데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5주차: Cursor

[학습목표]

현업에서 PL/SQL을 사용하는 패턴을 보면 대부분이 데이터 처리를 위해 Cursor를 사용합니다. Cursor의 개념 및 사용방법을 익혀 두어야 합니다.

[학습목차]

5-1 커서(Cursor)의 용어 및 개념
5-2 커서(Cursor)의 종류
5-3 명시적 커서(Explicit Cursor)
5-4 커서 속성(Cursor Attribute)
5-5 명시적 커서의 4 가지 유형

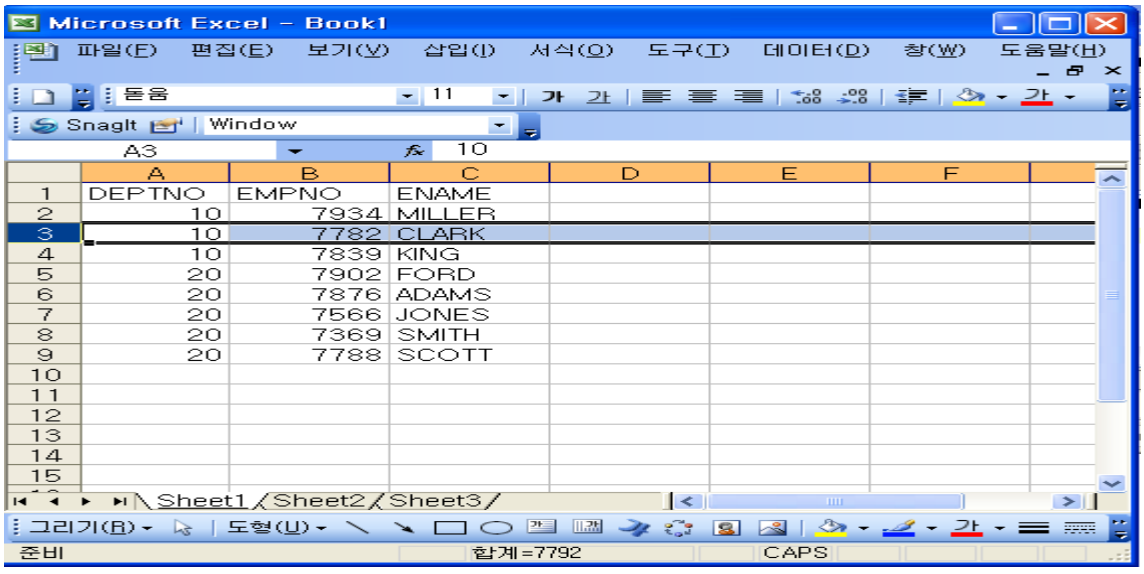
[애니메이션]

- 애니 1

까만 바탕의 모니터 화면에 현재의 위치를 가리키는 커서가 깜빡인다.
방향키를 누르면 해당 방향으로 이동하면서 커서가 깜빡이며 현재의 위치를 나타낸다.

- 애니 2

엑셀 Sheet 에서 커서가 깜빡이며 현재 데이터의 위치를 가리킨다. 방향키를 누름에 따라(상하로) 커서가 이동하면서 깜빡이며 다음 데이터나 이전 데이터를 표시한다.



데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5-1 커서(Cursor) 용어 및 개념

5-1-1 사전적 의미 :

①[일반용어] 계산자,측량기 등의 눈금이 달린 투명한 움직이는 관

②[컴퓨터 용어]깜빡이,컴퓨터 화면에서 점멸하여 현재의 위치를 알리는 것

<참고> 영어 단어의 사전적 의미나 우리가 일반적으로 알고 있는 컴퓨터용어의 의미는 무엇인가의 위치를 가리키는(Pointer)의 의미를 가지고 있습니다. 일반 컴퓨터 용어상 커서라는 단어와 현 Chapter에서 설명하려는 커서는 동일한 단어와 유사한 의미를 가집니다.

5-1-2 SQL,PL/SQL 관점의 의미

사용자(User)가 SQL을 전송하면 Oracle DBMS는 해당 SQL을 실행하기 위한 메모리 공간이 필요 하다.

예를 들면

▣ SQL을 실행하여 리턴되는 Rows

▣ 공유 메모리 영역(Shared Pool)에 저장되어 있는 Parsed Query에 대한 포인터

등을 저장하기 위해서 메모리 공간(Context Area)이 필요 합니다.

정의상으로는 Context Area 나 Cursor라는 용어가 동일한 의미로 사용되나 일반적으로 Context Area는 할당된 영역을 의미하며 Cursor는 해당 영역을 가리키는 포인터나 해당 영역을 제어하기 위한 핸들러(Handler)의 의미로 사용된다.

<참고 > Shared Pool, Parsed Query에 대한 내용은 SQL Tuning 과정이나 Oracle Admin 과정에서 별도의 학습을 해야 하며 내용이 어렵다면 마치 컴퓨터의 커서가 현재의 위치를 가리키는 역할을 하듯이 SQL,PL/SQL 관점에서도 SQL 실행을 위해서 할당된 메모리 영역의 위치를 가리킨다(즉 Pointer 역할) 라는 정도로만 기억을 하시기 바랍니다.

<참고> 개개의 서버 프로세스(Server Process)가 독립적으로 사용하는 PGA(Program Global Area) 라는 메모리 영역내에 SQL 처리를 위한 메모리 영역(Context Area)이 생성 됩니다.

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

<참고> 서버 프로세스(Server Process)란 사용자가(ex SQL*Plus) 전송한 SQL,PL/SQL을 실행하는 프로세스

5-2 커서(Cursor)의 종류

사용자 관점 에서 PL/SQL 커서는 2가지 유형 입니다.

① 명시적 커서(Implicit Cursor)

② 암시적 커서(Explicit Cursor)

5_CURSOR_1.SQL

```
DECLARE
  1  CURSOR  CUR_EMP IS
          SELECT EMPNO,JOB,SAL,COMM FROM EMP WHERE  DEPTNO = 10;
      V_ENAME     VARCHAR2(10);
      V_JOB       VARCHAR2(9);
      V_SAL       NUMBER(7,2);
      V_COMM      NUMBER(7,2);
BEGIN
      OPEN  CUR_EMP;
      LOOP
          FETCH  CUR_EMP  INTO  V_ENAME,V_JOB,V_SAL,V_COMM ;
          EXIT   WHEN  CUR_EMP%NOTFOUND;

       2  INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
              VALUES(V_ENAME,V_JOB,V_SAL,V_COMM);
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('TOTAL '||TO_CHAR(CUR_EMP%ROWCOUNT)||' rows processed');
      CLOSE   CUR_EMP;
      COMMIT;
END;
/
```

SELECT * FROM BONUS;

5-2-1 암시적 커서(Implicit Cursor)

암시적이라는 의미는 자동(Automatic)적 으로 라는 의미를 내포 한다.

PL/SQL Block 내에서 사용되는 모든 DML(Data Manipulation Language) 명령어 와 SELECT은 암시적 커서를 통해 처리 된다.

즉 Oracle DBMS가 SQL를 처리하기 위해 자동으로 즉 암시적으로 커서(Impicit Cursor)를 정의하여 해당 SQL을 실행 합니다.

Oracle DBMS가 커서를 아래의 4단계로 처리한다.

① Declare(정의) ② Open ③ Fetch ④ Close

암시적 커서는 4단계를 개발자(사용자)가 아닌 Oracle DBMS에서 자동으로 (암시적으로) 처리 합니다.

5-2-2 명시적 커서(Explicit Cursor)

명시적이라는 의미는 수동(Manual)적으로 라는 의미를 내포합니다.
Multiple Rows를 선택하여 처리 하기 위하여 사용자가 Explicit Cursor를 정의하여 해당 SQL을 실행 합니다.
명시적 커서는 위의 예제를 보시면 4단계를 거쳐 사용됩니다.
선언부(DECLARE)에서 ① Declare(정의)
실행부(BEGIN ~ END)에서 ② OPEN ③ FETCH ④CLOSE 를 통해
명시적으로 개발자가 제어를 하게 됩니다.

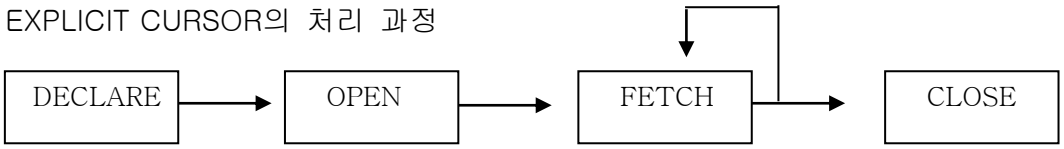
<참고 > 명시적 커서의 사용 목적은 다중행(Multiple Rows)을 조회 하고 처리

5-3 명시적 커서(Explicit Cursor)

EXPLICIT CURSOR는 PL/SQL BLOCK내에서 다중행(MULTIPLE-ROW)를 조회하여 데이타를 처리하려 할때 사용합니다.

명시적 커서는 4단계를 통해 처리 됩니다.

EXPLICIT CURSOR의 처리 과정



① DECLARE CURSOR

CURSOR의 이름 과 접근하는 데이터의 대상집합을 정의

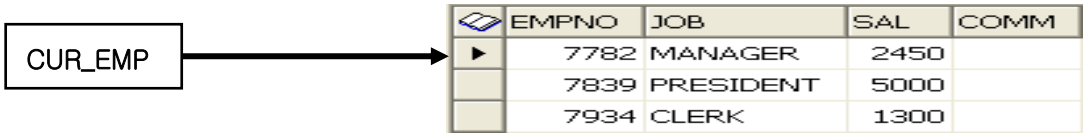
예제를 보면
CUR_EMP는 커서의 이름 입니다.
SELECT 이하는 커서를 통해 접근하는 대상 데이터 집합을 정의 합니다.

② OPEN CURSOR

BIND VARIABLES AND EXECUTE SQL 를 수행.

예제를 보면
BIND VARIABLES은 5-5 절에서 보게 됩니다.
EXECUTE SQL은 커서를 Open 하는 시점에 SELECT 구문을
실행하여 아래와 같은 결과 집합(ACTIVE SET 또는 RESULT SET)을

만드는 것을 의미 합니다. 아래의 결과 집합에 CUR_EMP 라는 이름을 붙여 커서의 이름을 가지고 제어 처리를 하는 핸들러(Handler)로 사용



EMPNO	JOB	SAL	COMM
7782	MANAGER	2450	
7839	PRESIDENT	5000	
7934	CLERK	1300	

③ FETCH CURSOR

CURSOR로 부터 CURRENT ROW의 DATA를 변수에 저장(LOAD) FETCH의 가져 온다는 사전적 의미를 가지고 있습니다.
즉 결과 집합(Active Set or Result Set)에서 현재 레코드의 데이터를 INTO 이하의 변수에 저장 합니다.
위의 결과 집합은 3개의 Row를 가지고 있으므로 LOOP ~ END LOOP 구간을 3번 반복 하며 1 Row 씩을 Fetch 를 수행 합니다.

< 참고>
Fetch는 1 Row 단위로 수행 한다. 1000만건의 대량의 데이터를 가지는 결과 집합인 경우 Fetch를 1000만번 수행해야 한다.
PL/SQL 학습 과정의 후반부인 Bulk Binding 과정에서 다루게 되지만 여러분도 미리 고민을 해보시기 바랍니다.

주의: ① CURSOR와 변수 사이에는 DATA TYPE, 갯수가 일치해야한다.

② FETCH시 DATA가 없는 경우에 NO DATA FOUND를 EXCEPTION을 유발하지 않기 때문에 사용자가 CHECK해주어야 한다.

예제상에서는
CUR_EMP%NOTFOUND
는 커서명(CUR_EMP)+속성연산자(%)+커서속성자(NOTFOUND)
를 통해 더 이상 FETCH할 데이터가 없는 경우 Loop를 종료하는 제어 처리를 수행한다.

④ CLOSE CURSOR

커서(CURSOR)를 종료한다. 결과집합(ACTIVE SET or RESULT SET)을 해제 하고 해당 자원을 반납 한다.
커서를 종료하는 경우에는 더 이상 결과 집합에 접근 할수 없다.

<참고> CLOSE CURSOR
① SELECT한 결과에 대한 처리를 다한후 CLOSE
② CURSOR를 REOPEN하기 위해서 CLOSE

<참고> C언어와 같은 3세대 언어에서 파일 처리 프로그램을 개발해본 경험이 있는 경우 PL/SQL 커서(Cursor)에 대한 개념을 쉽게 접근 할수 있다.명시적 커서를 핸들링하는 방법이 파일을 핸들링하는 방법과 유사 하기 때문에

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5-4 커서 속성(Cursor Attribute)

커서 속성자는 명시적/암시적이 동일하지만 의미상 약간의 차이가 나타난다.
각 의미의 차이를 기억 하십시요.

커서속성자	IMPLICIT CURSOR	EXPLICIT CURSOR
%ROWCOUNT	SQL문에의해 영향받은 ROW 총 갯수	FETCH된 누적 갯수
%FOUND	SQL문에의해 영향받은ROW 존재유무 TRUE or FALSE 리턴	현재 FETCH된 ROW존재유무 TRUE or FALSE 리턴
%NOTFOUND	FOUND의 반대값	FOUND의 반대값
%ISOPEN	항상FALSE이다	CURSOR의 Open 상태 확인

<참고>

PL/SQL 프로그래밍시 처리된 ROW의 개수를 Check해야할 필요가 종종 있습니다. ROWCOUNT 라는 커서 속성자가 유용한 이유 입니다.
명시적 커서를 사용시 LOOP를 통해 반복적으로 FETCH를 수행하며 LOOP 종료 시점을 결정하기 위해 FOUND 를 필수적으로 사용하게 되므로 기억을 해두시기 바랍니다.

암시적 커서 속성자 예제

```
IF SQL%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('SQL%FOUND = TRUE ');
ELSE
    DBMS_OUTPUT.PUT_LINE('SQL%NOTFOUND = FALSE ');
END IF;
```

암시적 커서는 커서의 이름이 없기 때문에 SQL(접두어) + % + 커서속성자 형태의 구문을 사용 한다.

명시적 커서 속성자 예제

```
EXIT WHEN CUR_EMP%NOTFOUND;
```

명시적 커서는 커서이름 + % + 커서속성자 형태의 구문을 사용 한다.

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

암시적 커서의 커서 속성자를 실습합니다.

(1) 동일한 데이터에 대한 DELETE 연산을 두번 수행하면서 변화되는 커서 속성자의 값을 관찰하시기 바랍니다.

(2) ROWCOUNT , FOUND 커서 속성자의 의미를 이해 하시기 바랍니다.

5_CURSOR_ATT_IMP.SQL

```
REM   5_CURSOR_ATT_IMP.SQL
SET SERVEROUTPUT ON
BEGIN
    //    첫번째 DELETE
     DELETE FROM EMP WHERE SAL > 2000;
     DBMS_OUTPUT.PUT_LINE('[1-DELETE]'||TO_CHAR(SQL%ROWCOUNT)||'ROWS IS DELETED');

     IF   SQL%FOUND   THEN
        DBMS_OUTPUT.PUT_LINE('SQL%FOUND = TRUE ');
     ELSE
        DBMS_OUTPUT.PUT_LINE('SQL%NOTFOUND = FALSE ');
     END IF;

     DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');

      //  두번째 DELETE : 동일한 DELETE 를 2번째 실행하므로 삭제할 데이타가 없음
      DELETE FROM EMP WHERE SAL > 2000;
      DBMS_OUTPUT.PUT_LINE('[2-DELETE]'||TO_CHAR(SQL%ROWCOUNT)||'ROWS IS DELETED');

        IF   SQL%FOUND   THEN
             DBMS_OUTPUT.PUT_LINE('SQL%FOUND = TRUE');
        ELSE
             DBMS_OUTPUT.PUT_LINE('SQL%NOTFOUND = FALSE');
        END IF;
       ROLLBACK;
END;
/
```

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

```
DECLARE
        CURSOR  CUR_EMP IS
                 SELECT EMPNO,JOB,SAL,COMM FROM EMP WHERE  DEPTNO = 10;

        V_ENAME         VARCHAR2(10);
        V_JOB           VARCHAR2(9);
        V_SAL           NUMBER(7,2);
        V_COMM          NUMBER(7,2);
BEGIN
        OPEN    CUR_EMP;
        LOOP

                FETCH  CUR_EMP  INTO  V_ENAME,V_JOB,V_SAL,V_COMM ;
                EXIT   WHEN  CUR_EMP%NOTFOUND;


                INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
                      VALUES(V_ENAME,V_JOB,V_SAL,V_COMM);

        END LOOP;

        DBMS_OUTPUT.PUT_LINE('TOTAL '||TO_CHAR(CUR_EMP%ROWCOUNT)||' rows processed');

        CLOSE   CUR_EMP;
        COMMIT;
END;
/
```

5_CURSOR_1.SQL

① 선언부(Declare Section)에서 명시적 커서의 이름과 커서를 정의

[질문] 커서 정의 부분의 조건절이 DEPTNO = 10 으로 정해져 있습니다.
10번 부서의 직원이 보너스 지급 대상자로 선정 되었지만 다음달 보너스 지급시에는 20번 부서 직원들에게 지급하는 경우가 발생할수 있습니다.
어떻게 해야 할지?

[질문] 커서 정의시 Join 연산이 가능할까?

```
CURSOR  JOIN_CUR IS
            SELECT D.DNAME,E.ENAME,E.SAL,S.GRADE
            FROM   SCOTT.EMP E,SCOTT.DEPT D,SCOTT.SALGRADE S
            WHERE   E.DEPTNO = 10 AND
                    E.DEPTNO = D.DEPTNO AND
                    E.SAL BETWEEN S.LOSAL AND S.HISAL
            ORDER BY D.DNAME,E.ENAME;
```

② 데이터 처리를 하기 위한 변수를 정의

커서(CURSOR)에서 데이터를 FETCH 할때 데이터를 저장하는 변수들,
커서에서 4개 컬럼을 정의하였는데 이번 예제에서는 4개의 변수를
정의 하였습니다.

[질문] (1) 복잡한 조인 문장으로 커서를 정의하는 컬럼이 100개인 경우라면 100개의 변수를 정의하고 FETCH 시에도 100개의 변수를 나열

해야 하는가?

(2) 테이블의 컬럼 정의가 변경되는 경우 관련된 PL/SQL BLOCK내의 변수 정의를 변경 해야 하는가?.

③ 커서 OPEN 시에는 변수가 사용된 경우 해당 변수와 바인딩을 한후 해당 SQL 을 EXECUTE 하여 SELECT 의 결과 집합을 만든다.

④ 결과집합의 레코드를 1건씩 INTO 이하의 변수에 저장합니다.
[질문] (1) 대용량 데이터인경우 LOOP를 돌면서 수많은 FECTH를 반복 해야 하는데 성능 문제 발생 가능성이 있다. 방법은?
(2) 커서를 정의하는 컬럼이 100개인 경우라면 INTO 이하에 100 개의 변수를 나열해야 한다. 방법은?

⑤ 커서 속성을 가지고 Loop 반복을 종료할 상황을 점검한후 fetch 할 데이터가 없는 경우 Loop 종료. Fetch할 데이터가 없는 경우 CUR_EMP%NOTFOUND 는 TRUE값을 RETURN 한다.

⑥ 데이터 처리 Logic

⑦ 커서를 명시적으로 종료

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5-5-2 커서와 참조변수(%TYPE)

5_CURSOR_2.SQL

```
DECLARE
        CURSOR  CUR_EMP IS
                 SELECT EMPNO,JOB,SAL,COMM FROM EMP WHERE  DEPTNO = 10;

        V_ENAME         EMP.ENAME%TYPE;
        V_JOB           EMP.JOB%TYPE;
        V_SAL           EMP.SAL%TYPE;
        V_COMM          EMP.COMM%TYPE;
BEGIN
        OPEN    CUR_EMP;
        LOOP

                FETCH  CUR_EMP  INTO  V_ENAME,V_JOB,V_SAL,V_COMM ;
                EXIT   WHEN  CUR_EMP%NOTFOUND;


                INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
                      VALUES(V_ENAME,V_JOB,V_SAL,V_COMM);

        END LOOP;

        DBMS_OUTPUT.PUT_LINE('TOTAL '||TO_CHAR(CUR_EMP%ROWCOUNT)||' rows processed');

        CLOSE   CUR_EMP;
        COMMIT;
END;
/
```

명시적 커서에 참조 속성을 사용하여 참조 변수를 적용하는 사례

선언부(Declare Section)에서 4개의 참조변수를 정의.

[질문] 개발시 편리성/유연성을 가지게 되었지만 커서내의 정의된 컬럼의갯수가 많아지면 예를 들면 50개의 컬럼이 사용된다면 50개의 변수 정의 INTO 이하에 50개의 변수를 나열해야 합니다. 방법은 ?

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5-5-3 커서와 참조변수 (%ROWTYPE)

5_CURSOR_3.SQL

```
DECLARE
        CURSOR  CUR_EMP IS
                 SELECT ENAME,JOB,SAL,COMM FROM EMP WHERE  DEPTNO = 10;

        R_CUR_EMP       CUR_EMP%ROWTYPE;
BEGIN
        OPEN    CUR_EMP;
        LOOP
                FETCH  CUR_EMP  INTO  R_CUR_EMP;

                EXIT   WHEN  CUR_EMP%NOTFOUND;

                INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
                      VALUES(R_CUR_EMP.ENAME,R_CUR_EMP.JOB,R_CUR_EMP.SAL,R_CUR_EMP.COMM);

        END LOOP;

        DBMS_OUTPUT.PUT_LINE('TOTAL '||TO_CHAR(CUR_EMP%ROWCOUNT)||' rows processed');

        CLOSE   CUR_EMP;
        COMMIT;
END;
/
```

커서에서 많은 컬럼을 사용하는 경우 변수 정의 와 사용에 많은 반복 행위로 개발 시 불편한 부분이 많았지만 ROWTYPE의 참조변수를 사용하여 개발의 편리성과 유연성의 장점을 얻을수 있다.

① 이전 단계에서는 4개의 변수를 정의 하였지만 커서를 참조하는
1개의 ROWTYPE 참조 변수를 정의하여 편리하게 되었다.

② FETCH 단계에서 INTO 이하의 변수 영역에서도 1개의 변수를
정의하게 되므로 간단 해졌죠.!!

③ ROWTYPE 참조 변수를 INSERT 구문에서 사용하는 부분을 잠시 관찰해보시기 바랍니다.

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5-5-4 Cursor for Loop

CURSOR를 자주 사용하는 경우에는 표기의 간소화 특징으로 인해 유용하다.
CURSOR 와 FOR LOOP의 결합한 기능이다.

FOR LOOP의 특징을 먼저 이해 하면 CURSOR FOR LOOP는 쉽게 이해 가능 하다.

```
FOR LOOP_IDX IN 1..100
LOOP
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(LOOP_IDX));
END LOOP;
```

① LOOP_IDX 는 for loop의 첨자 변수로 선언부(DECLARE SECTION)에서 정의하지 않고 FOR LOOP 내에서만 LOCAL변수 처럼 사용 된다.
선언부에 별도로 정의 하지 않고 FOR LOOP내에서만 LOCAL 변수 처럼 사용하면 된다는 점을 기억하시기 바랍니다.

② IN 은 LOOP 첨자변수의 범위를 나타냅니다.
위의 예제는 1 .. 100 1부터 1씩 증가하여 100에 도달하면 즉 100번 LOOP 를 수행하면 자동으로 종료를 한다.

2가지 사항을 기억 하시고 CURSOR FOR LOOP를 들여다 보죠!!

5-5-4-1 커서 FOR LOOP -1

5_CURSOR_4_1.SQL

```
DECLARE
        CURSOR  CUR_EMP IS
                SELECT ENAME,JOB,SAL,COMM FROM EMP WHERE  DEPTNO = 10;
BEGIN

        FOR  R_CUR_EMP IN CUR_EMP
        LOOP
                INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
                     VALUES(R_CUR_EMP.ENAME,R_CUR_EMP.JOB,R_CUR_EMP.SAL,R_CUR_EMP.COMM);

        END LOOP;

        --DBMS_OUTPUT.PUT_LINE('TOTAL '||TO_CHAR(CUR_EMP%ROWCOUNT)||' rows processed');

        COMMIT;
END;
/
```

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

① 선언부에서는 커서만 정의하고 커서로 데이터 처리시 사용되는 변수 미정의
[질문] 커서 정의는 이전과 동일하지만 변수를 선언하지 않은 이유?

② FOR LOOP의 2가지 특징중에

첫번째 : R_CUR_EMP 는 LOOP 첨자 변수로 자동 할당 되어 생성되는 변수
R_CUR_EMP CUR_EMP%ROWTYPE; 의 기능이 자동(암시적)으로 수행된 것.

두번째 : IN 다음에는 LOOP 첨자 변수의 데이터 범위가 나타나는데
CUR_EMP 라는 커서는 LOOP 첨자 변수가 가질수 있는 값의 범위 영역에 해당 한다.

③ OPEN ,FETCH , CLOSE 가 없다.
OPEN ,FETCH, CURSOR CLOSE를 자동(암시적)으로 수행 합니다.
커서 사용이 무척 간결 해졌다.

INSERT 구문의 VALUES절을 보시면 이전 커서 예제에서 사용되던 R_CUR_EMP.ENAME 이 그대로 사용됩니다. 자동(암시적)으로 커서 참조 ROWTYPE을 정의 했다는 것을 의미 합니다.

④ DBMS_OUTPUT 주석 처리
CURSOR FOR LOOP를 사용하게 되는 경우에는 해당 구문의 실행은 에러를 유발 합니다. 이유가 무엇일까요? 정답은 FOR LOOP 종료시 해당 커서를 자동(암시적)으로 종료 하기 때문에 LOOP를 벗어나게 되면 커서를 참조 할 수 없기 때문에 커서 속성을 사용할수 없게 됩니다. 쉬운 개념이지만 기억 해두실 필요가 있습니다. CURSOR FOR LOOP를 사용시 LOOP 밖에서 커서 속성을 사용하려면 사전에 변수에 별도로 저장하여 활용하는 방법을 사용 합니다.

5-5-4-2 커서 FOR LOOP -2

5_CURSOR_4_2.SQL

```
BEGIN

        FOR  R_CUR_EMP IN (SELECT ENAME,JOB,SAL,COMM FROM EMP WHERE  DEPTNO = 10)
        LOOP
                INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
                      VALUES(R_CUR_EMP.ENAME,R_CUR_EMP.JOB,R_CUR_EMP.SAL,R_CUR_EMP.COMM);

        END LOOP;

        --DBMS_OUTPUT.PUT_LINE('TOTAL '||TO_CHAR(CUR_EMP%ROWCOUNT)||' rows processed');

        COMMIT;
END;
/
```

SELECT * FROM BINUS;

① CURSOR FOR LOOP의 2번째 유형
커서의 정의를 선언부(DECLARE)에서 하지 않고 IN 절에서 정의하여 사용
커서 표기가 무척 간결해졌죠! 커서의 정의가 복잡한 경우에는 사용하기 불편 하여 표현의 한계가 있지만 간단한 쿼리의 커서인 경우에는 사용하기가 편리 합니다.

<참고? 프로그램 유지보수측면에서 보면 커서를 여러 개 사용하게 되면 선언부에 커서가 많아져서 프로그램을 작성하지 않은 사람이 프로그램을 파악할 때 가독성이 떨어진다. 그래서 이와 같이 간단하게 정의하여 사용 하는 경우도 있다. 물론 PL/SQL 모듈내에서 재사용되는 커서는 선언부에 선언을 하는것이 좋다

5-5-5 커서 파라미터(PARAMETER)

자! 그동안 커서의 조건절의 값을 정적으로 정해진 상수만을 사용해서 10번 부서 직원들에게만 보너스를 지급 받을 기회를 주었습니다.
유연한 프로그래밍을 해보죠~

커서를 정의할때 파라미터(PARAMETER)를 사용하여 실행시점에서 접근하는 데이터의 범위를 결정 할수 있습니다.

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5_CURSOR_5.SQL

```
DECLARE
        CURSOR  CUR_EMP(P_DEPTNO IN NUMBER) IS
                 SELECT ENAME,JOB,SAL,COMM FROM EMP WHERE  DEPTNO = P_DEPTNO;

        V_DEPTNO        DEPT.DEPTNO%TYPE;
BEGIN

        V_DEPTNO :=  20;

        FOR  R_CUR_EMP IN CUR_EMP(V_DEPTNO)
        LOOP
                INSERT  INTO BONUS(ENAME,JOB,SAL,COMM)
                      VALUES(R_CUR_EMP.ENAME,R_CUR_EMP.JOB,R_CUR_EMP.SAL,R_CUR_EMP.COMM);

        END LOOP;

        COMMIT;
END;
/
```

① 그동안의 실습과 달리 조건절에 상수 대신 동적인 변수가 할당 되었습니다.
DEPTNO = 10 ➔ DEPTNO = P_DEPTNO 으로 변경
커서의 이름 다음에 커서에게 넘겨줄 파라미터 변수를 정의 합니다.

② 변수 V_DEPTNO 에 값 20을 할당 합니다.
그동안은 커서 정의시에 상수가 지정 되었지만 커서에 파라미터를 사용하게 되므로 커서 실행전에 값을 동적으로 지정 할수 있게 되었습니다.

③ 이전 학습을 복습해보면 커서 OPEN 시점에 하는 중요한일 2가지가 있습니다.
첫번째는 BIND VARIABLE
두번째는 EXECUTE SQL
BIND는 묶는다 연결하다 라는 사전 의미를 가지고 있습니다.
위의 예제를 보면 값 20을 커서내의 P_DEPTNO 변수와 연결(BIND) 합니다.
BIND를 하고 나면 의미상으로 DEPTNO = P_DEPTNO ➔ DEPTNO = 20

구체적인 값이 지정되고 나서 해당 SQL을 실행 한다

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

[학습정리]

1. Cursor는 암시적 커서 와 명시적 커서로 나눌수 있다.
2. 명시적 커서는 다중행(Multiple Rows)을 처리 하기 위해서 사용된다.
3. 명시적 커서는 DECLARE, OPEN ,FETCH,CLOSE 4단계로 처리된다.
   DECLARE : 커서의 이름과 커서를 정의한다.
   OPEN : Bind Variables 와 Execute SQL 을 수행한다.
   FETCH : 데이터를 변수에 저장한다.
   CLOSE : 커서를 종료하고 자원을 반납한다.
4. Cursor for loop는 암시적 OPEN, 암시적 FETCH , 암시적 CLOSE를 통해 편리하게 Cursor를 제어할수 있다.

[심화학습] CURSOR FOR UPDATE

Long Transaction 이라고 가정 하기 위해 INSERT 실행후 5초를 WAIT 하도록 Oracle 제공 Package인 DBMS_LOCK을 사용 한다.

SYS 계정에서 DBMS_LOCK 을 실행할수 있는 권한을 부여 해야 사용할수 있다.

```
SQL> CONN SYS AS SYSDBA
SQL> GRANT EXECUTE ON DBMS_LOCK TO SCOTT;
```

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

5_CURSOR_for_update.SQL

```
REM   5_CURSOR_forupdate.SQL
SET   TIMING   ON
SELECT   TO_CHAR(SYSDATE,'HH24:MI:SS') AS START_TIME FROM DUAL;

DECLARE
        CURSOR   CUR_EMP(P_DEPTNO IN NUMBER) IS
                SELECT EMPNO,ENAME,JOB,SAL,COMM FROM EMP
                WHERE   DEPTNO = P_DEPTNO
                 FOR   UPDATE;
--               FOR   UPDATE   WAIT   10;
--               FOR    UPDATE   NOWAIT;
--                FOR   UPDATE   OF EMP.JOB
        V_DEPTNO        DEPT.DEPTNO%TYPE;
BEGIN
        V_DEPTNO   :=   20;
        FOR   R_CUR_EMP IN CUR_EMP(V_DEPTNO)
        LOOP
                UPDATE EMP
                SET      COMM =   ROUND(R_CUR_EMP.SAL * 0.05,0)
                WHERE   EMPNO = R_CUR_EMP.EMPNO;
              -- WHERE   CURRENT OF CUR_EMP;

                INSERT INTO BONUS(ENAME,JOB,SAL,COMM)
                VALUES(R_CUR_EMP.ENAME,R_CUR_EMP.JOB,R_CUR_EMP.SAL,
                        ROUND(R_CUR_EMP.SAL * 0.05,0));

                 DBMS_LOCK.SLEEP(5);
        END LOOP;
        COMMIT;
END;
/

SELECT   TO_CHAR(SYSDATE,'HH24:MI:SS') AS END_TIME FROM DUAL;
```

데이터베이스 프로그래밍 – PL/SQL Cursor (5차시)

① update(수정) 하기 위해 해당 데이타를 select 하는 것이니 다른 세션(session)에서 해당 데이타를 select 만 하고 update(수정) 하기 위해 건드리지 말라는 의미 입니다. 다른 세션이 건드리지 못하게 select 한 해당 데이타에 row level lock을 생성 합니다. select 시에도 배타적인 락(exclusive Lock)을 생성하게 되므로 사용에 주의가 필요.
약간 어려운 이야기 이지만 select는 공유락(shared lock)을 사용합니다.
조회하는 기능 이므로 최대한의 동시성을 위해서 입니다.

② cursor 정의에서 select 한 대상 데이타를
for loop 안에서 update(수정)을 하는 내용 입니다.
커서에서 FETCH된 해당 ROW를 수정하게 되는 경우에는 CURRENT절이
효율적 입니다,
만약 사원번호에 PK가 생성되어 있는 경우에는 Index Scan을 통해 해당 데이타에 접근을 하게 됩니다, 결론적으로 인덱스 스캔보다 WHERE CURRENT OF 커서명을 사용하는것이 데이타 접근에 더 효율적입니다,
이유는 CURRENT 절은 ROWID를 통해 해당 데이타에 직접 접근하는 방식 입니다,
Index Scan 을 하는 이유도 실상은 ROWID를 얻기 위함입니다,
쉽게 생각 하시면 인덱스 스캔은 2차에 걸쳐서 접근 하는것을 CURRENT절은
1차에 접근하여 1 단계가 줄어들기 때문입니다,

③ 수정하는 작업이 오래 걸리는(Long running Transaction) 트랜잭션이라는 상황을 만들기 위해 dbms_lock.sleep 을 사용하여 update시 마다 5초를 sleep 하게 하였습니다.
20번 부서에 5명이 근무하면 최소 25초 이상은 걸리는 트랜잭션이 됩니다.

실습 시나리오는
1. FOR UPDATE 를 2개의 세션에서 동시에 수행
2. FOR UPDATE WAIT 5 를 2개의 세션에서 동시에 수행
3. FOR UPDATE NOWAIT 를 2개의 세션에서 동시에 수행