데이터베이스 프로그래밍 - PL/SQL 기본문법(2차시)

2차시: 기본문법

[학습목표]

PL/SQL의 기본 문법 학습

[학습목차]

2-1 Block

PL/SQL IS A BLOCK-STRUCTURED LANGUAGE

2-1-1 Block 구조

```
DECLARE
        V_EMPNO         NUMBER(4)     := 0;     -- 변수선언 AND 초기값 할당
        V_ENAME         VARCHAR2(10);           -- 변수선언, 초기화를 하지 않은경우는?
        V_DEPTNO        NUMBER(2);

BEGIN
        V_EMPNO :=  7778;                       -- 대입연산자:  :=  , 비교연산자:  =
        V_ENAME := 'PL/SQL';

        INSERT INTO EMP(EMPNO,DEPTNO,ENAME)  VALUES(V_EMPNO,V_DEPTNO,V_ENAME);

        COMMIT;

EXCEPTION
        WHEN  OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('INSERT ERR :'||SQLERRM);
                ROLLBACK;

END;
/
```

BLOCK은 ①**선언부**, ②**실행부** ③ **예외처리부** 3부분으로 구성 .

① **선언부(DECLARE SECTION)는 정의하는 영역.**

변수,상수,커서,사용자정의 예외등을 정의.
선언부는 Block내에서 선택적 영역(OPTIONAL SECTION) 으로.
즉 BLOCK내에서 선언(정의)할 대상이 없는 경우 생략 가능

② 실행부(EXECUTION SECTION)는 실행하는 영역.
선언부에서 정의한 변수에 값을 대입하거나 연산을 실행하는 영역.
실행부는 Block 내에서 필수적 영역(MANDATORY SECTION)
BEGIN 으로 시작하여 END로 종료.

<참고> 처음 PL/SQL 접하는 학습자들은 종종 END 다음에 세미콜론(;)을 생략하여 구문 에러(Syntax Error)가 나타나는 경우 발생. PL/SQL의 문장 종결자는 세미콜론(;) 이고 BLOCK 의 문장 종결자도 세미콜론(;)

③ 예외처리부(EXCEPTION SECTION)는 예외 처리를 하는 영역 .
예외 처리부는 예외(실행시간에 발생한 에러)를 처리하는 영역입니다.
예외 처리부는 Block 내에서 선택적 영역(OPTIONAL SECTION)으로
Block내에서 처리해야할 예외가 없는 경우 생략 가능한 영역.

```
BEGIN
        INSERT INTO EMP(EMPNO,DEPTNO,ENAME)  VALUES(7779,30,'PLSQL');
        COMMIT;
END;
/
```

2-1-2 Nested Block (중첩 Block)

Nested Block은 Block안에 Block이 중첩되어 정의되는 것을 의미합니다.
PL/SQL은 논리적으로 연관된 PROGRAM문장을 BLOCK단위의 구조로 처리합니다.
NESTED BLOCK이 가능하기 때문에 구조화된 프로그램 처리를 할수 있습니다.. 즉 MODULE화된 프로그램이 가능합니다.

```
BEGIN
                ~
                BEGIN
                        ~
                END;

EXCEPTION
        WHEN  OTHERS THEN
                BEGIN
                        ~
                END;

END;
/
```

Nested Block은 실행부 와 예외처리부 에서 정의 가능

2-1-3 Block의 종류

PL/SQL은 Anonymous Block 과 Named Block(=Stored Block) 2가지 종류

① Named Block(=Stored Block)

```
CREATE OR REPLACE FUNCTION CALC_TAX ( SALARY_ARG IN NUMBER)
RETURN NUMBER
IS
        C_TAX_RATE      CONSTANT  NUMBER := 0.05;
        V_RESULT        NUMBER    NOT NULL := 0;
BEGIN
        V_RESULT := SALARY_ARG * C_TAX_RATE;

        IF V_RESULT < 10 THEN
                RETURN  V_RESULT;
        ELSIF V_RESULT  < 100 THEN
             RETURN  V_RESULT * 1.2;      -- 20% 가중치 부가
        ELSE
             RETURN  V_RESULT * 2.5;      -- 150% 가중치 부가
        END IF;
END CALC_TAX;
/
```

② Anonymous Block

```
DECLARE
      V_EMPNO       NUMBER(4)    := 0;    -- 변수선언 AND 초기값 할당
      V_ENAME       VARCHAR2(10);         -- 변수선언, 초기화를 하지 않은경우는?
      V_DEPTNO      NUMBER(2);

BEGIN
      V_EMPNO := 7778;                    -- 대입연산자: := , 비교연산자: =
      V_ENAME := 'PL/SQL';

      INSERT INTO EMP(EMPNO,DEPTNO,ENAME)  VALUES(V_EMPNO,V_DEPTNO,V_ENAME);

      COMMIT;

EXCEPTION
      WHEN  OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('INSERT ERR :'||SQLERRM);
            ROLLBACK;

END;
/
```

① Named Block(= Stored Block)

이름이 부여된 Block입니다. 위의 예제에서 CALC_TAX 가 BLOCK의 이름입니다. Named Block 이라는 용어보다는 Stored Block이라는 용어가 좀더 보편적으로 사용됩니다. Stored라는 용어가 사용된 이유는 DBMS Server내에 저장되는 BLOCK 이라는 의미에서 사용되는 용어.

② Anonymous Block

Anonymous는 사전적 의미 그대로 무명의,이름이 없는 Block.

데이터베이스 프로그래밍 - PL/SQL 기본문법(2차시)

	Named Block	Anonymous Block
Block Name	있다	없다
저장위치	DBMS Server내에 저장 (구체적으로는 Data Dictionary내에 저장)	Client Program내에 저장 되어 있거나 별도의 SQL Script 파일형태로 존재.
실행방식	Client Process가 DBMS Server내에 저장되 있는 Block을 호출하여 실행	실행시마다 Client Program에서 해당 Block Source를 DBMS Server에게 전달하여 실행

2-2 변수

일반적인 3,4세대 언어 처럼 PL/SQL BLOCK내에서 사용하는 변수를 선언할수 있습니다. 선언부(DECLARE SECTION) 에서 변수를 정의 하고 초기화 합니다.

```
DECLARE
        V_EMPNO         NUMBER(4)     := 0;     -- 변수선언 AND 초기값 할당
        V_ENAME         VARCHAR2(10);           -- 변수선언, 초기화를 하지 않은경우는?
        V_DEPTNO        NUMBER(2);

BEGIN
```

① PL/SQL내에서 변수 선언 구문은 관계형 테이블의 컬럼 정의와 구조가 동일

PL/SQL 변수 선언	V_EMPNO NUMBER(4)
관계형 테이블 컬럼정의	EMPNO NUMBER(4)

대입연산자	:=
비교연산자	=

② 변수를 초기화 하지 않고 정의만 한 경우에는 해당 변수에는 NULL 지정

< 참고>

```
DECLARE
    V_SAL               NUMBER(8)   NOT NULL    := 5000;        -- 급여
    C_TAX_RATE          CONSTANT    NUMBER(2,3) := 0.054;       -- 급여세율
BEGIN
```

- 변수 정의시 NOT NULL 이나 CONSTANT로 정의하여 변수의 데이터에

제약(Constraint)사항 정의 가능.

2-3 연산자

연산자 구분	내용
산술연산자	+ , / , * ,-,**
비교연산자	= , != , <> ,~=, <,>, <=,>=
논리연산자	AND,OR,NOT
SQL 연산자	LIKE, BETWEEN,IN, IS NULL

2-4 조건식

```
IF      condition1      THEN
            statement1

ELSIF   condition2    THEN
            statement2
ELSE
            statement3
END IF;
```

① ELSE IF 가 아니라 ELSIF
② END IF 부분도. 문장 종결자 ;
③ condition0이 TRUE 인경우 실행 되고 FALSE,NULL인 경우 패스

```
DECLARE
        V_EMPNO          NUMBER(4)     := 8888;    -- 변수선언 및 초기화
        V_DEPTNO         NUMBER(2);
        V_ENAME          VARCHAR2(10) := 'XMAN'; -- 변수선언 및 초기화
        V_JOB            VARCHAR2(9);
        V_SAL            NUMBER(7,2);
BEGIN
        V_DEPTNO := 20;          -- 변수에 값을 대입

        IF V_JOB IS NULL THEN
                V_JOB := '신입';
        END IF;

        IF V_JOB = '신입'    THEN
                V_SAL := 2000;
        ELSIF V_JOB IN ('MANAGER','ANALYST') THEN
                V_SAL := 3500;
        ELSE
                V_SAL := 2500;
        END IF;

        INSERT INTO EMP(DEPTNO,EMPNO,ENAME,SAL,JOB)
                     VALUES(V_DEPTNO,V_EMPNO,V_ENAME,V_SAL,V_JOB);

        COMMIT;
END;
/
```

① V_JOB 변수는 선언부에 선언한후 초기화를 하지 않아 NULL 할당 되고
IS NULL 연산자를 사용하여 처리.

② IF 조건식을 사용 하여 사원의 JOB에 따라 조건적으로 급여를 지급 하려는
로직 , ELSIF 를 보면 조건절의 BOOLEAN을 처리하기위해 IN 연산자가
사용되었다. PL/SQL내의 제어문(IF,LOOP,GOTO)문을 사용할 때 위의 예제 처럼
SQL 연산자를 사용할수 있다.

③ V_DEPTNO := 20; 변수에 값을 대입하는 대입연산자는 := 입니다.
그 아래 조건식의 V_JOB = '신입사원' 에서 비교 연산자는 = 입니다.

대입연산자	:=
비교연산자	=

2-5 반복문

PL/SQL에는 3가지 유형의 반복문.

① 기본 Loop
Ex) LOOP ~ END LOOP;

② FOR Loop
Ex) FOR ~ LOOP ~ END LOOP;

③ WHILE LOOP
Ex) WHILE ~ LOOP ~ END LOOP;

2-5-1 기본 LOOP

```
DECLARE
        LOOP_INDEX          NUMBER(4)  :=  1;
        MAX_LOOP_INDEX      NUMBER(4)  :=  30;
BEGIN
        LOOP
            DBMS_OUTPUT.PUT_LINE('LOOP COUNT => '||TO_CHAR(LOOP_INDEX));

            LOOP_INDEX  :=  LOOP_INDEX + 1;

            EXIT   WHEN  MAX_LOOP_INDEX  < LOOP_INDEX ;

        END  LOOP;
END;
/
```

데이터베이스 프로그래밍 - PL/SQL 기본문법(2차시)

① 기본 LOOP 구문. LOOP로 시작하고 END LOOP로 종료 됩니다.
END LOOP 다음에 세미콜론(;)을 문장 종결자로 사용해야 합니다.

PL/SQL 문법은 다양한 변형이 가능한 C 언어 문법과 달리 문법 자체가 규칙적으로 정형화 되어 있습니다.

IF ~ END IF;
LOOP ~ END LOOP;

IF 로 시작하면 END IF 와 세미콜론으로 종결 됩니다.
LOOP로 시작하면 END LOOP 와 세미콜론으로 종결 됩니다.

② LOOP 구문 사용시 EXIT절 부분에 주의를 기울여야 합니다.
LOOP는 구문은 명시적으로 LOOP를 종료하는 구문을 사용해야 합니다.
명시적으로 종료하는 일반적인 방법이 EXIT 구문을 사용하는 것 입니다.
일반 프로그래밍과 동일하게 LOOP를 종료하는 구문에 주의를 기울이지 않으면 무한 LOOP를 수행하게 됩니다.

< 참고 >
LOOP는 2가지 유형의 LOOP 종료 방식을 사용 할수 있습니다.
① EXIT를 사용하는 방법 ② IF 문을 사용하여 조건 CHECK후 EXIT를 하는방법

```
LOOP
    statement1;
    statement2;
    EXIT  [ WHEN  conditon ];

END  LOOP;
```

```
LOOP
    statement1;
    statement2;
    IF  condition  THEN
        EXIT;
    END IF;
END  LOOP;
```

2-5-2 FOR LOOP

FOR LOOP는 기본 LOOP 반복문에 LOOP COUNTER 기능이 추가된 형태입니다. LOOP COUNTER 라는 것은 FOR 문장에서 LOOP INDEX(LOOP 첨자)를 제어하는 것을 의미.

```
FOR  loop_index  in  [REVERSE] lower_bound .. upper_bound  LOOP
    statement1;
    statement2;
END  LOOP;
```

데이터베이스 프로그래밍 - PL/SQL 기본문법(2차시)

loop_index 의 특징을 살펴 봅시다.

① pl/sql block내에서 정수 데이터 타입(INTEGER TYPE)으로 암시적,자동으로(implicit)으로 선언된다.(integer type)

② 증가,감소 폭은 1 , 증가폭을 임의로 조정 할수 없습니다.

③ 참조만 가능하고 loop 내에서 임의로 수정할수 없습니다.
예를들면 다른 변수가 loop_index를 참조할수 있지만 해당 loop내에서 C 언어 처럼 개발자가 임의로 수정할수 없습니다.

ex) V_tmp := loop_index; (O)
loop_index := loop_index + 1; (X)

lower_bound .. upper_bound : loop_index 가 실행되는 시작 및 종료 범위

LOWER BOUND ~ loop index의 작은값
UPPER BOUND ~ loop index의 큰값
.. ~ FROM ~ TO의 의미입니다.

REVERSE ~ UPPER BOUND에서 LOWER BOUND로 1씩 감소

```
DECLARE
     LOOP_INDEX      NUMBER(4) :=  1;
     MAX_LOOP_INDEX  NUMBER(4) := 30;
BEGIN
        FOR   LOOP_INDEX  IN  1..30
     -- FOR   LOOP_INDEX  IN  30..1
     -- FOR   LOOP_INDEX  IN  REVERSE 1..30
     -- FOR   LOOP_INDEX  IN  REVERSE 30..1
     LOOP
               DBMS_OUTPUT.PUT_LINE('LOOP COUNT '||TO_CHAR(LOOP_INDEX));
     END LOOP;
END;
/
```

① FOR LOOP는 1 라인만 이해하면 기본 LOOP 구문과 동일 합니다.
위의 예제에서 LOOP_INDEX 라는 변수를 선언부에 정의 했습니다.
FOR LOOP에서 사용되는 LOOP 첨자 변수인 LOOP_INDEX 와 동일한 이름으로 정의되었습니다.
두변수는 이름만 동일하고 서로 아무런 상관 관계가 없습니다.
FOR LOOP내에서 사용되는 LOOP_INDEX 는 별도의 변수명 정의를 할 필요 없이LOOP내에서 자동으로 정의되여 지역변수처럼 LOOP내에서만 사용됩니다.

② IN 구문 다음에는 FOR LOOP의 실행 범위가 나타납니다.
주의 하실 사항은 시작값(작은값)이 먼저 종료값(큰값)이 나중에 나타납니다.
문법적으로 기억해두실 사항입니다.

마치 BETWEEN A AND B 구문에서 A 에는 작은값 B에는 큰값이 나와야 하는것과 유사 합니다. 자리 위치가 의미를 가지게 됩니다.

2-5-3 WHILE LOOP

WHILE LOOP : 기본적인 LOOP 반복문에 BOOLEAN CHECK 기능 추가

```
WHILE condition
LOOP
    statement1;
    statement2;
END LOOP;
```

FOR LOOP는 일정한 횟수만큼 LOOP 반복
WHILE LOOP는 조건식이 참(TRUE)일 동안 반복 합니다.

WHILE LOOP에서 중요하게 기억해두실 사항은
조건식(condition)이 TRUE인 경우만 실행되고 FALSE 나 NULL인 경우 실행되지 않는다는 부분입니다. 다른 언어로 개발해본 경험을 가지는 경우에 TRUE 와 FLASE 에 의한 WHILE LOOP 처리는 익숙하게 이해하는 부분입니다.
PL/SQL은 데이터 처리를 주요 목적으로 하는 언어 이기 때문에 BOOLEAN 연산에 NULL이 포함 되기에 NULL 과 관계된 연산의 결과는 기억해두셔야 하는 사항 입니다.

```
DECLARE
        V_INDEX         NUMBER(3) := 0;
      --V_INDEX         NUMBER(3);
      --V_INDEX         NUMBER(3) := 30;
BEGIN
        WHILE(V_INDEX  >= 0 )
        LOOP
                DBMS_OUTPUT.PUT_LINE(' WHILE LOOP ['||TO_CHAR(V_INDEX)||' ]');
                V_INDEX := V_INDEX - 1;
        END LOOP;
END;
/
```

① 조건식 V_INDEX > 0 의 BOOLEAN CHECK를 통해 TRUE일 동안에 LOOP를 수행

데이터베이스 프로그래밍 - PL/SQL 기본문법(2차시)

2-6 NULL

```
DECLARE
        V_NUM1  NUMBER(4,2);            -- 변수선언
        V_NUM2  NUMBER(4,2) := 30;      -- 변수 선언 AND 초기값 할당
BEGIN
        IF (V_NUM1 > 1 AND V_NUM2 < 31) THEN     --  NULL AND TRUE   ==> 결과는?

                DBMS_OUTPUT.PUT_LINE('(V_NUM1 > 1 AND V_NUM2 < 31) IS TRUE ');

        ELSIF NOT (V_NUM1 > 1 AND V_NUM2 < 31) THEN

                DBMS_OUTPUT.PUT_LINE('(V_NUM1 > 1 AND V_NUM2 < 31) IS FALSE ');

        ELSE
                DBMS_OUTPUT.PUT_LINE(' NOT TRUE, NOT FALSE... ???? ');

        END IF;
END;
/
```

<참고> 프로그램을 개발시 변수 선언후 초기화 하지 않는 경우NULL값이 존재하는 경우 연산(논리연산,비교연산)이 예상치 못하는 결과가 되는 경우가 있다

2-6-2 NULL과 BOOLEAN 연산

BOOLEAN	일반언어	2항 연산	TRUE,FALSE
	PL/SQL	3항 연산	TRUE,FALSE,NULL

NULL과 연관된 논리연산

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	
TRUE	FALSE
FALSE	TRUE
NULL	NULL

데이터베이스 프로그래밍 - PL/SQL 기본문법(2차시)

[학습정리]

1, BLOCK은 선언부,실행부,예외처리부로 구성된다,
선언부는 변수,상수,커서,사용자 정의 예외를 선언하는 선택적 영역이다,
실행부는 연산 처리 및 데이터 처리를 수행하는 필수적 영역이다,
예외처리부는 예외를 처리하는 선택적 영역이다,

2, 대입연산자는 := , 비교연산자는 = 이다,

3, NESTED BLOCK은 BLOCK내에 정의되는 BLOCK이며 모듈화된 프로그래밍이 가능하도록 한다,

4, BLOCK은 Anonymous Block 과 Named Block(Stored Block)으로 나눌수 있다,

5, 반복문 구문은 LOOP , FOR LOOP , WHILE LOOP 3가지 유형이 있다

6 IF문의 조건식은 TRUE일 경우 처리되고 FALSE 나 NULL일 경우는 처리되지 않는다,