**1. 개요**

여러 가지 제어 구조를 이용하여 PL/SQL 블럭에 있는 문장들의 논리적 흐름을 변경할 수 있다. 조건에 의해 분기하는 IF문을 이용한 조건 구조와 LOOPING구조(조건 없이 반복하는 BASIC루프, 계수를 이용하여 반복을 하는 FOR루프, 문장이 TRUE인 동안에 반복을 하는 WHILE루프, 루프를 종료하는 EXIT문)가 있다.

**1.1 IF문**

PL/SQL의 IF문장은 다른 언어의 IF문장과 거의 유사하다. 즉 일치하는 조건(TRUE,FALSE, NULL)에 따라 선택적으로 작업을 수행할 수 있게 해준다. TRUE면 THEN과 ELSE사이의 문장을 수행하고 FALSE나 NULL이면 ELSE와 END IF사이의 문장을 수행한다.

**1.1.1 Syntax**

|  |
| --- |
| **IF** condition **THEN**  statements;  [**ELSIF**  condition **THEN**]  statements;  [**ELSE**  statements;]  **END IF;** |

condition BOOLEAN 변수 또는 표현식을 기술할 수 있다.(TRUE,FALSE,NULL)

statements 하나 이상의 PL/SQL 또는 SQL문장을 기술한다.

ELSIF 처음식이 FALSE 또는 NULL일 경우 추가적인 조건이 필요한 경우에 사용

**1.1.2 단순 IF문장**

조건이 TRUE이면 THEN이하의 문장을 실행하고 조건이 FALSE나 NULL이면 END IF다음 문장을 수행한다.

**가) Syntax**

|  |
| --- |
| **IF** condition **THEN**  statements;  **END IF**; |

문제1) 이름,급여,부서번호를 입력받아 EMP 테이블에 자료를 등록하는 SCRIPT를 작성하여라. 단 10번 부서일 경우 입력한 급여의 20%를 추가하고 초기값이 9000부터 9999까지 1씩 증가하는 SEQUENCE(EMP\_EMPNO\_SEQ)작성하여 사용하고 아래의 표를 참고하여라.

|  |
| --- |
| 이 름: 홍길동  급 여: 2000  부서번호: 10 |

|  |
| --- |
| SET VERIFY OFF  ACCEPT p\_name PROMPT ' 이 름: '  ACCEPT p\_sal PROMPT ' 급 여: '  ACCEPT p\_deptno PROMPT ' 부서번호: '  DECLARE  v\_name VARCHAR2(10) := UPPER('&p\_name');  v\_sal NUMBER(7,2) := &p\_sal;  v\_deptno NUMBER(2) := &p\_deptno;  BEGIN  IF v\_deptno = 10 THEN  v\_sal := v\_sal \* 1.2;  END IF;  INSERT INTO emp(empno,ename,sal,deptno)  VALUES (empno\_sequence.NEXTVAL,v\_name,v\_sal,v\_deptno);  COMMIT;  END;  /  SET VERIFY ON |

☞ Guidelines

1. 충족하는 조건에 따라 선택적으로 작업을 수행할 수 있다.
2. 코드를 사용할 때 키워드의 철자를 바르게 기술하시오.(ELSIF, END IF)
3. 제어의 조건들이 TRUE이면 THEN과 END IF사이의 관련된 문장들이 수행됩니다. 그러나 FALSE나 NULL이면 END IF다음의 문장으로 제어가 넘어 갑니다.
4. ELSE절은 한번만 사용 가능 합니다.
5. 명확성을 위해 조건적으로 실행되는 문장을 들여쓰기 하는 것이 좋습니다.

**1.1.3 IF - THEN - ELSE 문장의 실행 흐름**

조건이 TRUE이면 THEN부터 ELSE사이의 문장을 수행하고 제어는 END IF다음 문장으로 넘어가고 FALSE나 NULL이면 ELSE부터 END IF사이의 문장을 수행하고 제어는 END IF다음의 문장으로 이동 된다.

**가) Syntax**

|  |
| --- |
| **IF** condition **THEN**  statements;  **ELSE**  statements;  **END IF**; |

문제2) 이름을 입력받아 그 사람의 업무가 MANAGER','ANALYST'이면 급여의 50%를 가산하여 갱신하고 업무가 MANAGER','ANALYST'이 아니면 20%를 가산하는 SCRIPT를 작성하여라.

|  |
| --- |
| SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_name PROMPT ' 이 름: '  DECLARE  v\_empno emp.empno%TYPE;  v\_name emp.ename%TYPE := UPPER('&p\_name');  v\_sal emp.sal%TYPE;  v\_job emp.job%TYPE;  BEGIN  SELECT empno,job  INTO v\_empno,v\_job  FROM emp  WHERE ename = v\_name;  IF v\_job IN ('MANAGER','ANALYST') THEN  v\_sal := v\_sal \* 1.5;  ELSE  v\_sal := v\_sal \* 1.2;  END IF;  UPDATE emp  SET sal = v\_sal  WHERE empno = v\_empno;  DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || '개의 행이 갱신되었습니다.');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE(v\_name || '는 자료가 없습니다.');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE(v\_name || '는 동명 이인입니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('기타 에러가 발생 했습니다.');  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.1.4 IF-THEN-ELSIF 문장의 실행 흐름**

가능하면 중첩 IF문장 대신 ELSIF절을 사용하여라. 코드를 읽고 이해하기가 더 쉬우며 로직을 정확하게 식별됩니다. ELSE절 안의 작업이 순수하게 다른 IF문으로 구성된다면 이것은 ELSIF절을 사용하는 것이 더욱 편리합니다. 조건과 수행이 각각 종료 시에 중첩 END IF에 대해 일일이 요구하지 않음으로써 코드를 더 명확하게 만들어 줍니다.

**가) Syntax**

|  |
| --- |
| **IF** condition **THEN**  statements;  **ELSIF**  condition **THEN**  statements;  **ELSIF** condition **THEN**  statements;  **ELSE**  statements;  **END IF**; |

문제3) 이름을 입력받아 업무를 조회하여 업무별로 급여를 갱신하는 SCRIPT를 작성하여라. 단 PRESIDENT:10%,MANAGER:20%,ANALYST:30%,SALESMAN:40%,CLERK:50%를 적용한다.

|  |
| --- |
| SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_name PROMPT ' 이 름: '  DECLARE  v\_empno emp.empno%TYPE;  v\_name emp.ename%TYPE := UPPER('&p\_name');  v\_sal emp.sal%TYPE;  v\_job emp.job%TYPE;  BEGIN  SELECT empno,job  INTO v\_empno,v\_job  FROM emp  WHERE ename = v\_name;  IF v\_job = 'PRESIDENT' THEN  v\_sal := v\_sal \* 1.1;  ELSIF v\_job = 'MANAGER' THEN  v\_sal := v\_sal \* 1.2;  ELSIF v\_job = 'ANALYST' THEN  v\_sal := v\_sal \* 1.3;  ELSIF v\_job = 'SALESMAN' THEN  v\_sal := v\_sal \* 1.4;  ELSIF v\_job = 'CLERK' THEN  v\_sal := v\_sal \* 1.5;  ELSE  v\_sal := NULL;  END IF;  UPDATE emp |
| SET sal = v\_sal  WHERE empno = v\_empno;  DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || '개의 행이 갱신되었습니다.');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE(v\_name || '는 자료가 없습니다.');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE(v\_name || '는 동명 이인입니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('기타 에러가 발생 했습니다.');  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.2 논리적 조건 설정**

비교 연산자를 써서 숫자, 문자 또는 날짜 식을 결합한 간단한 논리 조건을 만든다. 일반적으로 IS NULL연산자로 NULL값을 처리할 수 있다.

**1.2.1 식과 비교에서 널(NULL) 논리적 조건 설정**

널 값을 공 문자열로 처리하는 연결식(Concatenation)은 예외이지만 기타 다른 널 값을 포함하는 식은 널 값을 return하고 IS NULL비교의 결과는 TRUE나FALSE로 return된다.

**1.2.2 논리 테이블**

논리연산자 AND, OR, NOT을 가지고 단순한 BOOLEAN조건을 조합함으로써 복잡한 BOOLEAN조건을 구축할 수 있다.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AND | TRUE | FALUE | NULL |  | OR | TRUE | FALUE | NULL |  | NOT |  |
| TRUE | TRUE | FALUE | NULL |  | TRUE | TRUE | TRUE | TRUE |  | TRUE | FALUE |
| FALUE | FALUE | FALUE | FALUE |  | FALUE | TRUE | FALUE | NULL |  | FALUE | TRUE |
| NULL | NULL | FALUE | NULL |  | NULL | TRUE | NULL | NULL |  | NULL | NULL |

**1.3 LOOP문**

LOOP문은 일련의 문장(SQL,PL/SQL)들을 여러 번 반복하기 위해 많은 편의를 제공한다.

1. 조건 없이 반복 작업을 제공하기 위한 BASIC LOOP문
2. COUNT를 기본으로 작업의 반복 제어를 제공하는 FOR LOOP문
3. 조건을 기본으로 작업의 반복 제어를 제공하기 위한 WHILE LOOP문
4. LOOP를 종료하기 위한 EXIT문

**1.4 BASIC LOOP문**

가장 간단한 루프는 구분 문자인 LOOP와 END LOOP사이에 반복되는 문장 부분들로 이루어져 있다. 실행상의 흐름이 END LOOP에 도달할 때마다 그와 짝을 이루는 LOOP 문으로 제어가 되돌아간다. 이러한 루프를 무한 루프라 하며, 여기서 빠져나가려면 EXIT문을 사용한다. 기본 LOOP는 LOOP에 들어갈 때 조건이 이미 일치했다 할지라도 적어도 한번은 문장이 실행된다.

**1.4.1) Syntax**

|  |
| --- |
| **LOOP**  statement1;  statement2;  . . . . . .  **EXIT**  [**WHERE** condition];  **END LOOP**; |

**1.5 EXIT 문**

EXIT 문을 이용하면 END LOOP 문 다음 문으로 제어를 보내기 때문에 루프를 종료할 수 있다. EXIT는 IF문 내의 처리 작업으로서 또는 루프 내의 독립적인 문장으로서도 사용할 수 있다. 조건에 따라 루프를 종료할 수 있도록 WHEN 절을 덧붙일 수 있다. EXIT문에 직면하게 되면 조건이 평가 됩니다. 조건이 TRUE를 RETURN하면 LOOP을 끝내고 LOOP 후의 다음 문장으로 제어를 전달합니다. 기본 LOOP는 여러 개의 EXIT문장을 포함할 수 있다.

**1.5.1) Syntax**

|  |
| --- |
| **EXIT** [**WHEN** condition]; |

문제4) LOOP문으로 아래와 같이 출력하는 SCRIPT를 작성하여라.

|  |
| --- |
| \*  \*\*  \*\*\*  . . . . . . . . . .  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  PL/SQL procedure successfully completed. |
| SET SERVEROUTPUT ON  DECLARE  v\_cnt NUMBER := 1;  v\_str VARCHAR2(20) := NULL;  BEGIN  LOOP  v\_str := v\_str || '\*';  DBMS\_OUTPUT.PUT\_LINE(v\_str);  v\_cnt := v\_cnt + 1;  IF v\_cnt >= 20 THEN  EXIT;  END IF;  END LOOP;  END;  /  SET SERVEROUTPUT OFF |

문제5) EVEN\_ODD(ID:NUMBER(4) GUBUN:VARCHAR2(4)) 테이블을 작성하여 START숫자와 END 숫자를 입력 받아 그사이의 숫자를 ID에 ID의 숫자가 짝수이면 GUBUN에 “짝수”를 홀수이면 GUBUN에 “홀수”라고 입력하는 SCRIPT를 LOOP문으로 작성하여라.

|  |
| --- |
| DROP TABLE even\_odd;  CREATE TABLE even\_odd(  id NUMBER(4) CONSTRAINT even\_odd\_id\_pk PRIMARY KEY,  gubun VARCHAR2(4));  SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_start PROMPT ' START 숫자를 입력하시오 : '  ACCEPT p\_end PROMPT ' END 숫자를 입력하시오 : '  DECLARE  v\_start even\_odd.id%TYPE := &p\_start;  v\_end even\_odd.id%TYPE := &p\_end;  BEGIN  IF &p\_start > &p\_end THEN  DBMS\_OUTPUT.PUT\_LINE('START가 END보다 큼니다.');  ELSE  DELETE FROM even\_odd;  LOOP  IF MOD(v\_start,2) = 0 THEN  INSERT INTO even\_odd  VALUES (v\_start,'짝수');  ELSE  INSERT INTO even\_odd  VALUES (v\_start,'홀수');  END IF;  v\_start := v\_start + 1;  EXIT WHEN v\_start > v\_end;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE(&p\_start ||'부터 ' || &p\_end || '까지 ' ||  TO\_CHAR(&p\_end - &p\_start + 1) ||  '건의 자료가 입력되었습니다.');  END IF;  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.6 FOR LOOP문**

FOR LOOP는 기본 LOOP와 동일한 일반 구조를 가집니다. 그리고 PL/SQL이 수행되는 수를 정하기 위해 LOOP키 워드 앞에 제어문을 기술합니다. FOR LOOP문에서 사용되는 인덱스는 정수로 자동 선언되므로 따로 선언할 필요가 없다. FOR LOOP문은 LOOP을 반복할 때마다 자동적으로 1씩 증가 또는 감소한다. REVERSE는 1씩 감소함을 의미한다.

**1.6.1) Syntax**

|  |
| --- |
| **FOR**  index\_counter **IN**  [REVERSE] lower\_bound..upper\_bound **LOOP**  statement1;  statement2;  . . . . . .  **END LOOP**; |

index\_counter upper\_bound나 lower\_bound에 도달할 때까지 LOOP를 반복함으로써

1씩 자동적으로 증가하거나 감소되는 값을 가진 암시적으로 선언된

정수입니다.

REVERSE upper\_bound에서 lower\_bound까지 반복함으로써 인덱스가 1씩

감소되도록 합니다.

lower\_bound index\_counter값의 범위에 대한 하단 바운드값을 지정한다.

upper\_bound index\_counter값의 범위에 대한 상단 바운드값을 지정한다.

☞ Guidelines

1. 일련의 문장들은 두 바운드에 의해 카운트가 결정되고 증가될 때마다 실행 됩니다.
2. 루프 범위의 하단 바운드와 상단 바운드는 리터럴,변수,표현식이 가능하지만 정수로 기술되어야 합니다.
3. 루프 범위의 하단 바운드가 상단 바운드보다 더 큰 값이 기술되면 일련의 문장들은 수행되지 않습니다.
4. 루프 내에서만 인덱스 카운터를 참조할 수 있다. 즉 루프 밖에서는 정의되지 않는다.
5. 인덱스 카운터의 값을 참조하기 위해서 표현식을 사용한다.
6. := 좌측에 인덱스 카운터를 기술할 수 없다.

문제6) FOR문으로 아래와 같이 출력하는 SCRIPT를 작성하여라.

|  |
| --- |
| \*  \*\*  . . . . . . . . . .  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  PL/SQL procedure successfully completed. |

|  |
| --- |
| SET SERVEROUTPUT ON  DECLARE  v\_str VARCHAR2(10) := NULL;  BEGIN  FOR i\_idx IN 1..10 LOOP  v\_str := v\_str || '\*';  DBMS\_OUTPUT.PUT\_LINE(v\_str);  END LOOP;  END;  /  SET SERVEROUTPUT OFF |

문제7) EVEN\_ODD(ID:NUMBER(4) GUBUN:VARCHAR2(4)) 테이블을 작성하여 START숫자와 END 숫자를 입력 받아 그사이의 숫자를 ID에 ID의 숫자가 짝수이면 GUBUN에 “짝수”를 홀수이면 GUBUN에 “홀수”라고 입력하는 SCRIPT를 FOR문으로 작성하여라.

|  |
| --- |
| DROP TABLE even\_odd;  CREATE TABLE even\_odd(  id NUMBER(4) CONSTRAINT even\_odd\_id\_pk PRIMARY KEY,  gubun VARCHAR2(4));  SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_start PROMPT ' START 숫자를 입력하시오 : '  ACCEPT p\_end PROMPT ' END 숫자를 입력하시오 : '  DECLARE  BEGIN  IF &p\_start > &p\_end THEN  DBMS\_OUTPUT.PUT\_LINE('START가 END보다 큼니다.');  ELSE  DELETE FROM even\_odd;  FOR i\_idx IN &p\_start .. &p\_end LOOP  IF MOD(i\_idx,2) = 0 THEN  INSERT INTO even\_odd  VALUES (i\_idx,'짝수');  ELSE  INSERT INTO even\_odd  VALUES (i\_idx,'홀수');  END IF;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE(&p\_start ||'부터 ' || &p\_end || '까지 ' ||  TO\_CHAR(&p\_end - &p\_start + 1) ||  '건의 자료가 입력되었습니다.');  END IF;  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.7 WHILE LOOP문**

제어 조건이 TRUE인 동안만 일련의 문장을 반복하기 위해 WHILE LOOP문장을 사용한다. 조건은 반복이 시작될 때 체크하게 되어 LOOP내의 문장이 한번도 수행되지 않을 경우도 있다. LOOP을 시작할 때 조건이 FALSE이면 반복 문장을 탈출하게 된다.

**1.7.1) Syntax**

|  |
| --- |
| **WHILE**  condition **LOOP**  statement1;  statement2;  . . . . . .  **END LOOP**; |

condition BOOLEAN 변수 또는 표현식을 기술(TRUE,FALSE,NULL)

문제8) WHILE문으로 아래와 같이 출력하는 SCRIPT를 작성하여라.

|  |
| --- |
| \*  \*\*  . . . . . . . . . .  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  PL/SQL procedure successfully completed. |

|  |
| --- |
| SET SERVEROUTPUT ON  DECLARE  v\_cnt NUMBER := 1;  v\_str VARCHAR2(10) := NULL;  BEGIN  WHILE v\_cnt <= 10 LOOP  v\_str := v\_str || '\*';  DBMS\_OUTPUT.PUT\_LINE(v\_str);  v\_cnt := v\_cnt + 1;  END LOOP;  END;  /  SET SERVEROUTPUT OFF |

문제9) EVEN\_ODD(ID:NUMBER(4) GUBUN:VARCHAR2(4)) 테이블을 작성하여 START숫자와 END 숫자를 입력 받아 그사이의 숫자를 ID에 ID의 숫자가 짝수이면 GUBUN에 “짝수”를 홀수이면 GUBUN에 “홀수”라고 입력하는 SCRIPT를 WHILE문으로 작성하여라.

|  |
| --- |
| DROP TABLE even\_odd;  CREATE TABLE even\_odd(  id NUMBER(4) CONSTRAINT even\_odd\_id\_pk PRIMARY KEY,  gubun VARCHAR2(4));  SET VERIFY OFF  SET SERVEROUTPUT ON  ACCEPT p\_start PROMPT ' START 숫자를 입력하시오 : '  ACCEPT p\_end PROMPT ' END 숫자를 입력하시오 : '  DECLARE  v\_start even\_odd.id%TYPE := &p\_start;  v\_end even\_odd.id%TYPE := &p\_end;  BEGIN  IF v\_start > v\_end THEN  DBMS\_OUTPUT.PUT\_LINE('START가 END보다 큼니다.');  ELSE  DELETE FROM even\_odd;  WHILE v\_start <= v\_end LOOP  IF MOD(v\_start,2) = 0 THEN  INSERT INTO even\_odd  VALUES (v\_start,'짝수');  ELSE  INSERT INTO even\_odd  VALUES (v\_start,'홀수');  END IF;  v\_start := v\_start + 1;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE(&p\_start ||'부터 ' || &p\_end || '까지 ' ||  TO\_CHAR(&p\_end - &p\_start + 1) ||  '건의 자료가 입력되었습니다.');  END IF;  END;  /  SET VERIFY ON  SET SERVEROUTPUT OFF |

**1.8 중첩 LOOP와 레이블**

여러 단계로 루프를 중첩할 수 있습니다.WHILE루프 내에서 FOR루프를,FOR루프 내에서WHILE루프를 주첩할 수 있습니다.대개 중첩 루프가 종결되면 예외가 발생하지 않는 한 둘러싸는 루프가 종결되지 않습니다.레이블 명은 다른 식별자들과 동일한 규칙을 따릅니다.레이블은 같은 라인 또는 다음 라인에서 문장 앞에 위치됩니다.레이블 구분 문자 안에 LOOP라는 글자 앞에 레이블을 위치 시킴으로써 루프를 레이블 시킵니다.루프가 레이블이 되면END LOOP문장 후에 루프 이름을 선택적으로 쓸 수 있습니다.

**1.8.1) 사용 예**

|  |
| --- |
| BEGIN  <<outer\_loop>>  LOOP  v\_count := v\_count + 1;  EXIT WHEN v\_counter > 10;  <<inner\_loop>>  LOOP  . . . . . .  EXIT outer\_loop WHEN total\_done = ‘YES’;  . . . . . .  EXIT WHEN inner\_done = ‘YES’;  . . . . . .  END LOOP inner\_loop;  . . . . . .  END LOOP outer\_loop;  END; |

**◈ 연 습 문 제 ◈**

1. TEST 테이블을 생성하여 1부터 10사이의 값을 입력하는데 3,6,7을 제외하고 등록하여라. 단 TEST 테이블에는 ID(NUMBER(4)) column만 존재한다.

2. EMP 테이블에 등록하는 SCRIPT를 작성하는데 아래의 조건을 만족하도록 한다.

1. SQL\*Plus치환 변수를 사용하여 사원번호,이름,급여,부서번호를 입력받는다.
2. 급여가 0~999이면 40%, 1000~1499이면 30%, 1500~1999이면 20%, 2000~2999이면 10%, 3000이상이면 5%의 보너스를 받는 것으로 한다.

3. EMP 테이블에서 이름을 입력받아 급여를 200으로 나눈 숫자만큼 “\*”를 출력하하여라.

4. LOOP문장의 종류를 설명하여라.

5. START 숫자와 END를 입력받아 그 안에 있는 숫자를 출력하여라.

6. 중첩 LOOP를 설명하여라.

7. LOOP와 WHILE LOOP의 차이점을 설명하여라.