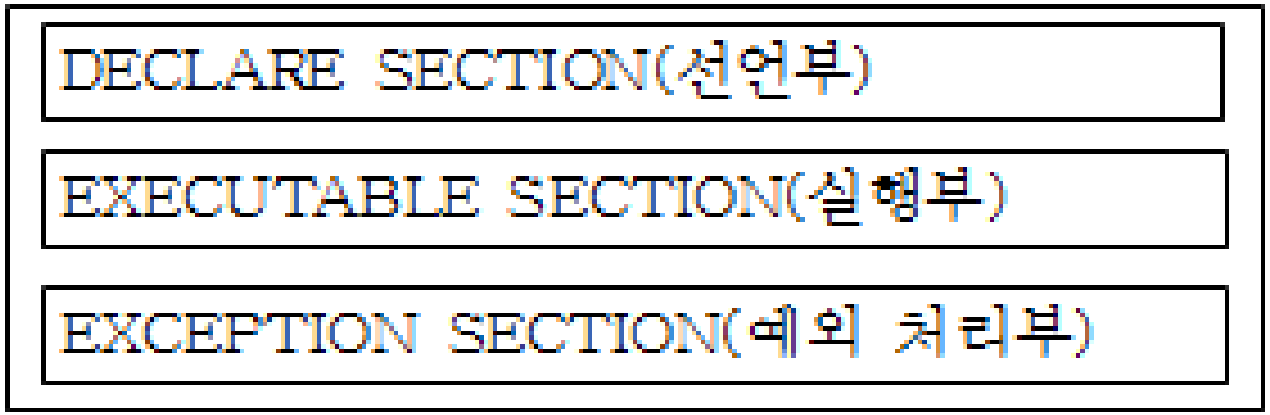
PL / SQL

PL/SQL 구조

- Oracle's Procedural Language extension to SQL의 약자
- 오라클 자체에 내장되어 있는 절차적 언어(Procedure Language)로서 SQL의 단점을 보완
- 오라클사에서는 데이터베이스 내의 데이터를 조작하기 위해서 SQL과 함께 PL/SQL을 제공
- PL/SQL은 SQL에 없는 다음과 같은 기능이 제공
  - 변수 선언
  - 비교 처리
  - 반복 처리

PL/SQL 구조

➢ PL/SQL은 다음과 같은 블록(BLOCK) 구조의 언어로서 크게 3 부분으로 나눌 수 있다.



PL/SQL 구조

- 선언부(DECLARE SECTION)
  - PL/SQL에서 사용하는 모든 변수나 상수를 선언하는 부분으로서 DECLARE로 시작.
- 실행부(EXECUTABLE SECTION)
  - 절차적 형식으로 SQL문을 실행할 수 있도록 절차적 언어의 요소인 제어문, 반복문, 함수 정의 등 로직을 기술할 수 있는 부분으로 BEGIN으로 시작.
- 예외 처리(EXCEPTION SECTION)
  - PL/SQL 문이 실행되는 중에 에러가 발생할 수 있는데 이를 예외 사항이라고 함. 이러한 예외 사항이 발생했을 때 이를 해결하기 위한 문장을 기술할 수 있는 부분으로 EXCEPTION으로 시작.

PL/SQL 구조

➢ PL/SQL 프로그램의 작성 요령

1. PL/SQL 블록내에서는 한 문장이 종료할 때마다 세미콜론(;)을 사용.

2. END뒤에 ;을 사용하여 하나의 블록이 끝났다는 것을 명시.

3. PL/SQL 블록의 작성은 편집기를 통해 파일로 작성할 수도 있고, 프롬프트에서 바로 작성할 수도 있음.

4. SQL*PLUS환경에서는 DECLARE나 BEGIN이라는 키워드로 PL/SQL블럭이 시작하는 것을 알 수 있음.

5. 단일 행 주석은 --이고 여러 행 주석 /* */임.

6. 쿼리문을 수행하기 위해서 /가 반드시 입력되어야 PL/SQL 블록은 행에 / 가 있으면 종결된 것으로 간주.

변수 선언과 대입문

- 쿼리문을 수행하고 난 후에 얻어진 결과를 컬럼 단위로 변수에 저장할 경우 다음과 같이 선언.

```
VEMPNO NUMBER(4);
VENAME VARCHAR2(10);
```

- PL/SOL에서 변수를 선언할 때 위와 같이 SQL에서 사용하던 자료형과 유사하게 선언하는 것을 스칼라(SCALAR) 변수라고 함.

- 숫자를 저장하기 위해서 VEMPNO 변수는 NUMBER로 선언하고 VENAME 변수는 문자를 저장하려면 VARCHAR2를 사용해서 선언.

대입문으로 변수에 값 지정하기

- PL/SQL에서는 변수의 값을 지정하거나 재지정하기 위해서 :=를 사용. := 의 좌측에 새 값을 받기 위한 변수를 기술하고 우측에 저장할 값을 기술.
  - *identifier* := *expression*;

- 선언부에서 선언한 변수에 값을 할당하기 위해서는 :=를 사용.

```
VEMPNO := 7788;
VENAME := 'SCOTT';
```

스칼라 변수 / 레퍼런스 변수

- PL/SQL에서 변수를 선언하기 위해 사용할 수 있는 데이터형은 크게 스칼라(Scalar)와 레퍼런스(Reference)로 나눌 수 있음.
- 스칼라
  - PL/SOL에서 변수를 선언할 때 사용되는 자료형은 SQL에서 사용하던 자료형과 거의 유사하다. 숫자를 저장하려면 NUMBER를 사용하고 문자를 저장하려면 VARCHAR2를 사용해서 선언

```
VEMPNO NUMBER(4);
VENAME VARCHAR2(10);
```

- 레퍼런스
  - 이전에 선언된 다른 변수 또는 데이터베이스 컬럼에 맞추어 변수를 선언하기 위해 %TYPE속성을 사용

```
VEMPNO EMP.EMPNO%TYPE;
VENAME EMP.ENAME%TYPE;
```

스칼라 변수 / 레퍼런스 변수

- 레퍼런스

```
VEMPNO EMP.EMPNO%TYPE;
          ①    ②

VENAME EMP.ENAME%TYPE;
          ①    ②
```

- %TYPE속성을 사용하여 선언한 VEMPNO 변수는 해당 테이블(①EMP)의 해당 칼럼(①EMPNO 혹은 ②)의 자료형과 크기를 그대로 참조해서 정의.
- 모든 개발자가 테이블에 정의된 칼럼의 자료형과 크기를 모두 파악하고 있다면 별 문제가 없겠지만, 대부분은 그렇지 못하기 때문에 오라클에서는 레퍼런스(REFERENCES) 변수를 제공.
- 컬럼의 자료형이 변경되더라도 칼럼의 자료형과 크기를 그대로 참조하기 때문에 굳이 레퍼런스 변수 선언을 수정할 필요가 없다는 장점.

스칼라 변수 / 레퍼런스 변수

- %TYPE이 칼럼 단위로 참조한다면 로우(행) 단위로 참조하는 %ROWTYPE이 있음.
- 데이터베이스의 테이블 또는 VIEW의 일련의 컬럼을 RECORD로 선언하기 위하여 %ROWTYPE을 사용함.
- 데이터베이스 테이블 이름을 %ROWTYPE 앞에 접두어를 붙여 RECORD를 선언하고 FIELD는 테이블이나 VIEW의 COLUMN명과 데이터 타입과 LENGTH을 그대로 가져올 수 있음.

```
VEMP EMP%ROWTYPE;
```

- %ROWTYPE을 사용 시 장점은 특정 테이블의 컬럼의 개수와 데이터 형식을 모르더라도 지정할 수 있음.
- SELECT 문장으로 row를 검색할 때 유리.

PL/SQL에서 SELECT 문

- 데이터베이스에서 정보를 추출할 필요가 있을 때 또는 데이터베이스로 변경된 내용을 적용할 필요가 있을 때 SQL을 사용.
- PL/SQL은 SQL에 있는 DML 명령을 지원합니다. 테이블의 행에서 질의된 값을 변수에 할당시키기 위해 SELECT문장을 사용.
- PL/SQL의 SELECT 문은 INTO절이 필요한데, INTO절에는 데이터를 저장할 변수를 기술.
- SELECT 절에 있는 컬럼은 INTO절에 있는 변수와 1대1대응을 하기에 개수와 데이터의 형, 길이가 일치하여야 함.
- SELECT 문은 INTO절에 의해 하나의 행만을 저장할 수 있음.

PL/SQL에서 SELECT 문

```
SELECT EMPNO, ENAME INTO VEMPNO, VENAME
 FROM EMP
 WHERE ENAME='SCOTT';
```

- VEMPNO, VENAME 변수는 컬럼(EMPNO, ENAME)과 동일한 데이터형을 갖도록 하기 위해서 %TYPE 속성을 사용.
- INTO 절의 변수는 SELECT에서 기술한 컬럼의 데이터형 뿐만 아니라 컬럼의 수와도 일치해야 함.

PL/SQL RECORD TYPE

```
SET SERVEROUTPUT ON
DECLARE
  -- 레코드 타입을 정의
  TYPE emp_record_type IS RECORD(
    v_empno    emp.empno%TYPE,
    v_ename    emp.ename%TYPE,
    v_job    emp.job%TYPE,
    v_deptno  emp.deptno%TYPE);

  -- 레코드로 변수 선언
  emp_record  emp_record_type;
```

PL/SQL RECORD TYPE

```
BEGIN
  -- SCOTT 사원의 정보를 레코드 변수에 저장
  SELECT empno,ename, job, deptno
    INTO emp_record
    FROM emp
    WHERE ename = UPPER('SCOTT');

  -- 레코드 변수에 저장된 사원 정보를 출력
  DBMS_OUTPUT.PUT_LINE('사원번호 : ' ||
     TO_CHAR(emp_record.v_empno));
  DBMS_OUTPUT.PUT_LINE('이    름: ' ||
     emp_record.v_ename);
  DBMS_OUTPUT.PUT_LINE('담당업무 : ' ||
     emp_record.v_job);
  DBMS_OUTPUT.PUT_LINE('부서번호 : ' ||
     TO_CHAR(emp_record.v_deptno));
END;
/
```

선택문

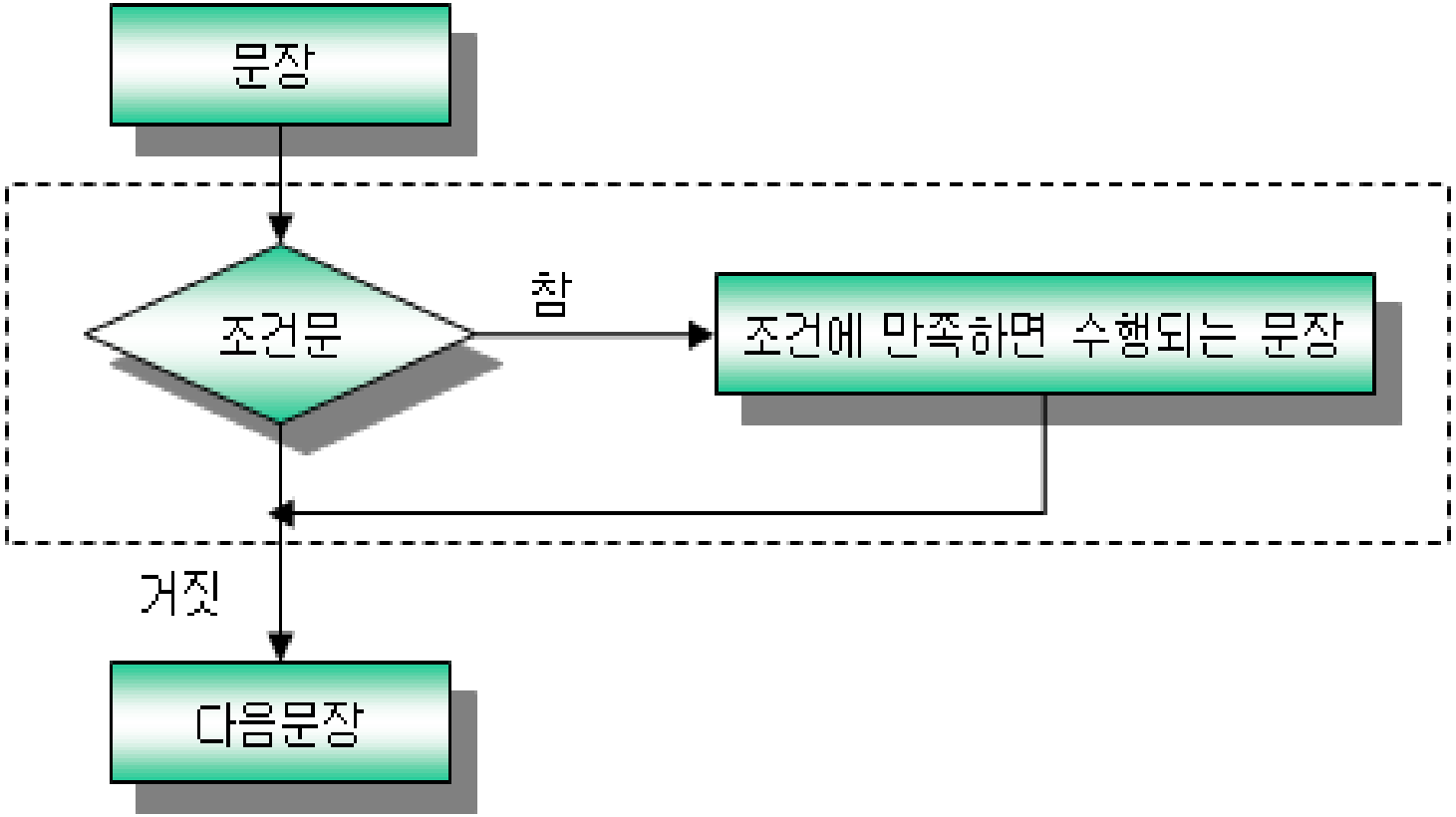
➢IF-THEN-END IF

- if문은 조건에 따라 어떤 명령을 선택적으로 처리하기 위해 사용하는 가장 대표적인 구문

IF *condition* THEN ..... 조건문
***statements;* ..... 조건에 만족할 경우 실행되는 문장**
END IF

선택문

- 조건이 TRUE이면 THEN이하의 문장을 실행하고 조건이 FALSE나 NULL이면 END IF다음 문장을 수행.



선택문

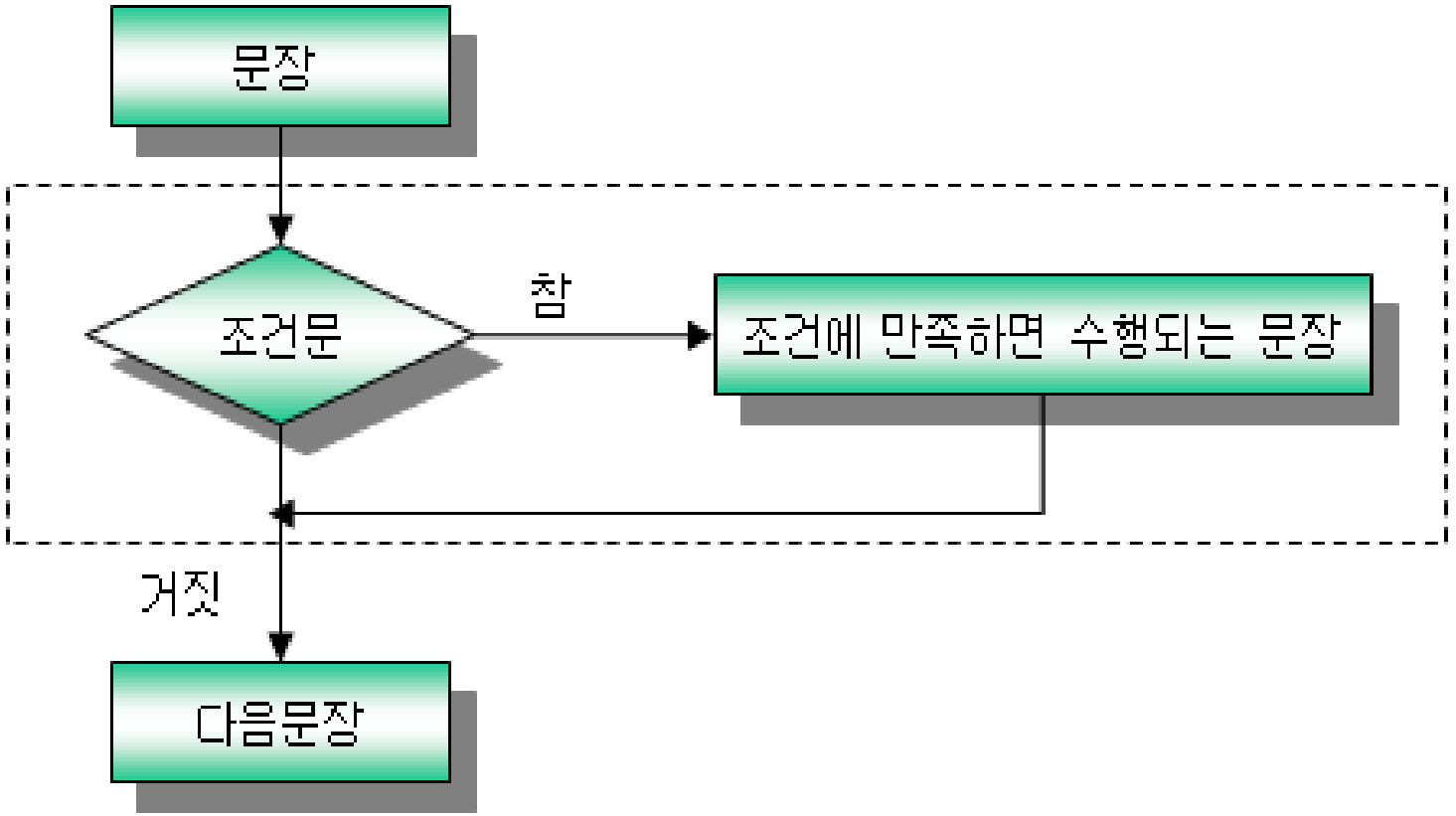
➢IF ~ THEN ~ ELSE ~ END IF

- if문은 조건에 따라 어떤 명령을 선택적으로 처리하기 위해 사용하는 가장 대표적인 구문

[문장1]
IF *condition* THEN ……. 조건문
***statements; ……* 조건에 만족할 경우 실행되는 문장[문장2]**
ELSE
***statements; ……* 조건에 만족하지 않을 경우 실행되는 문장[문장3]**
END IF
[문장4]

선택문

➢ [문장1]을 수행하고 if 문을 만나면 조건문을 검사. 그리고 그 결과가 참이면 [문장2]를 수행, 거짓이면 [문장3]을 수행. 그런 후에는 [문장4]를 수행.

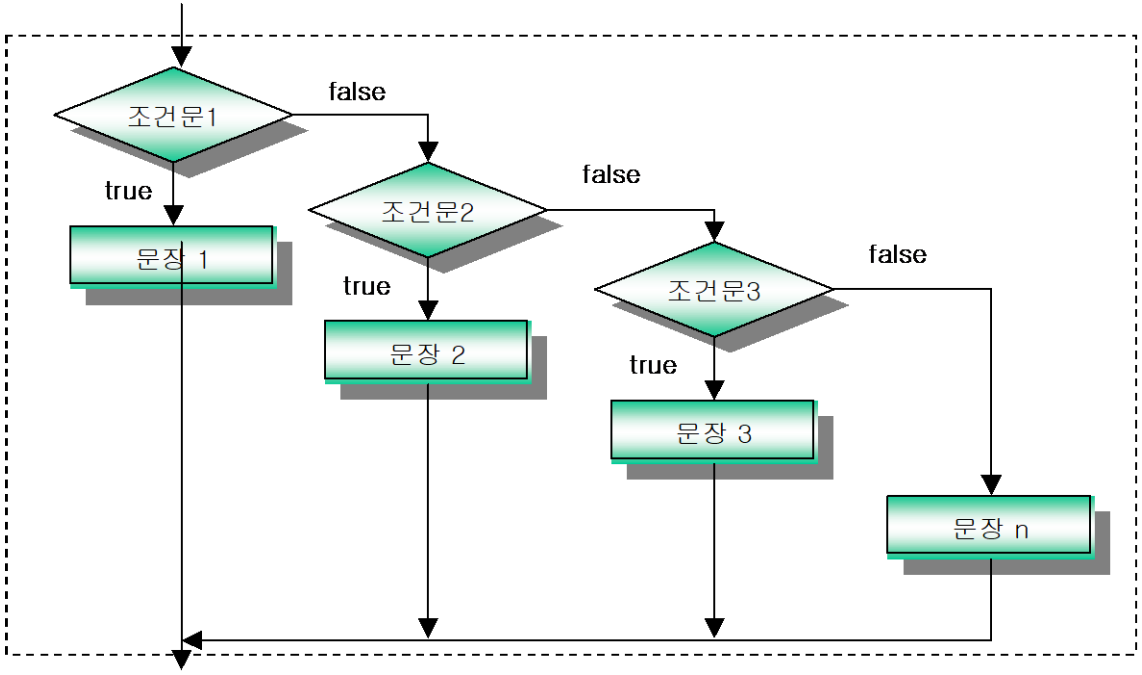


선택문

➢IF ~ THEN ~ ELSIF ~ ELSE ~ END IF

- IF ~ THEN ~ ELSE ~ END IF 문은 참 거짓을 선택하는 과정에서 한번만 사용되었지만, 이럴 경우 둘 중에 하나를 선택할 수 있음.
- 만일 그 경우의 수가 둘이 아닌 셋 이상에서 하나를 선택해야 할 경우에는 IF ~ THEN ~ ELSIF ~ ELSE ~ END IF 문을 사용해야 함.

```
IF condition THEN
statements;
ELSIF condition THEN
statements;
ELSIF condition THEN
statements;
ELSE
statements;
END IF
```



반복문

- 반복문은 SQL 문을 반복적으로 여러 번 실행하고자 할 때 사용.
- PL/SQL에서는 다음과 같이 다양한 반복문이 사용.

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

BASIC LOOP 문

➢BASIC LOOP 문으로 1부터 5까지 출력하기

```
SET SERVEROUTPUT ON
DECLARE
N NUMBER := 1;
BEGIN
LOOP
DBMS_OUTPUT.PUT_LINE( N );
N := N + 1;
IF N > 5 THEN
EXIT;
END IF;
END LOOP;
END;
/
```

FOR LOOP 문

- 반복되는 횟수가 정해진 반복문을 처리하기에 용이.
- FOR LOOP 문에서 사용되는 인덱스는 정수로 자동 선언되므로 따로 선언할 필요가 없다.
- FOR LOOP 문은 LOOP을 반복할 때마다 자동적으로 1씩 증가 또는 감소, REVERSE는 1씩 감소함을 의미.
- FOR LOOP 문으로 1부터 5까지 출력

```
SET SERVEROUTPUT ON
DECLARE
BEGIN
FOR N IN 1..5 LOOP
DBMS_OUTPUT.PUT_LINE( N );
END LOOP;
END;
/
```

WHILE LOOP 문

- 제어 조건이 TRUE인 동안만 일련의 문장을 반복하기 위해 WHILE LOOP 문장을 사용. 조건은 반복이 시작될 때 체크하게 되어 LOOP내의 문장이 한 번도 수행되지 않을 경우도 있음. LOOP을 시작할 때 조건이 FALSE이면 반복 문장을 탈출.

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

WHILE LOOP 문

- WHILE LOOP 문으로 1부터 5까지 출력

```
SET SERVEROUTPUT ON
DECLARE
N NUMBER := 1;
BEGIN
WHILE N <= 5 LOOP
DBMS_OUTPUT.PUT_LINE( N );
N := N + 1;
END LOOP;
END;
/
```