      CLERK           7698 81/12/03        950                    30
      7902 FORD       ANALYST         7566 81/12/03       3000                    20
      7934 MILLER     CLERK           7782 82/01/23       1300                    10

14 개의 행이 선택되었습니다.
```

```
SQL> SELECT * FROM emp
  2  WHERE empno = 7900;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 81/12/03       1045                    30

SQL>
```

명시적 커서(1/2)


```
SQL> DECLARE
  2         CURSOR dept_cnt IS
  3         SELECT b.dname, COUNT(a.empno) cnt
  4         FROM emp a, dept b
  5         WHERE a.deptno = b.deptno
  6         AND b.deptno = 30
  7         GROUP BY b.dname;
  8
  9         v_dname dept.dname%TYPE;
 10         emp_cnt NUMBER;
 11   BEGIN
 12         OPEN dept_cnt;
 13         FETCH dept_cnt INTO v_dname, emp_cnt;
 14         DBMS_OUTPUT.PUT_LINE('부서명 : ' || v_dname);
 15         DBMS_OUTPUT.PUT_LINE('사원수 : ' || emp_cnt);
 16         CLOSE dept_cnt;
 17   EXCEPTION
 18         WHEN OTHERS THEN
 19               DBMS_OUTPUT.PUT_LINE(SQLERRM || '에러 발생');
 20   END;
 21   /
부서명 : SALES
사원수 : 6

PL/SQL 처리가 정상적으로 완료되었습니다.
```

명시적 커서(2/2)


```
SQL> DECLARE
  2         v_empno emp.empno%TYPE;
  3         v_ename emp.ename%TYPE;
  4         v_sal emp.sal%TYPE;
  5
  6         CURSOR emp_list IS
  7         SELECT empno, ename, sal
  8         FROM emp;
  9   BEGIN
 10         OPEN emp_list;
 11         LOOP
 12               FETCH emp_list INTO v_empno, v_ename, v_sal;
 13                     EXIT WHEN emp_list%NOTFOUND;
 14         END LOOP;
 15         DBMS_OUTPUT.PUT_LINE('전체 데이터 수 : ' || emp_list%ROWCOUNT);
 16         CLOSE emp_list;
 17   EXCEPTION
 18         WHEN OTHERS THEN
 19               DBMS_OUTPUT.PUT_LINE('ERR MESSAGE : ' || SQLERRM);
 20   END;
 21   /
전체 데이터 수 : 14

PL/SQL 처리가 정상적으로 완료되었습니다.
```

FOR문에서 커서 사용


```
SQL> DECLARE
  2         CURSOR dept_cnt IS
  3         SELECT b.dname, COUNT(a.empno) cnt
  4         FROM emp a, dept b
  5         WHERE a.deptno = b.deptno
  6         GROUP BY b.dname;
  7   BEGIN
  8         FOR emp_list IN dept_cnt LOOP
  9               DBMS_OUTPUT.PUT_LINE('부서명 : ' || emp_list.dname);
 10               DBMS_OUTPUT.PUT_LINE('사원수 : ' || emp_list.cnt);
 11         END LOOP;
 12   EXCEPTION
 13         WHEN OTHERS THEN
 14               DBMS_OUTPUT.PUT_LINE(SQLERRM || '에러 발생');
 15   END;
 16   /
부서명 : ACCOUNTING
사원수 : 3
부서명 : RESEARCH
사원수 : 5
부서명 : SALES
사원수 : 6

PL/SQL 처리가 정상적으로 완료되었습니다.
```

파라미터가 있는 커서


```
SQL> DECLARE
  2         param_deptno dept.deptno%TYPE;
  3         CURSOR emp_list(v_deptno emp.deptno%TYPE) IS
  4         SELECT ename
  5         FROM emp
  6         WHERE deptno = v_deptno;
  7  BEGIN
  8         DBMS_OUTPUT.PUT_LINE(' ** 입력한 부서 사람들 ** ');
  9
 10         param_deptno := 10;
 11         FOR emplst IN emp_list(param_deptno) LOOP
 12               DBMS_OUTPUT.PUT_LINE('이름  : ' || emplst.ename);
 13         END LOOP;
 14  EXCEPTION
 15         WHEN OTHERS THEN
 16               DBMS_OUTPUT.PUT_LINE('ERR MESSAGE : ' || SQLERRM);
 17  END;
 18  /
** 입력한 부서 사람들 **
이름  : CLARK
이름  : KING
이름  : MILLER

PL/SQL 처리가 정상적으로 완료되었습니다.
```

WHERE CURRENT OF


```
SQL> DECLARE
  2         CURSOR emp_list IS
  3         SELECT empno
  4         FROM emp
  5         WHERE empno = 7934
  6         FOR UPDATE;
  7   BEGIN
  8         FOR emplst IN emp_list LOOP
  9               UPDATE emp
 10               SET job = 'SALESMAN'
 11               WHERE CURRENT OF emp_list;
 12               DBMS_OUTPUT.PUT_LINE('수정 성공');
 13         END LOOP;
 14
 15         COMMIT;
 16
 17   EXCEPTION
 18         WHEN OTHERS THEN
 19               DBMS_OUTPUT.PUT_LINE('ERR MESSAGE : ' || SQLERRM);
 20   END;
 21   /
수정 성공

PL/SQL 처리가 정상적으로 완료되었습니다.
```

```
SQL> SELECT * FROM emp
  2   WHERE empno = 7934;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7934 MILLER     SALESMAN         7782 82/01/23       1300                    10
```

프로시저

- **프로시저 코드**

```
SQL> CREATE OR REPLACE PROCEDURE update_sal
  2         /* IN Parameter */
  3         (v_empno IN NUMBER)
  4   IS
  5   BEGIN
  6         UPDATE emp
  7         SET sal = sal * 1.1
  8         WHERE empno = v_empno;
  9         COMMIT;
 10   END;
 11   /

프로시저가 생성되었습니다.
```

- **실행 예제**

```
SQL> SHOW ERROR
오류가 없음.
SQL> EXECUTE update_sal(7369);

PL/SQL 처리가 정상적으로 완료되었습니다.
```

함수

- **함수 코드**

```
SQL> CREATE OR REPLACE FUNCTION FC_update_sal
  2         (v_empno IN NUMBER)
  3         RETURN NUMBER
  4  IS
  5         v_sal emp.sal%TYPE;
  6  BEGIN
  7         UPDATE emp
  8         SET sal = sal * 1.1
  9         WHERE empno = v_empno;
 10
 11         COMMIT;
 12
 13         SELECT sal
 14         INTO v_sal
 15         FROM emp
 16         WHERE empno = v_empno;
 17  RETURN v_sal;
 18  END;
 19  /

함수가 생성되었습니다.
```

- **실행 예제**

```
SQL> VAR salary NUMBER;
SQL> EXECUTE :salary := FC_update_sal(7900);

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> PRINT salary;

    SALARY
----------
    1149.5
```

트리거(1/2)

- **트리거 코드**

```
SQL> CREATE OR REPLACE TRIGGER trigger_test
  2  BEFORE
  3  UPDATE ON dept
  4  FOR EACH ROW
  5  BEGIN
  6      DBMS_OUTPUT.PUT_LINE('변경 전 컬럼 값 : ' || :old.dname);
  7      DBMS_OUTPUT.PUT_LINE('변경 후 컬럼 값 : ' || :new.dname);
  8  END;
  9  /

트리거가 생성되었습니다.
```

```
SQL> SELECT * FROM dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

- **실행 예제**

```
SQL> UPDATE dept
  2  SET dname = 'MANAGEMENT'
  3  WHERE deptno = 10;
변경 전 컬럼 값 : ACCOUNTING
변경 후 컬럼 값 : MANAGEMENT

1 행이 업데이트되었습니다.

SQL> COMMIT;

커밋이 완료되었습니다.
```

```
SQL> SELECT * FROM dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 MANAGEMENT     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

트리거(2/2)

- **트리거 코드**

```
SQL> CREATE OR REPLACE trigger sum_trigger
  2  BEFORE INSERT OR UPDATE ON emp
  3  FOR EACH ROW
  4  DECLARE
  5       avg_sal NUMBER;
  6  BEGIN
  7       SELECT ROUND(AVG(sal),3)
  8       INTO avg_sal
  9       FROM emp;
 10       DBMS_OUTPUT.PUT_LINE('급여 평균 : ' || avg_sal);
 11  END;
 12  /

트리거가 생성되었습니다.
```

- **실행 예제**

```
SQL> INSERT INTO emp(empno, ename, job, hiredate, sal, deptno)
  2  VALUES(1000, 'LION', 'SALESMAN', SYSDATE, 5000, 30);
급여 평균 : 2093.179

1 개의 행이 만들어졌습니다.

SQL> COMMIT;

커밋이 완료되었습니다.
```

-- INSERT문이 실행되기 전까지의 급여 평균 출력

실습 과제

Q1> EMP 테이블의 직책 종류를 이름 순서대로 조회하고 직책의 수를 조회하는 SCRIPT를 작성하시오.
(%TYPE, 묵시적 CURSOR, FOR문 사용)

```
직책명
-------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

직책의 수: 5

PL/SQL 처리가 정상적으로 완료되었습니다.
```

줄바꿈 **:chr(10)||chr(13)**

Q2> DEPT 테이블의 내용을 조회하는 SCRIPT를 작성하시오.
(%ROWTYPE, 명시적 CURSOR, LOOP 사용)

```
부서번호      부서명          위      치
--------    ---------      --------
10          MANAGEMENT     NEW YORK
20          RESEARCH       DALLAS
30          SALES          CHICAGO
40          OPERATIONS     BOSTON

PL/SQL 처리가 정상적으로 완료되었습니다.
```

데이터 공간 맞춤 : **rpad**함수 사용
Ex) rpad(컬럼명, 데이터 길이(숫자))

실습 과제

Q3> 새로운 사원의 정보를 입력 받아 EMP 테이블에 등록하는 프로시저를 작성하고 실행하시오. 단, 새로운 사원의 부서번호는 해당 사원의 관리자(MGR)의 부서번호를 조회하여 입력하고, HIREDATE는 현재 일자로 입력한다. 또한, 사원의 직책(JOB)이 'SALESMAN' 인 경우, COMM에 10을 입력하고 나머지 직책에는 NULL을 입력한다. 추가로 UNIQUE 제약이 있는 사원번호에 대해 중복되는 데이터를 입력할 경우 '중복된 사원번호입니다.' 라는 오류 메시지를 출력한다.

(%TYPE, EXCEPTION-미리 정의된 예외 처리 사용)

※ 프로시저 실행 후 emp 테이블을 조회하시오.

```
SQL> EXECUTE emp_input(7900, 'ALVIN', 'SALESMAN', 7788, SYSDATE, 1500);
중복된 사원번호 입니다.

PL/SQL 처리가 정상적으로 완료되었습니다.
```

```
SQL> EXECUTE emp_input(7904, 'ALVIN', 'SALESMAN', 7788, SYSDATE, 1500);

PL/SQL 처리가 정상적으로 완료되었습니다.
```

실습 과제

Q4> EMP 테이블에서 사원의 번호를 입력 받아 부서번호를 출력하는 함수를 작성하여라.

```
SQL> VAR g_deptno NUMBER
SQL> EXECUTE :g_deptno := empno_deptno(7788);
부서번호 : 20

PL/SQL 처리가 정상적으로 완료되었습니다.
```

Q5> SHOPS 테이블에 데이터를 새로 입력할 때 실행하기 전에 총 수익, 총 비용, 총 순이익을 출력하는 트리거를 작성하고 실행해보자.

(%TYPE 사용, 순이익=수익-비용)

```
SQL> INSERT INTO SHOPS VALUES (500, '동국상점5', 2500000, 800000);
<INSERT문이 실행되기 전까지의 손익 현황>
총 수익 : 11300000
총 비용 : 4900000
총 순이익 : 6400000

1 개의 행이 만들어졌습니다.
```

실습 과제

- 제출 방식 : E-Class를 통하여 제출
- 제출 내용 : spool file
- 제출 형식 : 학번_이름_주차
  - Ex)학번_홍길동_10주차.sql
- 제출 기한 : 수업 시작 시간으로 부터 24시간 이내 제출
  - 제출 기한 위반 시 감점 기준
    - 지각 제출 시 과제 점수에서 40% 감점
    - 1일 초과 당 20% 추가 감점 (단, 4일 이후 제출 불가)