<< Less 07 명시적 커서 사용 >>

중요 학습 내용:

- 암시적 커서 및 명시적 커서 구분
- 명시적 커서를 사용하는 이유 설명
- 명시적 커서 선언 및 제어
- 간단한 루프 및 커서 FOR 루프를 사용하여 데이터 패치(fetch)
- 파라미터가 포함된 커서 선언 및 사용
- FOR UPDATE 절을 사용하여 행 잠금
- WHERE CURRENT OF 절을 사용하여 현재 행 참조

• 본 문서의 실습은 별도의 언급이 없는 한, SQL*Developer를 이용하여
hr 계정으로 데이터베이스에 원격하여 수행합니다.

<< 커서(Cursors) 개요 >>

- 커서는 오라클-서버에 의해서 할당된
  private memory area(개별적인 메모리 영역)에 대한 포인터입니다.

  오라클-서버에 의해 실행되는 모든 SQL문장은 관련된
  개별 Cursor를 가집니다.
- 커서는 변수가 아닙니다.
- 오라클-서버에서 실행되는 모든 SQL 문에는 연관된 개별 커서가
  있습니다.

- 암시적 커서(Implicit Cursor)
  : 모든 DML 및 SELECT 문에 대해 데이터베이스 서버에
  의해서 선언하고 관리됩니다(4장 문서 참조).

- 명시적 커서(Explicit Cursor)
  : 프로그래머가 선언하고 관리합니다.

<< 명시적 커서 작업 >>

- 명시적 커서를 사용하여 [여러 행을 반환하는 SELECT]문에 의해
  반환되는 각 row를 개별적으로 처리할 수 있습니다.

  여러 행 질의에 의해 반환되는 행집합을 "활성집합(active set)"이라고
  하며 활성 집합의 크기는 검색조건을 만족하는 행 수와 같습니다.

  PL/SQL 프로그램은 커서를 OPEN하여, 질의에 의해 반환되는 행을
  처리한 후, 커서를 닫습니다(CLOSE).
  커서는 활성 집합에서 현재 위치를 표시합니다.

<< 명시적 커서의 기능 >>

• 질의에 의해 반환된 행을 첫번째 행부터 차례로 처리할 수 있습니다.

• 현재 처리중인 행을 추적합니다.

• 프로그래머가 PL/SQL 블록에서 명시적 커서를
  수동으로 제어할 수 있습니다.

==========
<< 명시적 커서 제어 >>
==========

• DECLARE : 커서를 선언(명명된 SQL 영역 생성)
  | • 커서의 이름을 지정하고
  | 커서 내에서 수행될 질의구조(SELECT문)를 정의.
  |
• OPEN : 커서를 엽니다(활성 집합 식별).
  | • 커서에 선언된 질의를 실행하여
  | 참조되는 모든 변수를 바인드 합니다.
  | 질의에 의해 식별된 행들을 활성집합이라고 합니다.
  | 이제 인출(FETCH)이 가능합니다.
  |
• FETCH : 현재 행을 해당 변수에 로드.
  |
  EMPTY? : 기존 행 테스트(행이 있으면 FETCH로 돌아감)
  | • 행을 인출할 대마다 남은 행이 있는지 커서를 테스트하고
  | 처리할 행이 더 이상 없으면 커서를 닫습니다.
  |
• CLOSE : 활성 집합 해제
  • 커서를 다시 열어 새로운 활성집합을 생성할 수 있습니다.

<< 명시적 커서 사용과정 >>

1. 커서를 선언

2. 커서를 열기(OPEN)

3. 행을 인출(FETCH)
   [여러 행일 때 반복됨]

4. 커서를 닫기(CLOSE)

<< 1. 커서 선언: PL/SQL 선언부에 명시 >>

[기본 문법]

```
CURSOR cursor_name
IS
  select_statement ;
```

[문법 설명을 위한 예제]

```
DECLARE

      locid NUMBER:= 1700;

      CURSOR emp_cursor
      IS
         SELECT employee_id, last_name
         FROM hr.employees
         WHERE department_id =30 ;

      CURSOR dept_cursor
```

```
IS
   SELECT *
   FROM hr.departments
   WHERE location_id = locid ;
...
```

(주의) 커서 선언 시 사용되는 SELECT 문에는
INTO 절을 적으면 안됩니다.

특정 순서로 행을 처리해야 한다면
질의에 ORDER BY 절을 사용합니다.

커서는 조인, 서브쿼리 등을 포함하는
유효한 ANSI SELECT 문일 수 있습니다.

<< 2. 커서 열기: PL/SQL 실행부에 명시 >>

[기본 문법]

```
OPEN cursor_name ;
```

[문법 설명을 위한 예제]

```
OPEN emp_cursor ;

OPEN dept_cursor ;
```

<< Cursor 열기(Open) 시에 수행되는 작업 >>

1. 중요한 처리 정보 및 관련 내용 영역(context area)을 위한
   메모리를 동적으로 할당합니다.

2. SELECT 문을 분석(PARSING)합니다.

3. 입력 변수를 바인드 합니다.
   즉, 해당 메모리 주소를 확보하여 입력 변수의 값을 설정합니다.

4. SELECT 문을 실행하여 활성 집합 즉, 검색조건을 만족하는 행 집합을
   식별합니다.

   (참고) 활성집합의 행은 OPEN문에서 변수에 저장하는 것이 아니라
   FETCH문에서 저장하게 됩니다.

5. 포인터의 위치를 활성 집합의 첫번째 행 앞으로 지정만 합니다.

(참고)

- 커서 선언 시에 SELECT 문장에 FOR UPDATE 절이 사용된 경우에는
  해당 행을 잠급니다.
- 커서가 열려 있을 때 질의에 의해 반환되는 행이 없더라도
  예외가 발생하지 않습니다.
- 그러나, 커서명%ROWCOUNT 커서 속성(Cursor Attribute)을 이용하여,
  인출(FETCH) 후, 커서의의 상태를 테스트 할 수 있습니다.

<< 3. 커서에서 데이터 패치(fetch): PL/SQL 실행부에 명시 >>

[기본 문법]

```
FETCH cursor_name INTO [ 변수1, 변수2,...
                       | record_name    ];
```

- 현재 행 값을 변수로 가져옵니다.
- 포인터를 식별된 집합 내의 다음 행으로 이동합니다.
- INTO 절에 [동일한 개수의 변수] 또는 [하나의 레코드]를 명시합니다.
- 동일한 개수의 변수를 INTO에 명시할 때
  해당 열의 위치와 순서가 동일해야 합니다.

  또는 커서에 대한 레코드를 정의하여
  FETCH INTO 절에서 해당 레코드를 참조합니다.
- 커서의 행 포함 여부를 테스트 합니다.

<< 4. 커서 닫음(CLOSE;) PL/SQL 실행부에 명시 >>

- 항상 PL/SQL 명시적 커서를 사용한 경우, CURSOR의 모든 레코드를 처리 후,
  사용한 커서를 닫아줍니다.

- 커서들 닫으면, 세션에 할당된 메모리(PGA)에, 명시적 커서가 사용했던
  메모리 영역이 사용 해제되고, 커서가 삭제됩니다.

<< 명시적 커서 사용 실습:1 >>

```
SQL> SET SERVEROUTPUT ON
SQL>

SQL> DECLARE
        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name
           FROM hr.employees
           WHERE department_id =30;

           empno hr.employees.employee_id%TYPE;

           lname hr.employees.last_name%TYPE;

     BEGIN
           OPEN emp_cursor;
           LOOP
                FETCH emp_cursor INTO empno, lname;

                EXIT WHEN emp_cursor%NOTFOUND ;
                DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
           END LOOP;

        -- CLOSE emp_cursor ;
     END;
     /
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
119 Colmenares
```

```
PL/SQL procedure successfully completed.

SQL>



<< 명시적 커서 사용 실습:2 >>

SQL> SET SERVEROUTPUT ON
SQL>

SQL> DECLARE
          v_empno  hr.employees.employee_id%type;
          v_ename  hr.employees.last_name%type;

          CURSOR emp_cursor
          IS
             SELECT employee_id, last_name
             FROM hr.employees ;  --107개 레코드
     BEGIN
          OPEN emp_cursor;

          FOR i IN 1..10
          LOOP
                FETCH emp_cursor INTO v_empno, v_ename;  --FETCH 문
                DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)||' '||v_ename);
          END LOOP ;  --처음부터 10개 레코드만 처리됩니다.
          CLOSE emp_cursor; -- 남아있는 모든 레코드는 사라집니다.
     END ;
     /
174 Abel
166 Ande
130 Atkinson
105 Austin
204 Baer
116 Baida
167 Banda
172 Bates
192 Bell
151 Bernstein

PL/SQL procedure successfully completed.

SQL>  --명시적 커서에 담긴 레코드 개수만큼 처리를 수행하도록
      --반복 횟수를 지정하는 것을 권장합니다.



<< 명시적 커서 사용 실습:3 >>

SQL> SET SERVEROUTPUT ON
SQL>
SQL> DECLARE
        v_empno  hr.employees.employee_id%type;

        v_ename  hr.employees.last_name%type;

        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name
           FROM hr.employees
           ORDER BY 1 ;  -- 정렬됨
     BEGIN
        OPEN emp_cursor;
        FOR i IN 1..10
        LOOP
```

```
            FETCH emp_cursor INTO v_empno, v_ename;
            DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)||' '||v_ename);
        END LOOP ;

        DBMS_OUTPUT.PUT_LINE('---------------');
        DBMS_OUTPUT.PUT_LINE(emp_cursor%ROWCOUNT);
        DBMS_OUTPUT.PUT_LINE('---------------');

        LOOP
          FETCH emp_cursor INTO v_empno, v_ename;
          EXIT WHEN emp_cursor%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE( v_empno ||' '||v_ename);
        END LOOP;

          DBMS_OUTPUT.PUT_LINE('===============');
          DBMS_OUTPUT.PUT_LINE(emp_cursor%ROWCOUNT);
          DBMS_OUTPUT.PUT_LINE('---------------');
        CLOSE emp_cursor;
    END ;
    /
100 King
101 Kochhar
102 De Haan
103 Hunold
104 Ernst
105 Austin
106 Pataballa
107 Lorentz
108 Greenberg
109 Faviet
---------------
10
---------------
110 Chen
111 Sciarra
112 Urman
113 Popp
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
119 Colmenares
120 Weiss
121 Fripp
122 Kaufling
123 Vollman
124 Mourgos
125 Nayer
126 Mikkilineni
127 Landry
128 Markle
129 Bissot
130 Atkinson
131 Marlow
132 Olson
133 Mallin
134 Rogers
135 Gee
136 Philtanker
137 Ladwig
138 Stiles
139 Seo
140 Patel
141 Rajs
142 Davies
143 Matos
144 Vargas
```

```
145 Russell
146 Partners
147 Errazuriz
148 Cambrault
149 Zlotkey
150 Tucker
151 Bernstein
152 Hall
153 Olsen
154 Cambrault
155 Tuvault
156 King
157 Sully
158 McEwen
159 Smith
160 Doran
161 Sewall
162 Vishney
163 Greene
164 Marvins
165 Lee
166 Ande
167 Banda
168 Ozer
169 Bloom
170 Fox
171 Smith
172 Bates
173 Kumar
174 Abel
175 Hutton
176 Taylor
177 Livingston
178 Grant
179 Johnson
180 Taylor
181 Fleaur
182 Sullivan
183 Geoni
184 Sarchand
185 Bull
186 Dellinger
187 Cabrio
188 Chung
189 Dilly
190 Gates
191 Perkins
192 Bell
193 Everett
194 McCain
195 Jones
196 Walsh
197 Feeney
198 OConnell
199 Grant
200 Whalen
201 Hartstein
202 Fay
203 Mavris
204 Baer
205 Higgins
206 Gietz
==============
107
--------------

PL/SQL procedure successfully completed.
```

```
SQL>
```

(참고) 하나의 PL/SQL BLOCK 구조 내에서
같은 이름의 커서를 2번 OPEN할 수는 없습니다.

```
SQL> DECLARE
        v_empno  hr.employees.employee_id%type;

        v_ename  hr.employees.last_name%type;

        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name
           FROM hr.employees
           ORDER BY 1 ;  -- 정렬됨
     BEGIN
        OPEN emp_cursor;
        FOR i IN 1..10
        LOOP
                FETCH emp_cursor INTO v_empno, v_ename;
                DBMS_OUTPUT.PUT_LINE (TO_CHAR(v_empno)||' '||v_ename);
        END LOOP ;

        DBMS_OUTPUT.PUT_LINE('---------------');
        DBMS_OUTPUT.PUT_LINE(emp_cursor%ROWCOUNT);
        DBMS_OUTPUT.PUT_LINE('---------------');

        OPEN emp_cursor;
        LOOP
          FETCH emp_cursor INTO v_empno, v_ename;
          EXIT WHEN emp_cursor%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE( v_empno ||' '||v_ename);
        END LOOP;

          DBMS_OUTPUT.PUT_LINE('===============');
          DBMS_OUTPUT.PUT_LINE(emp_cursor%ROWCOUNT);
          DBMS_OUTPUT.PUT_LINE('---------------');
        CLOSE emp_cursor;
     END;
     /
DECLARE
*
ERROR at line 1:
ORA-06511: PL/SQL: cursor already open
ORA-06512: at line 8
ORA-06512: at line 23


SQL>
```

<< 4. 커서 닫기: PL/SQL 실행부에 명시 >>

[기본 문법]

```
CLOSE cursor_name ;
```

- 행 처리를 완료한 후 반드시 커서를 닫습니다.
- 사용하던 커서를 닫은 후에 필요한 경우
  커서를 다시 열 수 있습니다.

(주의) 닫혀진 커서에서 데이터 인출(FETCH)을
시도하지 마십시오.
인출을 시도하면 INVALID_CURSOR 예외가 발생합니다.

커서를 닫지 않고 PL/SQL 블록을 종료할 수 있지만
자원을 해제시키기 위해서는 선언한 커서를
명시적으로 닫는 것이 좋습니다.

<< OPEN_CURSORS 초기화 파라미터 >>

각 세션에 대해 열리는 최대 커서 수를 제한하는데 사용됨.

디폴트 값은 OPEN_CURSORS=50

<< 커서 및 레코드 활용 실습 >>

- PL/SQL 레코드로 값을 인출(FETCH)하여
결과행 집합의 행을 처리합니다.

1> 실습용 테이블 생성

```
SQL> CREATE TABLE hr.temp_list (empid, empname)
     AS
        SELECT employee_id, last_name
        FROM hr.employees
        WHERE 1=2 ;

Table created.

SQL>
```

2> 커서와 레코드를 포함하는 PL/SQL 익명블록 작성 및 실행

```
SQL> DECLARE
        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name
           FROM hr.employees
           WHERE department_id = 30 ;

        emp_record emp_cursor%ROWTYPE ;

     BEGIN
        OPEN emp_cursor;
        LOOP
                FETCH emp_cursor INTO emp_record ;
                EXIT WHEN emp_cursor%NOTFOUND ;
                --INSERT INTO hr.temp_list (empid, empname)
                --VALUES (emp_record.employee_id, emp_record.last_name) ;

                INSERT INTO hr.temp_list VALUES emp_record ;

        END LOOP ;
```

```
        COMMIT   ;
        CLOSE emp_cursor ;
    END ;
    /
PL/SQL procedure successfully completed.

SQL>
```

3> 처리된 내용 확인

```
SQL> SELECT *
    FROM hr.temp_list ;

    EMPID EMPNAME
--------- ------------------------------
       114 Raphaely
       115 Khoo
       116 Baida
       117 Tobias
       118 Himuro
       119 Colmenares

6 rows selected.

SQL>
```

<< CURSOR-FOR-LOOP 문 개요 >> 매우 중요합니다.

[기본 문법]

```
FOR record_name IN cursor_name
LOOP
        statement1;
        statement2;
        . . .
END LOOP;

record_name : 레코드 이름
cursor_name : 커서 이름
statement1  : 실행문1
statement2  : 실행문1
```

- CURSOR-FOR-LOOP 문]을 사용하면
  명시적 커서를 바로 처리 할 수 있습니다.
- 레코드가 암시적으로 선언됩니다.
- 자동으로 커서의 열기, 인출, 종료 및
  닫기가 수행됩니다.

(지침)

- 루프를 제어하는 레코드는 암시적으로 선언되므로
  선언하지 마세요.
- 필요한 경우 루프 도중에 커서 속성을 테스트 합니다.
- 필요한 경우 FOR 문에서 커서 이름 뒤에
  커서의 파라미터를 괄호로 묶어 제공할 수 있습니다.
- 커서 작업을 명시적으로 처리해야 하는 경우에는

커서 FOR 루프를 사용하지 마십시오.

<< CURSOR-FOR-LOOP 문 예제 >>

```
SQL> SET SERVEROUTPUT ON
SQL>


SQL> SELECT * FROM hr.temp_list ;

     EMPID EMPNAME
---------- ------------------------------
       114 Raphaely
       115 Khoo
       116 Baida
       117 Tobias
       118 Himuro
       119 Colmenares

6 rows selected.

SQL>



SQL> DECLARE
        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name
           FROM hr.employees
           WHERE department_id =30 ;

     BEGIN
        FOR emp_record IN emp_cursor
        LOOP
              EXIT WHEN emp_cursor%ROWCOUNT > 3 ;
                 DBMS_OUTPUT.PUT_LINE
                 (emp_record.employee_id||' ' ||emp_record.last_name);
        END LOOP;
        -- 위의 CURSOR-FOR-LOOP 실행을 통해서
        -- 작업된 내용은 모두 인출되었고 커서가 닫혔습니다.

        -- 아래의 CURSOR-FOR-LOOP 실행 중에
        -- 다시 커서가 열리고, 로직 실행 후, 다시 커서가 닫힙니다.

        FOR emp_record IN emp_cursor
        LOOP
                DBMS_OUTPUT.PUT_LINE
                (emp_record.employee_id  ||' ' ||emp_record.last_name);
                INSERT INTO hr.temp_list VALUES emp_record ;
        END LOOP;
        COMMIT   ;
     END;
     /
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
119 Colmenares
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
```

```
119 Colmenares

PL/SQL procedure successfully completed.

SQL>

-- 실행 후, hr.temp_list 입력된 데이터 확인

SQL> set pagesize 5000

SQL> SELECT * FROM hr.temp_list ;

     EMPID EMPNAME
---------- ------------------------------
       114 Raphaely
       115 Khoo
       116 Baida
       117 Tobias
       118 Himuro
       119 Colmenares
       114 Raphaely
       115 Khoo
       116 Baida
       117 Tobias
       118 Himuro
       119 Colmenares

12 rows selected.

SQL>
```

(참고) 위의 CURSOR-FOR-LOOP 문을 기본-LOOP로 바꿔서
코드를 작성하면 아래처럼 CURSOR 처리와 관련된 내용을
모두 명시해 주어야 합니다.

```
        DECLARE
                CURSOR emp_cursor
                IS
                  SELECT employee_id, last_name
                  FROM hr.employees
                  WHERE department_id =30;

                emp_record emp_cursor%ROWTYPE ; -- 레코드 선언 코드
BEGIN
        OPEN emp_cursor; -- 커서 OPEN 코드
        LOOP
                FETCH emp_cursor INTO emp_record ; -- 커서 FETCH 코드
                EXIT WHEN emp_cursor%NOTFOUND;

                DBMS_OUTPUT.PUT_LINE( empno ||' '||lname);
        END LOOP;

        CLOSE emp_cursor ; -- 커서 CLOSE 코드
END;
/
```

<< 명시적 커서 속성(Attribute) >>

```
cursor_name%ISOPEN : 커서가 열려 있으면 TRUE로 평가합니다.
                     커서가 닫혀 있으면 FALSE로 평가합니다.
-------------------------------------------------------------------
cursor_name%NOTFOUND : 가장 최근에 인출에서 행을 반환하지 않으면
                       TRUE로 평가 합니다.
```

cursor_name%FOUND	: 가장 최근의 인출에서 행을 반환하면 TRUE로 평가합니다.
cursor_name%ROWCOUNT	: 지금까지 반환된 전체 행 수를 평가 합니다. - 행의 정확한 개수를 검색할 때 사용. - 숫자 FOR 루프를 사용하여 행을 인출할 때 사용. - 단순 루프를 사용하여 행을 인출하고 루프 종료시점을 결정할 때 사용.

<< %ISOPEN Attribute >>

- 커서가 열려 있는 경우에만 행을 인출합니다.
- %ISOPEN 커서 속성은 인출 작업을 수행하지 전에 커서가 열려 있는지 테스트 하는데 사용합니다,

```
IF NOT emp_cursor%ISOPEN THEN
        OPEN emp_cursor;
END IF;
LOOP
        FETCH emp_cursor...
```

<< %ROWCOUNT와 %NOTFOUND 사용예제 >>

- %ROWCOUNT 커서 속성을 사용하여 인출된 행 수를 정확히 검색합니다.
  커서나 커서 변수가 열려만 있으면 %ROWCOUNT는 0 입니다.
  즉, 첫번째 인출 작업을 수행하기 전에 %ROWCOUNT의 결과는 0 입니다.
- %NOTFOUND 커서 속성을 사용하여 루프 종료 시기를 결정합니다.
- LOOP의 EXIT 조건을 작성할 때 아래의 EXIT문을 사용하는 것을 권장합니다.

```
EXIT WHEN 커서이름%NOTFOUND or 커서이름%NOTFOUND IS NULL ;
```

(예제)

```
SQL> SET SERVEROUTPUT ON
SQL>
SQL> DECLARE
        empno hr.employees.employee_id%TYPE;
        ename hr.employees.last_name%TYPE;
        CURSOR emp_cursor
        IS
          SELECT employee_id, last_name
          FROM hr.employees ;
     BEGIN
        OPEN emp_cursor;
        LOOP
          FETCH emp_cursor INTO empno, ename;
          EXIT WHEN emp_cursor%ROWCOUNT > 10 OR
                    emp_cursor%NOTFOUND ;
          DBMS_OUTPUT.PUT_LINE(TO_CHAR(empno)||' '|| ename);
        END LOOP;
        CLOSE emp_cursor;
     END ;
     /
174 Abel
166 Ande
```

```
130 Atkinson
105 Austin
204 Baer
116 Baida
167 Banda
172 Bates
192 Bell
151 Bernstein

PL/SQL procedure successfully completed.

SQL>


-- 위의 기본-LOOP로 작성된 코드를 CURSOR-FOR-LOOP로
   작성하면 아래와 같습니다.

SQL> DECLARE
        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name
           FROM hr.employees;
     BEGIN
        FOR emp_record IN emp_cursor
        LOOP
          EXIT WHEN emp_cursor%ROWCOUNT > 5 ;
          DBMS_OUTPUT.PUT_LINE(TO_CHAR(emp_record.employee_id)
                                          ||' '|| emp_record.last_name);
        END LOOP;

     END ;
     /
174 Abel
166 Ande
130 Atkinson
105 Austin
204 Baer

PL/SQL procedure successfully completed.

SQL>



-- 사번 순서로 처음 5명의 업무이력을 검색

SQL> SET SERVEROUT ON
SQL>

SQL> DECLARE
        v_employee_id  hr.employees.employee_id%TYPE;
        v_job_id       hr.employees.job_id%TYPE;
        v_start_date   DATE ;
        v_end_date     DATE ;

        CURSOR emp_cursor
        IS
           SELECT employee_id, job_id, start_date, end_date
           FROM hr.job_history
           ORDER BY 1 ;
     BEGIN
        OPEN emp_cursor;
        LOOP
               FETCH emp_cursor INTO v_employee_id
                                    ,v_job_id
                                    ,v_start_date
                                    ,v_end_date   ;
```

```
                DBMS_OUTPUT.PUT_LINE
                ('Employee #: '||v_employee_id
                 ||' held the job of ' || v_job_id ||' from '
                 ||v_start_date || ' to ' || v_end_date
                );

                EXIT WHEN emp_cursor%ROWCOUNT > 4 OR
                          emp_cursor%NOTFOUND ;
        END LOOP ;
        CLOSE emp_cursor ;
     END ;
     /
Employee #: 101 held the job of AC_ACCOUNT from 21-SEP-97 to 27-OCT-01
Employee #: 101 held the job of AC_MGR from 28-OCT-01 to 15-MAR-05
Employee #: 102 held the job of IT_PROG from 13-JAN-01 to 24-JUL-06
Employee #: 114 held the job of ST_CLERK from 24-MAR-06 to 31-DEC-07
Employee #: 122 held the job of ST_CLERK from 01-JAN-07 to 31-DEC-07

PL/SQL procedure successfully completed.

SQL>



<< subquery를 사용하는 커서 FOR 루프 >>

        - cursor를 선언할 필요가 없습니다.


SQL> SET SERVEROUTPUT ON
SQL>

SQL> BEGIN
        FOR emp_record IN (SELECT employee_id, last_name
                           FROM hr.employees
                           WHERE department_id =30)
        LOOP
                --EXIT WHEN emp_cursor%ROWCOUNT > 3 ; 이 코드를 사용못함
                DBMS_OUTPUT.PUT_LINE
                ( emp_record.employee_id ||' '||emp_record.last_name);
        END LOOP;
     END;
     /
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
119 Colmenares

PL/SQL procedure successfully completed.

SQL>


-- 위의 코드를 커서를 이용한 커서-FOR-루프 구문으로 변경하면
   아래와 같습니다.

SQL> SET SERVEROUT ON
SQL>
SQL> DECLARE
        CURSOR emp_cursor IS
                SELECT employee_id, last_name
                FROM hr.employees
                WHERE department_id =30;
     BEGIN
        FOR emp_record IN emp_cursor
        LOOP
```

```
                EXIT WHEN emp_cursor%ROWCOUNT > 3 ;
                DBMS_OUTPUT.PUT_LINE
                ( emp_record.employee_id||' ' ||emp_record.last_name);

        END LOOP;
     END;
     /
114 Raphaely
115 Khoo
116 Baida

PL/SQL procedure successfully completed.

SQL>
```

<< 파라미터가 포함된 커서 >>

[기본 문법]

```
CURSOR cursor_name [(parameter_name datatype, ...)]
IS
   select_statement;
```

- 커서가 열리고 질의가 실행되면
  커서에 파리미터 값을 전달합니다.

- 매번 다른 활성집합을 사용하여
  명시적 커서를 여러번 엽니다.

```
   OPEN cursor_name(parameter_value,.....) ;
```

커서 FOR 루프에 있는 커서에 파라미터를 전달할 수 있습니다.
즉, 블록에서 명시적 커서를 여러번 열고 닫으면서 매번 다른
활성집합을 반환할 수 있습니다.
실행될 때마다 이전 커서는 닫히고 새 파라미터 집합을 사용해 커서가
다시 열립니다.

커서 선언의 각 형식 매개변수는 반드시 OPEN 문의 실제 매개변수와
대응되어야 합니다.

이러한 파라미터 데이터 유형은 스칼라 변수의 데이터 유형과 동일하지만
크기는 지정되지 않습니다.

파라미터의 이름은 커서의 질의 표현식에서 해당 파라미터를 참조할 때
사용됩니다.

파라미터 표기법은 동일한 커서를 반복적으로 참조할 때 특히 유용합니다.

```
SQL> SET SERVEROUT ON
SQL>
SQL> DECLARE
        CURSOR emp_cursor (cp_deptno NUMBER) --NUMBER 에서는 최대허용
        IS                                   --자리수를 명시하면 않됩니다.
           SELECT employee_id, last_name, department_id
           FROM hr.employees
           WHERE department_id = cp_deptno ;

        empid NUMBER;

        lname VARCHAR2(30);
```

```
   v_dno NUMBER ;

BEGIN

   OPEN emp_cursor (10);    --OPEN 시에만 커서이름(값) 사용
   LOOP                     --FETCH, CLOSE 시에는 (값) 명시 않함
           FETCH emp_cursor INTO empid, lname, v_dno;
           EXIT WHEN emp_cursor%NOTFOUND ;
           DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                 '  last_name: '||lname||
                                 '  deptno :'||v_dno);
   END LOOP ;
   CLOSE emp_cursor;

   OPEN emp_cursor (20);
   LOOP
           FETCH emp_cursor INTO empid, lname, v_dno;
           EXIT WHEN emp_cursor%NOTFOUND ;
           DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                 '  last_name: '||lname||
                                 '  deptno :'||v_dno);
   END LOOP ;
   CLOSE emp_cursor;

   OPEN emp_cursor (30);
   LOOP
           FETCH emp_cursor INTO empid, lname, v_dno;
           EXIT WHEN emp_cursor%NOTFOUND ;
           DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                 '  last_name: '||lname||
                                 '  deptno :'||v_dno);
   END LOOP ;
   CLOSE emp_cursor;

   OPEN emp_cursor (40);
   LOOP
           FETCH emp_cursor INTO empid, lname, v_dno;
           EXIT WHEN emp_cursor%NOTFOUND ;
           DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                 '  last_name: '||lname||
                                 '  deptno :'||v_dno);
   END LOOP ;
   CLOSE emp_cursor;

   OPEN emp_cursor (50);
   LOOP
           FETCH emp_cursor INTO empid, lname, v_dno;
           EXIT WHEN emp_cursor%NOTFOUND ;
           DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                 '  last_name: '||lname||
                                 '  deptno :'||v_dno);
   END LOOP ;
   CLOSE emp_cursor;

   OPEN emp_cursor (60);
   LOOP
           FETCH emp_cursor INTO empid, lname, v_dno;
           EXIT WHEN emp_cursor%NOTFOUND ;
           DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                 '  last_name: '||lname||
                                 '  deptno :'||v_dno);
   END LOOP ;
   CLOSE emp_cursor;
END ;
/
empno  : 200  last_name: Whalen  deptno :10
empno  : 201  last_name: Hartstein  deptno :20
```

```
empno  : 202  last_name: Fay  deptno :20
empno  : 114  last_name: Raphaely  deptno :30
empno  : 115  last_name: Khoo  deptno :30
empno  : 116  last_name: Baida  deptno :30
empno  : 117  last_name: Tobias  deptno :30
empno  : 118  last_name: Himuro  deptno :30
empno  : 119  last_name: Colmenares  deptno :30
empno  : 203  last_name: Mavris  deptno :40
empno  : 198  last_name: OConnell  deptno :50
empno  : 199  last_name: Grant  deptno :50
empno  : 120  last_name: Weiss  deptno :50
empno  : 121  last_name: Fripp  deptno :50
empno  : 122  last_name: Kaufling  deptno :50
empno  : 123  last_name: Vollman  deptno :50
empno  : 124  last_name: Mourgos  deptno :50
empno  : 125  last_name: Nayer  deptno :50
empno  : 126  last_name: Mikkilineni  deptno :50
empno  : 127  last_name: Landry  deptno :50
empno  : 128  last_name: Markle  deptno :50
empno  : 129  last_name: Bissot  deptno :50
empno  : 130  last_name: Atkinson  deptno :50
empno  : 131  last_name: Marlow  deptno :50
empno  : 132  last_name: Olson  deptno :50
empno  : 133  last_name: Mallin  deptno :50
empno  : 134  last_name: Rogers  deptno :50
empno  : 135  last_name: Gee  deptno :50
empno  : 136  last_name: Philtanker  deptno :50
empno  : 137  last_name: Ladwig  deptno :50
empno  : 138  last_name: Stiles  deptno :50
empno  : 139  last_name: Seo  deptno :50
empno  : 140  last_name: Patel  deptno :50
empno  : 141  last_name: Rajs  deptno :50
empno  : 142  last_name: Davies  deptno :50
empno  : 143  last_name: Matos  deptno :50
empno  : 144  last_name: Vargas  deptno :50
empno  : 180  last_name: Taylor  deptno :50
empno  : 181  last_name: Fleaur  deptno :50
empno  : 182  last_name: Sullivan  deptno :50
empno  : 183  last_name: Geoni  deptno :50
empno  : 184  last_name: Sarchand  deptno :50
empno  : 185  last_name: Bull  deptno :50
empno  : 186  last_name: Dellinger  deptno :50
empno  : 187  last_name: Cabrio  deptno :50
empno  : 188  last_name: Chung  deptno :50
empno  : 189  last_name: Dilly  deptno :50
empno  : 190  last_name: Gates  deptno :50
empno  : 191  last_name: Perkins  deptno :50
empno  : 192  last_name: Bell  deptno :50
empno  : 193  last_name: Everett  deptno :50
empno  : 194  last_name: McCain  deptno :50
empno  : 195  last_name: Jones  deptno :50
empno  : 196  last_name: Walsh  deptno :50
empno  : 197  last_name: Feeney  deptno :50
empno  : 103  last_name: Hunold  deptno :60
empno  : 104  last_name: Ernst  deptno :60
empno  : 105  last_name: Austin  deptno :60
empno  : 106  last_name: Pataballa  deptno :60
empno  : 107  last_name: Lorentz  deptno :60

PL/SQL procedure successfully completed.

SQL>


-- 다음은 위의 코드의 기본 LOOP를 CURSOR FOR LOOP 문으로 변경하여
-- 작성한 코드입니다.
```

```
SQL> DECLARE
        CURSOR emp_cursor (cp_deptno NUMBER) --NUMBER 에서는 최대허용
        IS                                   --자리수를 명시하면 않됩니다.
           SELECT employee_id, last_name, department_id
           FROM hr.employees
           WHERE department_id = cp_deptno ;

     BEGIN
        FOR emp_rec IN emp_cursor(10)
        LOOP
          EXIT WHEN emp_cursor%NOTFOUND ;
          DBMS_OUTPUT.PUT_LINE ('empno  : '||emp_rec.employee_id||
                                '  last_name: '||emp_rec.last_name||
                                '  deptno :'||emp_rec.department_id);
        END LOOP;

        FOR emp_rec IN emp_cursor(20)
        LOOP
          EXIT WHEN emp_cursor%NOTFOUND ;
          DBMS_OUTPUT.PUT_LINE ('empno  : '||emp_rec.employee_id||
                                '  last_name: '||emp_rec.last_name||
                                '  deptno :'||emp_rec.department_id);
        END LOOP;


        FOR emp_rec IN emp_cursor(30)
        LOOP
          EXIT WHEN emp_cursor%NOTFOUND ;
          DBMS_OUTPUT.PUT_LINE ('empno  : '||emp_rec.employee_id||
                                '  last_name: '||emp_rec.last_name||
                                '  deptno :'||emp_rec.department_id);
        END LOOP;

        FOR emp_rec IN emp_cursor(40)
        LOOP
          EXIT WHEN emp_cursor%NOTFOUND ;
          DBMS_OUTPUT.PUT_LINE ('empno  : '||emp_rec.employee_id||
                                '  last_name: '||emp_rec.last_name||
                                '  deptno :'||emp_rec.department_id);
        END LOOP;

        FOR emp_rec IN emp_cursor(50)
        LOOP
          EXIT WHEN emp_cursor%NOTFOUND ;
          DBMS_OUTPUT.PUT_LINE ('empno  : '||emp_rec.employee_id||
                                '  last_name: '||emp_rec.last_name||
                                '  deptno :'||emp_rec.department_id);
        END LOOP;

        FOR emp_rec IN emp_cursor(60)
        LOOP
          EXIT WHEN emp_cursor%NOTFOUND ;
          DBMS_OUTPUT.PUT_LINE ('empno  : '||emp_rec.employee_id||
                                '  last_name: '||emp_rec.last_name||
                                '  deptno :'||emp_rec.department_id);
        END LOOP;

     END ;
     /



-- 아래의 코드는 "파라미터가 있는 커서" 대신
-- SQL*Developer/SQL*Plus의 치환변수를 이용하여
-- 코드를 실행할 때마다, 값을 지정해서 서로 다른 커서의 활성집합을
-- 이용하도록 작성된 코드입니다.

SQL> SET SERVEROUT ON
```

```
SQL>
SQL> SET VERIFY OFF
SQL>
SQL> DECLARE
        v_deptno NUMBER(3) := &dept_no ;
        empid    NUMBER;
        lname    VARCHAR2(30);
        v_dno    NUMBER ;

        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name,department_id
           FROM hr.employees
           WHERE department_id = v_deptno;

     BEGIN
        OPEN emp_cursor;
        LOOP
                FETCH emp_cursor INTO empid ,lname,v_dno;
                EXIT WHEN emp_cursor%NOTFOUND ;
                DBMS_OUTPUT.PUT_LINE
                (   'empno  : '||empid||'  last_name: '
                  |||lname||'  deptno :'||v_dno
                );
        END LOOP ;
        CLOSE emp_cursor;
     END ;
     /
Enter value for dept_no: 10
empno  : 200  last_name: Whalen  deptno :10

PL/SQL procedure successfully completed.

SQL>
SQL> /
Enter value for dept_no: 20
empno  : 201  last_name: Hartstein  deptno :20
empno  : 202  last_name: Fay  deptno :20

PL/SQL procedure successfully completed.

SQL>
SQL> /
Enter value for dept_no: 30
empno  : 114  last_name: Raphaely  deptno :30
empno  : 115  last_name: Khoo  deptno :30
empno  : 116  last_name: Baida  deptno :30
empno  : 117  last_name: Tobias  deptno :30
empno  : 118  last_name: Himuro  deptno :30
empno  : 119  last_name: Colmenares  deptno :30

PL/SQL procedure successfully completed.

SQL>




-- 파라미터를 이용하는 커서는 실행 로직 중에서 값을 변경하면서
-- 하나의 실행로직 안에서 동일한 커서의
-- 활성집합의 내용이 바뀌어야 할 때(단, 컬럼구성은 동일,
-- 서로 다른 행에서 가져온 값들) 사용합니다.

SQL> DECLARE
        v_deptno   NUMBER(4) ;
        empid      NUMBER;
        lname      VARCHAR2(30);
        v_dno      NUMBER ;
```

```
        CURSOR emp_cursor (cp_deptno NUMBER)
        IS
           SELECT employee_id, last_name, department_id
           FROM hr.employees
           WHERE department_id = cp_deptno;

     BEGIN
        DBMS_OUTPUT.PUT_LINE ('========================================') ;
        v_deptno := 30 ;

        OPEN emp_cursor (v_deptno);
        LOOP
                FETCH emp_cursor INTO empid ,lname,v_dno;
                EXIT WHEN emp_cursor%NOTFOUND ;
                DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                      '  last_name: '||lname||
                                      '  deptno :'||v_dno);
        END LOOP ;
        DBMS_OUTPUT.PUT_LINE ('----------------------------------------') ;
        CLOSE emp_cursor;

        v_deptno := 100 ;

        OPEN emp_cursor (v_deptno);
        LOOP
                FETCH emp_cursor INTO empid ,lname,v_dno;
                EXIT WHEN emp_cursor%NOTFOUND ;
                DBMS_OUTPUT.PUT_LINE ('empno  : '||empid||
                                      '  last_name: '||lname||
                                      '  deptno :'||v_dno);
        END LOOP ;
        DBMS_OUTPUT.PUT_LINE ('----------------------------------------') ;
        CLOSE emp_cursor;
     END ;
     /
========================================
empno  : 114  last_name: Raphaely  deptno :30
empno  : 115  last_name: Khoo  deptno :30
empno  : 116  last_name: Baida  deptno :30
empno  : 117  last_name: Tobias  deptno :30
empno  : 118  last_name: Himuro  deptno :30
empno  : 119  last_name: Colmenares  deptno :30
----------------------------------------
empno  : 108  last_name: Greenberg  deptno :100
empno  : 109  last_name: Faviet  deptno :100
empno  : 110  last_name: Chen  deptno :100
empno  : 111  last_name: Sciarra  deptno :100
empno  : 112  last_name: Urman  deptno :100
empno  : 113  last_name: Popp  deptno :100
----------------------------------------

PL/SQL procedure successfully completed.

SQL>
```

<< FOR UPDATE 절 >>

[기본 문법]

```
        SELECT ...
        FROM ...
        WHERE ....
        FOR UPDATE [OF column_reference][NOWAIT | WAIT n] ;
```

여기서 column_reference: 질의가 수행되는 테이블의 열이며
열 목록을 사용할 수도 있습니다.

- 커서 질의에 FOR UPDATE 절을 추가하면
  해당 커서를 열때 영향을 받는 행을 잠글 수 있습니다.

- 명시적 잠금을 사용하여 트랜잭션 기간 동안
  다른 세션의 액세스를 거부할 수 있습니다.

- 오라클 서버는 트랜잭션 종료시에 잠금을 해제하므로
  FOR UPDATE 를 사용할 경우에는
  명시적 커서에서의 인출 작업사이에 커밋을 수행하지 마십시오.

- 여러 테이블을 질의할 경우 FOR UPDATE 절을 사용하여
  행 잠금을 특정 테이블로만 제한할 수 있습니다.

  이 때 FOR UPDATE 절이 해당 테이블의 열을
  참조하는 경우에만 테이블 행이 잠깁니다.

  즉, FOR UPDATE col_name(s)은 col_name(s)이 들어있는 테이블에서만
  행을 잠급니다.

  SELECT.... FOR UPDATE 문은 갱신 또는 삭제될 행을
  식별한 다음 결과집합에 있는 각 행을 잠급니다.

  행의 기존 값을 기반으로 갱신을 수행하려는 경우에는
  갱신전에 다른 사용자가 해당 행을 변경하는 것을 방지해야 하므로
  이 기능을 사용하면 유용합니다.

```
SQL> SET SERVEROUT ON
SQL>
SQL> DECLARE
        CURSOR emp_cursor
        IS
           SELECT a.employee_id, a.last_name, b.department_name
           FROM hr.employees2 a INNER JOIN hr.departments2 b
           ON (a.department_id = b.department_id)
           WHERE  a.department_id = 80
           FOR UPDATE OF a.salary NOWAIT
           ORDER by 2 ;


           /*
              ========================================================
              (설명)

              FOR UPDATE : SELECT-문장으로 액세스 되는 행에 락을 건다.
              OF a.salary: salary 컬럼이 정의된 테이블에만 락을 겁니다.

              NOWAIT : 커서가 사용하는 행에 다른세션에서
                       락을 먼저 걸었으면 에러을 내고
                       커서 실행을 종료(디폴트는  WAIT)
              ========================================================
           */

     BEGIN
        FOR emp_record IN emp_cursor
        LOOP
        --      EXIT WHEN emp_cursor%ROWCOUNT > 3 ;
                DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
```

```
                                  ||'  ' ||emp_record.last_name
                                  ||'  ' ||emp_record.department_name);
      END LOOP;
      COMMIT ;  -- FOR UPDATE 옵션으로 액세스되는 행에 걸린 락-해제
   END;
   /
174  Abel  Sales
166  Ande  Sales
167  Banda  Sales
172  Bates  Sales
151  Bernstein  Sales
169  Bloom  Sales
148  Cambrault  Sales
154  Cambrault  Sales
160  Doran  Sales
147  Errazuriz  Sales
170  Fox  Sales
163  Greene  Sales
152  Hall  Sales
175  Hutton  Sales
179  Johnson  Sales
156  King  Sales
173  Kumar  Sales
165  Lee  Sales
177  Livingston  Sales
164  Marvins  Sales
158  McEwen  Sales
153  Olsen  Sales
168  Ozer  Sales
146  Partners  Sales
145  Russell  Sales
161  Sewall  Sales
159  Smith  Sales
171  Smith  Sales
157  Sully  Sales
176  Taylor  Sales
150  Tucker  Sales
155  Tuvault  Sales
162  Vishney  Sales
149  Zlotkey  Sales

PL/SQL procedure successfully completed.

SQL>
```

<< WHERE CURRENT OF 절 개요 >>

[기본 문법]

```
WHERE CURRENT OF cursor_name
```

- 커서를 사용하여 현재 행을 갱신 및 삭제합니다.
- 커서 질의에 FOR UPDATE 절을 명시해서 행에 LOCK을 걸어야합니다.
- WHERE CURRENT OF 절을 사용하여 명시적 커서에서 현재 행을 참조합니다.
- ROWID를 이용하지 않고도 현재 커서에서 참조하는 행을 갱신하고 삭제할 수 있게 합니다.
- 추가적으로 FETCH문에 의해 가장 최근에 처리된 행을

참조하기 위해서 "WHERE CURRENT OF 커서이름 "
절로 DELETE나 UPDATE문 작성이 가능합니다..

(예제1)

```
SQL> COL last_name format a30
     COL job_id format a15

     SELECT employee_id, last_name, job_id
     FROM hr.employees2
     WHERE department_id = 30 ;

EMPLOYEE_ID LAST_NAME                      JOB_ID
----------- ------------------------------ ---------------
        114 Raphaely                       PU_MAN
        115 Khoo                           PU_CLERK
        116 Baida                          PU_CLERK
        117 Tobias                         PU_CLERK
        118 Himuro                         PU_CLERK
        119 Colmenares                     PU_CLERK

6 rows selected.

SQL>
```

-- 데이터베이스의 테이블에 저장된 사번 114 인 직원정보는
   다음의 값들로 구성된 하나의 행입니다.

   114 "Den Raphaely" DRAPHEAL 515.127.4561 07-DEC-94 PU_MAN 11000 100 30

```
SQL> SET SERVEROUTPUT ON ;
SQL>

SQL> DECLARE
        CURSOR emp_cursor
        IS
           SELECT employee_id, last_name
           FROM hr.employees2
           WHERE  department_id = 30
           FOR UPDATE;
     BEGIN
        FOR emp_record IN emp_cursor
        LOOP
                DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
                                    ||'   ' ||emp_record.last_name);

                UPDATE hr.employees2
                SET job_id = 'ST_CLERK'
                WHERE CURRENT OF emp_cursor;

                DBMS_OUTPUT.PUT_LINE('Updated successfully');

            --  DELETE FROM hr.employees2
            --  WHERE CURRENT OF emp_cursor;

            --  DBMS_OUTPUT.PUT_LINE('Deleted successfully');
        END LOOP;
             -- COMMIT ;

     EXCEPTION
               WHEN OTHERS THEN
                 DBMS_OUTPUT.PUT_LINE('ERR MESSAGE : ' || SQLERRM);
     END;
     /
114  Raphaely
```

```
Updated successfully
115  Khoo
Updated successfully
116  Baida
Updated successfully
117  Tobias
Updated successfully
118  Himuro
Updated successfully
119  Colmenares
Updated successfully

PL/SQL procedure successfully completed.

SQL>


-- PL/SQL 실행 후, 테이블의 UPDATE된 컬럼값 확인

SQL> SELECT employee_id, last_name, job_id
     FROM hr.employees2
     WHERE  department_id = 30 ;

EMPLOYEE_ID LAST_NAME                 JOB_ID
----------- ------------------------- ---------------
        114 Raphaely                  ST_CLERK
        115 Khoo                      ST_CLERK
        116 Baida                     ST_CLERK
        117 Tobias                    ST_CLERK
        118 Himuro                    ST_CLERK
        119 Colmenares                ST_CLERK

6 rows selected.

SQL>

        (정리) UPDATE 문의 [WHERE CURRENT OF 커서이름] 절은
               현재 FETCH된 레코드를 참조합니다.


-- 이 후 실습을 위하여 트랜잭션을 롤백시킵니다.

SQL> ROLLBACK ;

Rollback complete.

SQL>



<< WHERE CURRENT OF 실습 예제2 >>

        - 다음의 목적에 해당하는 PL/SQL 코드를 작성하시오.

          실습을 위하여 hr.employees2, hr.departments2  테이블을
          사용하시오.

          부서코드 60에 속한 사원에 대해 루프를 수행하여 급여가
          5000보다 적은지를 검사한 후에,
          급여가 5000보다 적으면 10% 인상.


SQL> SELECT employee_id, last_name, salary
     FROM hr.employees2
     WHERE  department_id = 60 ;
```

```
EMPLOYEE_ID LAST_NAME                      SALARY
----------- ------------------------------ ----------
        103 Hunold                           9000
        104 Ernst                            6000
        105 Austin                           4800
        106 Pataballa                        4800
        107 Lorentz                          4200

SQL>



SQL> DECLARE
        CURSOR sal_cursor
        IS
           SELECT e.department_id, e.employee_id, e.last_name, e.salary
           FROM hr.employees2 e INNER JOIN hr.departments2 d
                                ON(d.department_id = e.department_id)
           WHERE d.department_id = 60
           FOR UPDATE OF e.salary NOWAIT ;
     BEGIN
        FOR emp_record IN sal_cursor
        LOOP
                DBMS_OUTPUT.PUT_LINE('----------------------------------------');
                DBMS_OUTPUT.PUT_LINE('Before Update');
                IF emp_record.salary < 5000
                THEN
--                      UPDATE hr.employees2
--                      SET salary = emp_record.salary *  1.10
--                      WHERE CURRENT OF sal_cursor ;
--              DBMS_OUTPUT.PUT_LINE('after Update');
                       DELETE FROM hr.employees2
                       WHERE CURRENT OF emp_cursor;
                END IF ;
        END LOOP ;
     END ;
     /
----------------------------------------
Before Update
----------------------------------------
Before Update
----------------------------------------
Before Update
after Update
----------------------------------------
Before Update
after Update
----------------------------------------
Before Update
after Update

PL/SQL procedure successfully completed.

SQL>


-- PL/SQL 실행 후, 테이블의 UPDATE된 컬럼값 확인

SQL> SELECT employee_id, last_name, salary
     FROM hr.employees2
     WHERE  department_id = 60 ;

EMPLOYEE_ID LAST_NAME                      SALARY
----------- ------------------------------ ----------
        103 Hunold                           9000
        104 Ernst                            6000
        105 Austin                           5280
```

```
        106 Pataballa                              5280
        107 Lorentz                                4620

SQL>

-- 이 후 실습을 위하여 hr.employees2 에 대한 변경사항을 롤백합니다.

SQL> ROLLBACK ;

Rollback complete.

SQL>
```

(참고) 위에서 처럼
DELETE 문 또는 UPDATE 문에
[WHERE CURRENT OF 커서이름]절을 명시하여
커서에 의하여 가장 최근에 패치된 레코드와
관련된 행을 변경할 수 있습니다.

<< [WHERE CURRENT OF 커서이름]절 사용 시 주의사항 >>

조인 문장이 사용된 커서에서,
아래처럼 커서 선언 시에 명시된 SELECT 문장에
FOR UPDATE 옵션을 기술할 때,
(1) FOR UPDATE 만 명시하거나
(2) 두 테이블의 컬럼을 모두 명시한 경우,

[WHERE CURRENT OF 커서이름]절이 명시된 UPDATE/DELETE 문이
실행되지 않고, 해당 행에 락만 걸립니다.

따라서, 조인SELECT문 커서를 이용하여 테이블의 행을 수정/삭제
하는 경우, 커서의 SELECT에 명시된 FOR UPDATE 에 반드시
of 컬럼명 옵션을 추가하여,
"UPDATE/DELETE 되는 하나의 테이블을 지정"해야 합니다.

<< 테스트 1 >>

```
-- 익명블록 실행 전 hr.employees2 테이블의 관련 데이터 확인

SQL> SELECT employee_id, last_name, salary
     FROM hr.employees2
     WHERE  department_id = 60 ;

EMPLOYEE_ID LAST_NAME                      SALARY
----------- ------------------------- ----------
        103 Hunold                          9000
        104 Ernst                           6000
        105 Austin                          4800
        106 Pataballa                       4800
        107 Lorentz                         4200

SQL>
```

-- 커서로 액세스 되는 행의 SALARY를 UPDATE 하기 위한
익명블록을 작성하여 실행합니다.

이 때 커서를 정의하는 문장에서 FOR UPDATE 만 사용했습니다.

```
SQL> DECLARE
        CURSOR sal_cursor
        IS
           SELECT e.department_id, e.employee_id, e.last_name, e.salary
```

```
        FROM hr.employees2 e INNER JOIN hr.departments2 d
                             ON(d.department_id = e.department_id)
        WHERE d.department_id = 60
        FOR UPDATE NOWAIT ;
    BEGIN
       FOR emp_record IN sal_cursor
       LOOP
              IF emp_record.salary < 5000
              THEN
                     UPDATE hr.employees2
                     SET salary = emp_record.salary *  1.10
                     WHERE CURRENT OF sal_cursor ;
              END IF ;
       END LOOP ;
    END ;
    /

PL/SQL procedure successfully completed.

SQL>


-- PL/SQL 실행 후, 테이블의 UPDATE된 컬럼값 확인

SQL> SELECT employee_id, last_name, salary
     FROM hr.employees2
     WHERE  department_id = 60 ;

EMPLOYEE_ID LAST_NAME                      SALARY
----------- ------------------------- ----------
        103 Hunold                          9000
        104 Ernst                           6000
        105 Austin                          4800
        106 Pataballa                       4800
        107 Lorentz                         4200

SQL>

        ==> 관련 행에 대한 UPDATE가 수행되지 않았습니다.


-- 행에 걸린 락을 해제하기 위하여 트랜잭션을 롤백합니다.

SQL> ROLLBACK ;

Rollback complete.

SQL>



<< 테스트 2 >>

-- 커서로 액세스 되는 행을 삭제하기 위한
   익명블록을 작성하여 실행합니다.
   이 때 커서를 정의하는 문장에서  FOR UPDATE 만 사용했습니다.

SQL> DECLARE
        CURSOR sal_cursor
        IS
           SELECT e.department_id, e.employee_id, e.last_name, e.salary
           FROM hr.employees2 e INNER JOIN hr.departments2 d
                                ON(d.department_id = e.department_id)
           WHERE d.department_id = 60
           FOR UPDATE ;
    BEGIN
       FOR emp_record IN sal_cursor
```

```
        LOOP
                IF emp_record.salary < 5000
                THEN
                        DELETE FROM hr.employees2
                        WHERE CURRENT OF sal_cursor ;
                END IF ;
        END LOOP ;
     END ;
     /

PL/SQL procedure successfully completed.

SQL>


-- PL/SQL 실행 후, 테이블의 DELETE된 행 확인

SQL> SELECT employee_id, last_name, salary
     FROM hr.employees2
     WHERE  department_id = 60 ;

EMPLOYEE_ID LAST_NAME                 SALARY
----------- ------------------------- ----------
        103 Hunold                      9000
        104 Ernst                       6000
        105 Austin                      4800
        106 Pataballa                   4800
        107 Lorentz                     4200

SQL>

        ==> 행이 삭제되지 않았습니다.


-- 행에 걸린 락을 해제하기 위하여 트랜잭션을 롤백합니다.

SQL> ROLLBACK ;

Rollback complete.

SQL>


<< 테스트 3 >>

-- 익명블록 실행 전 hr.employees2 테이블의 관련 데이터 확인

SQL> SELECT employee_id, last_name, salary
     FROM hr.employees2
     WHERE  department_id = 60 ;

EMPLOYEE_ID LAST_NAME                 SALARY
----------- ------------------------- ----------
        103 Hunold                      9000
        104 Ernst                       6000
        105 Austin                      4800
        106 Pataballa                   4800
        107 Lorentz                     4200

SQL>


-- 커서로 액세스 되는 행을 삭제하기 위한
   익명블록을 작성하여 실행합니다.
   이 때 커서를 정의하는 문장에서  FOR UPDATE 에 [OF e.salary]를
   명시했습니다.
```

```
SQL> DECLARE
        CURSOR sal_cursor
        IS
           SELECT e.department_id, e.employee_id, e.last_name, e.salary
           FROM hr.employees2 e INNER JOIN hr.departments2 d
                              ON(d.department_id = e.department_id)
           WHERE d.department_id = 60
           FOR UPDATE OF e.salary ;
     BEGIN
        FOR emp_record IN sal_cursor
        LOOP
                IF emp_record.salary < 5000
                THEN
                        DELETE FROM hr.employees2
                        WHERE CURRENT OF sal_cursor ;
                END IF ;
        END LOOP ;
     END ;
     /

PL/SQL procedure successfully completed.

SQL>


-- PL/SQL 실행 후, 테이블의 DELETE된 행 확인

SQL> SELECT employee_id, last_name, salary
     FROM hr.employees2
     WHERE  department_id = 60 ;

EMPLOYEE_ID LAST_NAME                      SALARY
----------- ------------------------------ ----------
        103 Hunold                           9000
        104 Ernst                            6000

SQL>

        --> 5000 보다 작은 SALARY를 가지는
            행이 삭제되었습니다.

-- 이 후 실습을 위하여 hr.employees2의 변경작업을 롤백합니다.

SQL> ROLLBACK ;

Rollback complete.

SQL>
```

<< 서브쿼리가 포함된 커서 >>

```
        -- 커서를 선언하는 SELECT 문에
           아래처럼 서브쿼리를 사용할 수 있습니다.
```

예제

```
SQL> SET SERVEROUT ON
SQL>

SQL> DECLARE
        CURSOR my_cursor
        IS
```

```
        SELECT  t1.department_id, t1.department_name, t2.staff
        FROM hr.departments t1
              INNER JOIN (SELECT department_id, COUNT(*) AS STAFF
                          FROM hr.employees
                          GROUP BY department_id) t2
                       ON (t1.department_id = t2.department_id)
        WHERE t2.staff >= 3 ;
    BEGIN
       FOR emp_record IN my_cursor
       LOOP
       --      EXIT WHEN my_cursor%ROWCOUNT > 3 ;
               DBMS_OUTPUT.PUT_LINE( emp_record.department_id
                                    ||'  ' ||emp_record.department_name
                                    ||'  ' ||emp_record.staff);
       END LOOP;
       COMMIT ;
    END;
    /
60  IT  5
90  Executive  3
30  Purchasing  6
50  Shipping  45
80  Sales  34
100  Finance  6

PL/SQL procedure successfully completed.

SQL>
```