**1. 커서의 개념**

ORACLE SERVER은 SQL문장을 실행하 위하여 Private SQL Area이라 불리는 작업 영역을 사용합니다. Private SQL Area에 이름을 붙이고 저장된 정보를 액세스하기 위해 PL/SQL CURSOR를 사용한다. 블록의 실행 부분이 SQL문장을 실행할 때 PL/SQL은 SQL식별자를 가지는 암시적 CURSOR를 생성하고 자동적으로 이 CURSOR를 관리합니다. 명시적 CURSOR는 명시적으로 선언되고 프로그래머에 의해 명명됩니다.

**1.1 CURSOR의 종류**

**1.1.1 암시적 CURSOR**

ORACLE SERVER은 명시적으로 선언된 CURSOR와 관련 없는 각 SQL문장을 수행하기 위해 CURSOR를 암시적으로 생성하여 사용한다. PL/SQL은 SQL CURSOR로써 가장 최근의 암시적 CURSOR를 참조할 수 있도록 해 줍니다. SQL CURSOR를 제어하기 위해 OPEN,FETCH,CLOSE를 사용할 수 없지만 가장 최근에 실행된 SQL문장에 대한 정보를 얻기 위한 CURSOR속성을 사용할 수 있다.(SQL%ROWCOUNT,SQL%FOUND,SQL%NOTFOUND,SQL%ISOPEN등)

**1.1.2 명시적 CURSOR**

다중 행 SELECT 문장에 의해 RETURN되는 각 행을 개별적으로 처리하기 위해 명시적 CURSOR를 사용합니다. 다중 행 질의에 의해 RETURN된 행의 집합은 result set이라 불립니다. 그것의 크기는 검색 조건에 일치하는 행의 수입니다.

**가) 명시적 CURSOR의 함수**

1. 질의에 의해 RETURN된 첫번째 행부터 행 하나씩 처리할 수 있다.
2. 현재 처리되는 행의 트랙을 유지 합니다.
3. 프로그래머가 PL/SQL 블록에서 수동으로 제어할 수 있습니다.

♣ 참고

암시적 CURSOR에 대한 인출(FETCH)은 배열 인출(FETCH)이며, 두번째 행의 존재는 여전히 TOO\_MANY\_ROWS 예외가 발생합니다. 그러므로 다중 인출을 수행하고 작업 영역에서 구문 분석된 질의를 재실행하기 위해 명시적 CURSOR를 사용할 수 있습니다.

**1.2 명시적 CURSOR의 제어**

명시적 CURSOR를 사용하기 위해서는 4가지 단계를 거처야 한다.

1. 수행되기 위한 질의의 구조를 정의하고 이름을 지정함으로써 CURSOR를 선언한다.
2. CURSOR를OPEN한다. OPEN문장은 질의를 실행하고 참조되는 임의의 변수를 바인드 합니다. 질의에 의해 식별된 행을 active set이라 불리고 인출(FETCH) 가능합니다.
3. CURSOR에서 데이터를 인출(FETCH)합니다. FETCH문장은 CURSOR에서 변수로 현재 행을 로드합니다. 각 인출(FETCH)은 활성 셋(active set)에서 다음 행으로 그 포인터를 이동하도록 합니다.
4. CURSOR를 CLOSE합니다. CLOSE 문장은 행의 할성 셋(active set)을 해제 합니다. 이제 새로운 할성 셋(active set)을 생성하기 위해 CURSOR를 다시 OPEN할 수 있습니다.

NO

YES

DECLARE OPEN FETCH EMPTY? CLOSE

* active set을 release
* 현재 행을 READ하여 변수에 전달
* active set을 지정
* 이름있는SQL영역 생성
* 처리할 행이 있는가를 검사
* 처리할 행 있다면 FETCH를 반복

[그림 20-1] EXPLICIT CURSOR의 흐름도

**1.3 DECLARE CURSOR**

명시적으로 CURSOR를 선언하기 위해 CURSOR문장을 사용한다. 질의 내에서 변수를 참조할 수 있지만 CURSOR 문장 전에 선언되어야 한다.

**1.3.1 Syntax**

|  |
| --- |
| **CURSOR** cursor\_name **IS**  select\_statement; |

cursor\_name PL/SQL 식별자

select\_statement INTO절이 없는 SELECT 문장

**1.4 OPEN CURSOR**

질의를 수행하고 검색 조건을 충족하는 모든 행으로 구성된 결과 셋을 생성하기 위해 CURSOR를 OPEN한다. CURSOR는 이제 결과 셋에서 첫번째 행을 가리킴니다.

**1.4.1 Syntax**

|  |
| --- |
| **OPEN**  cursor\_name; |

**나) OPEN문장은 다음의 작업을 수행한다.**

1. 문맥 영역에 대해 동적으로 메모리를 할당하여 중요한 프로세싱 정보를 포함 합니다.
2. SELECT문장을 구문 분석합니다.
3. 입력 변수를 바인드 합니다.
4. 결과 셋을 식별합니다. 즉 검색 조건을 충족시키는 행의 집합입니다. OPEN문장이 실행될 때 결과 셋에 있는 행을 변수로 읽어 들이지 않습니다. 그대신 FETCH문장이 행을 읽습니다.
5. 포인터는 활성 셋에서 첫번째 행에 위치합니다.

**1.5 FETCH CURSOR**

FETCH 문장은 결과 셋에서 하나의 행을 읽어 들입니다. 각 인출(FETCH) 후에 CURSOR는 결과 셋에서 다음 행으로 이동한다.

**1.5.1 Syntax**

|  |
| --- |
| **FETCH**  cursor\_name **INTO** {variable1[,variable2, . . . .] | record\_name}; |

☞ Guidelines

1. SELECT 문장의 열과 같은 개수의 변수를 FETCH문장의 INTO절에 포함시켜 좌측부터 1대1 대응 되도록 데이형과 길이가 같아야 합니다.
2. CURSOR에 대한 레코드를 정의하고 FETCH INTO절에서 레코드를 참조할 수 있습니다.
3. CURSOR가 RETURN할 행을 포함하는지 테스트합니다. FETCH시 아무 값도 읽지 않아도 즉 활성 셋에서 프로세스할 남겨진 행이 없는 경우에도 오류가 발생되지 않습니다.
4. FETCH문장은 다음 작업을 수행합니다.
5. 활성 셋에서 다음 행으로 포인터를 이동합니다.
6. 현재 행에 대한 정보를 출력 PL/SQL변수로 읽어 들입니다.
7. 포인터가 활성 셋의 끝에 위치하게 되면 CURSOR는 FOR LOOP를 탈출 합니다.

**1.6 CLOSE CURSOR**

CLOSE문장은 CURSOR를 사용할 수 없게 하고 결과 셋의 정의를 해제합니다. SELECT 문장이 다 처리된 완성 후에는 CURSOR를 닫습니다. 필요하다면 CURSOR를 다시 열수도 있습니다. 그러므로 활성 셋을 여러 번 설정할 수 있다.

**1.6.1 Syntax**

|  |
| --- |
| **CLOSE** cursor\_name; |

☞ Guidelines

CLOSE문장은 context area를 해제 합니다. 커서를 닫지 않고 PL/SQL블록을 종료하는 것이 가능하다 할 지라도 리소스를 다시 사용 가능하게 하기 위해 명시적으로 선언된 임의의 커서를 닫는 습관을 들여야 합니다. 데이터베이스 매개변수(initial parameter file)에서 OPEN\_CURSORS매개변수에 의해 결정되는 사용자마다 해당하는 커서의 수에는 최대 한계가 있습니다. 디폴트로 OPEN\_CURSORS=50입니다.

**1.7 명시적 CURSOR의 속성**

명시적 CURSOR로 CURSOR에대해 상태 정보를 얻기 위한 4가지 속성이 있습니다.

|  |  |  |
| --- | --- | --- |
| 속 성 | 타 입 | 설 명 |
| %ISOPEN | BOOLEAN | CURSOR가 열리면 TRUE |
| %NOTFOUND | BOOLEAN | 가장 최근의 인출(FETCH)이 행을 RETURN하지 않으면 TRUE |
| %FOUND | BOOLEAN | 가장 최근의 인출(FETCH)이 행을 RETURN하면 TRUE |
| %ROWCOUNT | NUMBER | 지금까지 RETURN된 행의 총 수 |

**1.8 복수 인출(FETCH) 제어**

명시적 CURSOR에서 여러 행을 처리하기 위해서 반복적으로 인출(FETCH)을 수행하는 루프를 정의합니다. 결과적으로 활성 셋의 모든 행은 처리되고 인출(FETCH)이 실패하면 %NOTFOUND속성을 TRUE로 설정한다. CURSOR에 대해 참조하기 전에 각 인출(FETCH)의 성공을 테스트하기 위해 명시적 CURSOR를 사용합니다.

문제1) 부서번호를 입력받아 사원번호,이름,급여를 출력하는 SCRIPT를 작성하여라.

|  |
| --- |
| SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_deptno PROMPT ' 부서번호를 입력하시오 : '  DECLARE  v\_deptno emp.deptno%TYPE := &p\_deptno;  v\_empno emp.empno%TYPE;  v\_ename emp.ename%TYPE;  v\_sal emp.sal%TYPE;  v\_sal\_total NUMBER(10,2) := 0;  CURSOR emp\_cursor IS  SELECT empno,ename,sal  FROM emp  WHERE deptno = v\_deptno  ORDER BY empno;  BEGIN  OPEN emp\_cursor;  DBMS\_OUTPUT.PUT\_LINE('사번 이 름 급 여');  DBMS\_OUTPUT.PUT\_LINE('---- ---------- ----------------');  LOOP  FETCH emp\_cursor INTO v\_empno,v\_ename,v\_sal;  EXIT WHEN emp\_cursor%NOTFOUND;  v\_sal\_total := v\_sal\_total + v\_sal;  DBMS\_OUTPUT.PUT\_LINE(RPAD(v\_empno,6) ||  RPAD(v\_ename,12) || LPAD(TO\_CHAR(v\_sal,'$99,999,990.00'),16));  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('----------------------------------');  DBMS\_OUTPUT.PUT\_LINE(RPAD(TO\_CHAR(v\_deptno),2) || '번 부서의 합 ' ||  LPAD(TO\_CHAR(v\_sal\_total,'$99,999,990.00'),16));  CLOSE emp\_cursor;  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.9 CURSOR와 RECORD**

테이블에서 열의 구조를 사용하기 위해 RECORD를 정의할 수 있다. 또한 명시적 CURSOR에서의 열 목록을 기초로 하여 RECORD를 정의할 수 있습니다. 이것은 단순히 인출할 수 있기 때문에 활성 셋의 행을 처리하기가 편리하다. 그러므로 행 값은 RECORD의 해당 필드 안으로 직접 LOAD된다.

문제2) DEPT 테이블의 내용을 조회하는 SCRIPT을 작성하여라. 단 %ROWTYPE을 사용하여라.

|  |
| --- |
| SET SERVEROUTPUT ON  DECLARE  dept\_record dept%ROWTYPE;  CURSOR dept\_cursor IS  SELECT \*  FROM dept  ORDER BY deptno;  BEGIN  OPEN dept\_cursor;  DBMS\_OUTPUT.PUT\_LINE('부서번호 부 서 명 위 치');  DBMS\_OUTPUT.PUT\_LINE('-------- ------------- ------------');  LOOP  FETCH dept\_cursor INTO dept\_record;  EXIT WHEN dept\_cursor%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(LPAD(dept\_record.deptno,2) || ' '  || RPAD(dept\_record.dname,15) || RPAD(dept\_record.loc,12));  END LOOP;  CLOSE dept\_cursor;  END;  /  SET SERVEROUTPUT OFF |

문제3) DEPT 테이블의 내용을 조회하는 SCRIPT을 작성하여라. 단 RECORD TYPE을 선언하여 사용하여라.

|  |
| --- |
| SET SERVEROUTPUT ON  DECLARE  TYPE dept\_record\_type IS RECORD  (v\_deptno dept.deptno%TYPE,  v\_dname dept.dname%TYPE,  v\_loc dept.loc%TYPE);  dept\_record dept\_record\_type;  CURSOR dept\_cursor IS  SELECT \*  FROM dept  ORDER BY deptno;  BEGIN  OPEN dept\_cursor;  DBMS\_OUTPUT.PUT\_LINE('부서번호 부 서 명 위 치');  DBMS\_OUTPUT.PUT\_LINE('-------- ------------- ------------');  LOOP  FETCH dept\_cursor INTO dept\_record;  EXIT WHEN dept\_cursor%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(LPAD(dept\_record.v\_deptno,2) || ' '  || RPAD(dept\_record.v\_dname,15) ||  RPAD(dept\_record.v\_loc,12));  END LOOP;  CLOSE dept\_cursor;  END;  /  SET SERVEROUTPUT OFF |

**1.10 CURSOR와 FOR LOOP**

CURSOR FOR LOOP는 명시적 CURSOR에서 행을 처리합니다. LOOP에서 각 반복마다 CURSOR를 열고 행을 인출(FETCH)하고 모든 행이 처리되면 자동으로 CURSOR가 CLOSE되므로 사용하기가 편리합니다.

**1.10.1 Syntax**

|  |
| --- |
| **FOR** record\_name **IN**  cursor\_name **LOOP**  statement1;  statement2;  . . . . . .  **END LOOP**; |

record\_name 암시적으로 선언된 RECORD 이름

cursor\_name 선언되어 있는 CURSOR의 이름

☞ Guidelines

1. LOOP를 제어하는 RECORD를 선언하지 마십시오.
2. 필요하다면 LOOP내에서 CURSOR의 속성을 이용하십시오.
3. 필요하다면 FOR문 안에서 CURSOR이름 다음에 괄호로 CURSOR에 대한 매개변수를 묶어 사용하십시오.
4. CURSOR 작업이 수동으로 처리되어야 할 때는 FOR LOOP를 사용하지 마십시오.
5. LOOP가 시작될 때 질의를 정의할 수 있습니다. 질의 표현식은 SELECT 부속문장이라 불리고 CURSOR는 FOR LOOP내에서만 사용할 수 있습니다. 이름을 가지고 CURSOR가 선언되지 않기 때문에 그 속성을 사용할 수는 없습니다.

문제4) CURSOR FOR LOOP를 사용하여 DEPT 테이블의 자료를 조회하여라.

|  |
| --- |
| SET SERVEROUTPUT ON  DECLARE  CURSOR dept\_cursor IS  SELECT \*  FROM dept  ORDER BY deptno;  BEGIN  DBMS\_OUTPUT.PUT\_LINE('부서번호 부 서 명 위 치');  DBMS\_OUTPUT.PUT\_LINE('-------- ------------- ------------');  FOR dept\_record IN dept\_cursor LOOP  DBMS\_OUTPUT.PUT\_LINE(LPAD(dept\_record.deptno,2) || ' '  || RPAD(dept\_record.dname,15) || RPAD(dept\_record.loc,12));  END LOOP;  END;  /  SET SERVEROUTPUT OFF |

**1.11 SUBQUERY를 사용한 CURSOR FOR LOOP**

PL/SQL이 SUBQUERY를 치환 하도록 하기 때문에 CURSOR는 선언할 필요가 없다.

**1.11.1 Syntax**

|  |
| --- |
| **FOR** record\_name **IN** (subquery) **LOOP**  statement1;  statement2;  . . . . . .  **END LOOP**; |

subquery SELECT문장을 기술

문제5) SUBQUERY를 사용한 CURSOR FOR LOOP를 이용하여 DEPT 테이블의 내용을 조회하여라.

|  |
| --- |
| SET SERVEROUTPUT ON  DECLARE  BEGIN  DBMS\_OUTPUT.PUT\_LINE('부서번호 부 서 명 위 치');  DBMS\_OUTPUT.PUT\_LINE('-------- ------------- ------------');  FOR dept\_record IN (SELECT \* FROM dept ORDER BY deptno) LOOP  DBMS\_OUTPUT.PUT\_LINE(LPAD(dept\_record.deptno,2) || ' '  || RPAD(dept\_record.dname,15) || RPAD(dept\_record.loc,12));  END LOOP;  END;  /  SET SERVEROUTPUT OFF |

**◈ 연 습 문 제 ◈**

1. CURSOR란?

2. CURSOR의 종류를 설명하여라.

3. 담당 업무를 입력받아 이름,업무,급여,부서명,근무지를 출력하여라. 단 두개의 CURSOR(이름,업무,급여:EMP\_CURSOR, 부서명,근무지:DEPT\_CURSOR)를 이용하여 작성하여라.

4. EMP\_DEPTNO\_TOTAL(deptno:NUMBER(2), total:NUMBER(10,2))이라는 테이블을 생성한다.

5. 2번에서 생성한 테이블에 각 부서별 급여의 합을 입력하여라. 단 CURSOR을 이용한다.

6. SQL\*Plus 치환 변수로 급여가 많은 사람을 출력하기 위한 인원수를 입력받아 인원수에 해당하는 사원의 정보를 이름,업무,급여를 출력하여라.

7. 동일한 급여를 받는 사원의 정보를 이름,업무,급여를 출력하여라.