Writing Executable Statements

1. PL/SQL 블록 구문 및 지침

- PL/SQL은 SQL의 기능을 확장한 것이므로 SQL에 적용되는 일반 구문 규칙은 PL/SQL 언어에도 적용
- 구분자(Delimiter)
  - 단순기호 (+ - * / = @ ;)
  - 혼합기호 (<> != || -- /* */ :=)
- 식별자(Identifier)
  - 상수, 변수, 예외 사항, 커서, 커서 변수, 하위 프로그램 및 패키지를 포함한 PL/SQL 프로그램 항목 및 단위의 이름을 지정하는 데 사용
  - 최고 30자 까지 가능
  - 예약어는 큰 따옴표(" ")로 묶어야 함
  - 영문자로 시작
  - 데이터베이스 테이블 열 이름과 동일하면 안됨
- 리터럴(Literal)
  - 식별자로 표현되지 않는 명시적 숫자, 문자, 문자열 또는 부울 값
  - 문자 및 날짜 리터럴은 작은 따옴표(' ')로 묶어야 함

1. PL/SQL 블록 구문 및 지침 (계속)

- PL/SQL 블록은 행에 슬래시(/)가 있으면 종결
- 코드에 주석 달기
  - 단일 행 주석에는 대시 두개 (--)를 접두어로 붙임
  - 여러 행 주석은 기호 /*와 */ 사이에 작성

```
.....
   v_sal         NUMBER(9,2);
BEGIN
   /* Compute the annual salary based on the monthly
      salary input from the user */
   v_sal := &p_monthly_sal * 12;
END;    -- This is the end of the block
```

2. PL/SQL 함수

2-1. PL/SQL에서 SQL 함수

- PL/SQL에서 사용 가능한 SQL 함수
  - 단일 행 숫자
  - 단일 행 문자
  - 데이터 유형 변환
  - 날짜
- PL/SQL에서 사용 불가능한 SQL 함수
  - DECODE 함수
  - 그룹 함수

```
v_mailing_address    := v_name||CHR(10)||
                        v_address||CHR(10)||v_state||
                        CHR(10)||v_zip;
```

```
v_ename              := LOWER(v_ename);
```

2-2. 데이터 유형 변환

- 데이터 유형을 비교 가능한 유형으로 변환
- 데이터 유형을 혼합 사용하면 오류가 발생하고 성능에 영향을 줌
- 변환 함수
  - TO_CHAR
  - TO_DATE
  - TO_NUMBER

```
DECLARE
        v_date          VARCHAR2(15);
BEGIN
        SELECT          TO_CHAR(hiredate, 'MON.DD,YYYY')
        INTO            v_date
        FROM            emp
        WHERE           empno = 7839;
END;
```

3. PL/SQL 연산자

- PL/SQL 연산자는 SQL에서 사용되는 것과 동일
- 사용 가능한 연산자와 우선순위(위에서 아래로)

연산자	연 산
**, NOT	지수, 논리적 부정
+, -	항등식, 부정
*, /	곱하기, 나누기
+, -, ||	더하기, 빼기, 연결
=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	비교
AND	논리곱
OR	논리합

```
v_total      := v_count + 1;
```

```
v_equal      := (v_n1 = v_n2);
```

```
v_valid      := (v_empno IS NOT NULL);
```

4. 중첩 블록 및 변수 범위

4-1. 중첩 블록 및 변수 범위 지침

- 실행문이 사용 가능한 경우 언제라도 명령문 중첩 가능
- 중첩 블록은 하나의 명령문
- **EXCEPTION** 섹션 중첩 블록 포함 가능
- 객체의 범위는 객체를 참조할 수 있는 프로그램 영역
- 식별자(Identifier)는 비검증(Unqualified) 식별자를 참조할 수 있는 영역에서 확인
  - 블록은 상위 블록 조회 가능
  - 블록은 하위 블록 조회 불가능

4-2. 중첩 블록 및 변수 범위 활용 예제 (1)

```
...
        x BINARY_INTEGER;
BEGIN
        ...
        DECLARE
                y   NUMBER;
        BEGIN
                ...
        END;
        ...
END;
```

x의 범위

y의 범위

4-3. 중첩 블록 및 변수 범위 활용 예제 (2)

```
...
DECLARE
     v_sal      NUMBER(7,2) := 60000;
     v_comm     NUMBER(7,2) := v_sal * .20;
     v_message VARCHAR2(255):='` eligible for commission';
BEGIN
  DECLARE
     v_sal           NUMBER(7,2) := 50000;
     v_comm          NUMBER(7,2) := 0;
     v_total_comp    NUMBER(7,2) := v_sal+ v_comm;
  BEGIN
     v_message := 'CLERK not '||v_message;
     v_comm := v_sal * .30;
  END;
     v_message := 'SALESMAN'||v_message;

END;
```

4-4. 중첩 블록 및 변수 범위 활용 예제 (3)

- 앞장의 PL/SQL 블록을 평가하고 범위 지정 규칙에 따라 다음 값을 각각 판별하시오.
  - 서브 블록에서 **v_message**의 값
  - 메인 블록에서 **v_message**의 값
  - 서브 블록에서 **v_total_comp**의 값
  - 메인 블록에서 **v_total_comp**의 값
  - 서브 블록에서 **v_comm**의 값
  - 메인 블록에서 **v_comm**의 값

Writing Executable Statements

```
DECLARE
     v_weight  NUMBER(3) := 600;
     v_message VARCHAR2(255):='Product 10012';
BEGIN
  DECLARE
     v_weight       NUMBER(7,2) := 50000;
     v_message      VARCHAR2(255) := 'Product 11001';
     v_new_locn     VARCHAR2(50) := 'Europe';
  BEGIN
     v_weight := v_weight + 1;
     v_new_locn := 'Western ' || v_new_locn;
  END;

     v_weight   := v_weight + 1;
     v_message  := v_message  || ' is in stock';
     v_new_locn := 'Western ' || v_new_locn;
END;
```