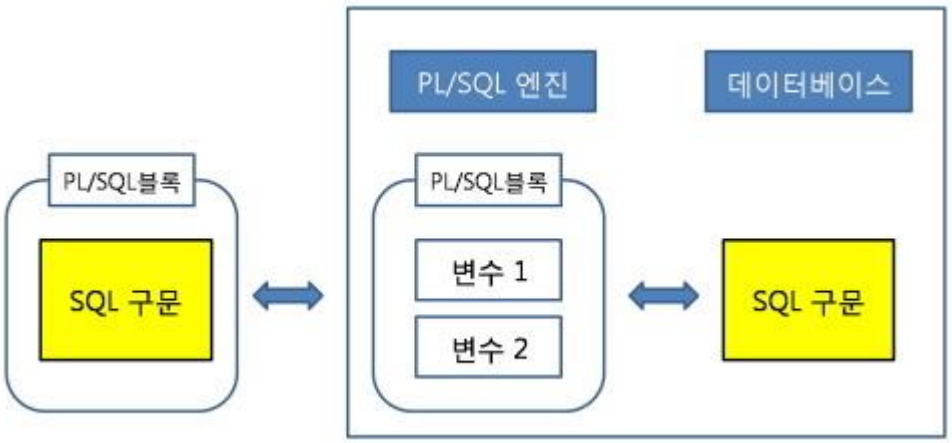
1. PL/SQL이란?

PL/SQL(Procedural Language/SQL)은 오라클 데이터베이스 환경에서 실행되는 절차적인 데이터베이스 프로그래밍 언어이다. 표준 SQL과 3세대 언어의 강력한 일부 기능을 포함한 SQL의 확장 언어이다. PL/SQL에서는 프로그램 단위를 블록(block)이라 부르며, 애플리케이션 로직들을 작성한다.

PL/SQL은 여러 개의 SQL문과 PL/SQL명령문을 프로그램 단위와 같이 PL/SQL 블록(block)에 실행을 보여주고 있다. SQL문을 수행하는데 있어서 PL/SQL 블록을 사용하는 하나의 장점은 네트워크 트래픽(Network Traffic)의 감소를 들 수 있다.



☞ 오라클 환경에서 PL/SQL을 학습하는 이유

- 오라클 개발 도구를 수행하는 모든 프로그래밍의 기초가 된다.
- 클라이언트가 아닌 서버 상에서 프로세스를 수행하는데 PL/SQL을 사용한다.
- PL/SQL을 사용하면 업무 규칙이나 복잡한 로직을 캡슐화(Encapsulation)할 수 있어. 모듈화(Modularity)와 추상화(Abstraction)가 가능하다.
- 데이터베이스 트리거를 통하여 데이터베이스 무결성을 제약하는 복잡한 규칙의 코딩과 변경 내역, 데이터를 복사할 수 있다.
- PL/SQL은 독립적인 프랫폼 수준을 제공한다.

☞ PL/SQL에서 제공하는 명령문은

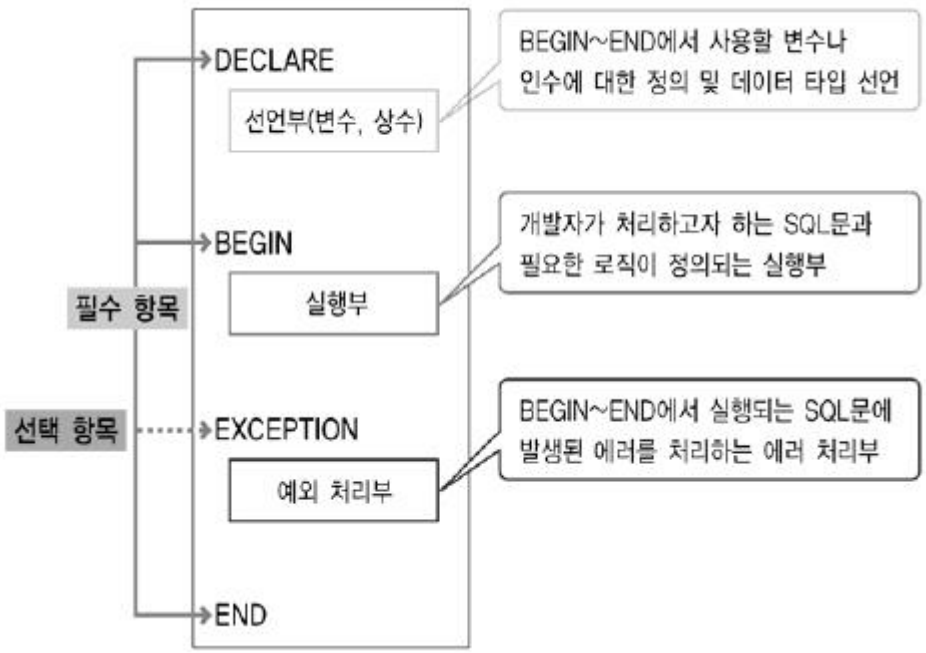
- 모든 SQL문, 변수 및 상수 등의 선언문, 대입문, 조건 판단문, 제어 흐름문, 반복 처리문 등으로 블록을 작성한다.

☞ **PL/SQL로 작성하는 것**

- SQL*Plus 스크립트(Script)
- 프로시저(Procedure), 함수(Function) 서브프로그램
- 패키지(Package)
- 데이터베이스 트리거(Database Trigger)
- 애플리케이션 로직(Application Logic)등이다.

2. PL/SQL의 기본 블록 구조

PL/SQL의 프로그램 단위는 블록(Block)이다. 각 블록은 DECLARE(선언절), BEGIN(실행절), EXCEPTION(예외처리절)의 예약어로 구분하고, BEGIN(실행절)은 필수절이며, DECLARE(선언절)이나 EXCEPTION(예외처리절)은 선택절이다. 블록은 END 예약어로 끝나고, 블록의 마지막 라인에 /를 입력하여 실행한다.



2.1. DECLARE

선언절로 선택절이며, 블록에서 사용되는 변수와 상수, 프로시저와 함수 서브프로그램, 커서 등을 선언한다.

2.2. BEGIN

실행절로 필수절이며, 블록에서 처리할 명령문들을 절차적으로 기술한다. 실행절에는 SQL문, 대입문, 반복 처리문, 조건 판단문, 제어 흐름문, 커서 처리문 등을 기술할 수 있다.

BEGIN 필수절에는 처리에 필요한 명령문들을 절차적으로 기술한다. 절차적으로 기술된 순서에 의해 명령문이 실행될 때,

- 오류가 발행되지 않으면, 블록은 정상적으로 종료한다.
- 오류가 발생되면, 실행을 중단하고, EXCEPTION절로 이동한다.

  √ EXCEPTION절이 기술되어 있지 않으면, 발생한 오류 메시지를 출력하고, 강제로 종료된다.

  √ EXCEPTION절이 기술되어 있으면, 예외처리문을 실행하고, 정상적으로 종료한다. 만약, 오류에 관한 예외처리문이 없더라도 정상적으로 종료한다.

2.3. EXCEPTION

예외처리절이며 선택절이다. BEGIN절에 기술된 명령문, 실행시 오류가 발생하였을 때, 오류 처리에 관한 예외처리 명령문을 기술한다.

3. PL/SQL의 블록 작성 및 실행

3.1. PL/SQL 블록의 작성

☞ PL/SQL 블록을 작성하기 위한 순서

- '무엇을 어떻게 처리할 것인가?'를 분석한다.
- DECLARE를 기술하고, 블록에서 필요한 변수나 상수, 서브프로그램, 커서 등을 문법에 맞추어 한 줄에 하나씩 기술한다.
- BEGIN을 기술하고, 절차적인 실행 순서에 의하여 명령문을 기술한다.
- EXCEPTION을 기술하고, 실행절에 기술된 명령문이 실행될 때 발생될 수 있는 오류에 대한 예외처리문을 기술한다.
- 마지막으로 END;을 기술한다.
- / 입력하여 실행한다.
- PL/SQL 블록에서 한 문장이 종료할 때마다 세미콜론(;)을 기술한다.

예) SG_Scores 테이블에서 학번이 'C0901'인 학생의 평균점수를 계산하여 출력하는 익명의 블록을 작성.

4. 선언절에 변수, 상수 선언

PL/SQL 블록에서 사용하는 수(number)는 상수와 변수가 있다.

• 상수(constant number)는 프로그램이 실행하여 종료될 때까지 일정한 값을 가지는 수이며, 상수에는 어떤 값을 대입하여 사용할 수 없다. 12, 87과 같은 숫자형 리터럴과 'Computer', '홍길동', 'T'와 같은 문자형 리터럴이 있다.

• 변수(varuable number)는 프로그램이 실행하여 종료될 때까지 변하는 수이며, 값을 저장할 때마다 변한다. i, Course_ID등과 같이 문자로 기술하여 지정한다.

4.1. PL/SQL의 데이터 타입

데이터타입	설 명
BOOLEAN	논리적인 데이터(True, False, Unknown)을 저장할 때 사용
BINARY_INTEGER	부호가 있는 정수를 저장할 때 사용 데이터 범위는 -2147483647부터 2147483647까지
• NATURAL	정수 중에서 0부터 2147483647까지 저장할 때 사용
• POSITIVE	정수 중에서 1부터 2147483647까지 저장할 때 사용
%TYPE	기존 테이블의 한 칼럼에 정의된 데이터타입을 참조
%ROWTYPE	기존 테이블의 모든 칼럼에 정의된 데이터타입을 참조

4.2. 변수와 상수 선언

변수나 상수는 단지 하나의 값을 가질 수 있고, 데이터타입과 크기를 지정하며, 초기 값을 지정할 수 있다.

문법	변수명 데이터타입(크기) [:=초기값]; 상수명 CONSTANT 데이터타입(크기) :=상수값 ;

☞ **변수나 상수를 선언할 때는**

• SQL 객체명과 동일한 규칙으로 정의한다.

• 변수와 상수는 반드시 데이터타입과 일치하는 값을 기술한다.

• 한 줄에 한 개의 변수나 상수를 정의한다.

• 초기 값이 NULL이거나 0인 경우 생략할 수 있다.

```
P_count          NUMBER(3)    :=0;          ☜ 변수 선언
Zip_Code         VARCHAR2(9);               ☜ 변수 선언
K    CONSTANT    POSITIVE     := 100;       ☜ 상수 선언
```

4.3. 테이블을 참조하여 변수 선언

%TYPE과 %ROWTYPE 데이터타입은 기존 테이블에 정의된 데이터타입과 크기를 참조하여 변수를 선언한다. %TYPE은 단일 칼럼을 참조하고, %ROWTYPE은 복수 칼럼을 참조한다. 이 변수 선언은 테이블의 한 칼럼이나 행의 데이터를 임시로 저장하여 사용할 때 매우 유용하게 사용될 수 있다.

☞ %TYPE 데이터타입

기존 테이블의 칼럼에 선언된 데이터타입과 크기를 참조하여 한 개의 변수를 선언한다. 이때 테이블 칼럼의 제약조건은 적용되지 않는다. %TYPE은 테이블의 칼럼에 대한 데이터타입이나 크기가 변경되더라도 수정할 필요가 없기 때문에 테이블 칼럼에 관련된 데이터와 비교하거나, 데이터를 저장하는 변수를 선언할 때 유용하다.

문법	변수명	테이블명.칼럼명%TYPE ;

예) SG_Scores 테이블을 참조하여 C0901 학번과 평균점수를 저장하기 위한 변수를 %TYPE 형식을 이용하여 선언하시오.

```
DECLARE
    v_student_ID        SG_Scores.Student_ID%TYPE :='C0901';
    v_age               SG_Scores.Score%TYPE      :=0;
    v_grade             SG_Scores.Grade%TYPE;
    v_cnt               NUMBER(2)                 :=0;
.....
```

☞ %ROWTYPE 데이터타입

기존 테이블의 각 칼럼에 정의된 데이터타입과 크기를 참조하여, 변수명에 테이블 칼럼 수와 동일한 복수 개의 변수가 선언되며, 각 기억 장소의 구분은 "변수명.칼럼명"으로 구분한다.

문법	변수명	테이블명.%ROWTYPE ;

%ROWTYPE 데이터타입은 테이블 칼럼에 대한 데이터타입이나 크기를 알 필요가 없고, 테이블의 구조가 변경되더라도 수정할 필요가 없다. %ROWTYPE 데이터타입은 SELECT문의 결과 행을 저장하는 변수를 지정할 때 자주 유용하다.

예) SG_Scores 테이블을 참조하여 C0901학번과 평균점수를 저장하기 위한 변수를 %ROWTYPE 을 이용하여 선언하시오.

4.4. PL/SQL 테이블과 사용자 정의 레코드 선언

PL/SQL 테이블과 사용자 정의 레코드는 하나의 이름으로 복수 개의 변수를 선언하는 방법이다.

☞ PL/SQL 테이블 선언

데이터타입과 크기가 동일한 기억장소가 동적으로 복수 개 선언된다. 테이블 선언을 위한 데이터타입을 먼저 선언하고, 선언된 테이블타입을 참조하여 테이블을 선언한다.

문법	TYPE 데이터타입명 IS TABLE OF 데이터타입(크기) INDEX BY BINARY_INTEGER; 테이블명 테이블타입명;

PL/SQL 테이블 변수의 각 기억장소 구분은 "테이블명(첨자명)"을 사용한다. 첨자명은 정수로 된 첨자명이거나, 양의 정수로 된 첨자이어야 한다. 테이블의 크기는 처음부터 정해지지 않고, 사용할 때 동적으로 결정된다.

▶ PL/SQL 테이블 선언 및 사용 예

```
DECLARE
        TYPE    Table_type   IS   TABLE OF          ☜ 테이블타입 선언
                VARCHAR2(20)
        INDEX   BY   BINARY_INTEGER;
        V_Dept_Name          Table_type;          ☜ 테이블변수 선언
BEGIN
        ....
        V_Dept_Name(2)      :='컴퓨터공학과';
        ......
END;
```

☞ 사용자 정의 레코드 선언

데이터타입과 크기가 다른 기억장소가 복수 개 선언된다. 사용자가 레코드타입을 먼저 선언하고, 이 레코드 타입을 이용하여 사용자 정의 레코드를 선언한다.

문법	TYPE 레코드타입명 IS RECORD (필드명1 데이터타입(크기) [NOT NULL] [초기값], ... 필드명n 데이터타입(크기) [NOT NULL] [초기값]); 레고크명 레코드타입명;

각 기억 장소의 구분은 "레코드명.필드명"으로 구분한다.

▶ 사용자 정의 레코드 선언 및 사용 예

```
DECLARE
        TYPE    Record_Type    IS RECORD                    ☜ 레코드타입 선언
                Dept_ID          Department.Dept_ID%type;
                Dept_Name        Department.Dept_Name%type;
        Dept_Rec                 Record_Type;                ☜ 레코드변수 선언
        ……
BEGIN
       ……
       Dept_Rec.Dept_ID       :='컴공';
       Dept_Rec.Dept_Name   :='컴퓨터공학과';
       ……
END;
```

5. 조건 판단문

IF~THEN~ELSIF문을 이용해 조건 결과가 참이면 THEN 이하의 명령문을 처리하고, 거짓이면 ELSIF 절을 실행한다. 조건이 모두 거짓이면 ELSE절을 실행한다. ELSIF절과 ELSE절은 선택절이다. ELSIF문은 최대 16개까지 반복하여 사용할 수 있다.

문법	IF 조건1 THEN 명령문1; …… ; 명령문N; [ELSIF 조건2 [THEN 명령문1; …… ; 명령문N;] [ELSIF 조건3 …… [ELSE 명령문1; …… ; 명령문N;] END IF;

▶ IF ~ THEN ~ ELSIF문의 코딩 예

```
IF    v.Score >= 90
            THEN   v.Grade   := 'A' ;
            ELSIF    v.Score >= 80
                     THEN   v.Grade   := 'B' ;
                     ELSIF   v.Score >= 70
                              THEN   v.Grade   := 'C' ;
```

```
                    ELSIF     v.Score >= 60
                              THEN    v.Grade   := 'D' ;
        ELSE     v.Grade   := 'F';
END IF;
```

6. 반복문

6.1. LOOP문

LOOP문은 LOOP와 END LOOP내의 명령문들을 반복 처리하는 무한 루프문이다.

문법	LOOP 명령문1; ···· ; 명령문N; END LOOP;

LOOP~END LOOP문의 처리 명령문에는 무한루프에서 탈출하는 명령문이 반드시 있어야 한다.

☞EXIT문

무한루프로부터 탈출하는 명령문으로, 두 가지 종류가 있다.

문법	① EXIT; ② EXIT [레이블명] WHEN 조건;

- EXIT문은 무한루프부터 조건 없이 탈출한다.
- EXIT WHEN 조건은 조건이 참(true)일 때 탈출한다.

예) 1에서 10까지 반복하여 TEMP 테이블에 저장하시오.

6.2. WHILE ~ LOOP문

조건이 참(true)일 때, LOOP~END LOOP 내의 명령문들을 반복 처리한다. 조건이 거짓(false)이면 반복 처리는 종료된다. 따라서 반복 명령문 내에 조건을 처리하는 명령문이 내포되어 있어야 한다.

문법	WHILE 조건 LOOP 명령문1; ·····; 명령문N; END LOOP;

예) 1에서 10까지 반복하여 TEMP 테이블에 저장하시오.

6.3. FOR~LOOP문

FOR~LOOP문은 첨자변수가 초기값부터 1씩 증가하여 최종값이 될 때까지 LOOP~END LOOP 내의 명령문들을 반복 처리한다. 선택인 REVERSE는 1씩 감소하면서 반복 처리한다.

문법	FOR 첨자변수 IN [REVERSE] 초기값... 최종값 LOOP 명령문1; ·····; 명령문N; END LOOP;

예) 1에서 10까지 반복하여 TEMP 테이블에 저장하시오.

7. NULL문

어떤 처리도 하지 않는다. 단지 IF문 등에서 문법 형식을 갖추기 위해 사용될 수 있다.

▶ NULL 예제
IF a>0 THEN NULL; ELSE i = i+1; END IF;

8. 제어문

GOTO문은 레이블명으로 이동한다. 이동할 위치는 <<레이블명>>으로 지정한다.

문법	GOTO 레이블명;

9. 주석 달기

PL/SQL 블록에 처리되는 내용을 설명하기 위해 주석을 기술한다. 주석은 실행과는 무관하며, 주석을 기술하는 방법이 두 가지가 있다.

- -- 문자열

이것은 주로 단일 줄에 주석을 달 때 사용한다.

- /* 문자열 */

이것은 주로 여러 줄로 주석을 달 때 사용한다.

10. PL/SQL에서의 SELECT문

실행 과정은 SQL의 SELECT문과 유사하다. 단지 검색된 행의 수가 한 행이 되어야 하고, 검색한

값을 INTO절의 변수에 저장한다. 검색된 행의 수가 0행(NO_DATA_FOUND)이거나 복수 행(TOO_MANY_ROWS)일 때 오류가 발생된다. 또한 ORDER BY절은 사용하지 않는다.

문법	SELECT 칼럼명1, 칼럼명2, 리터럴, 함수, 수식, …… INTO 변수명, …… FROM 테이블명1, 테이블명2, 뷰명1, …… WHERE 검색조건1 …… GROUP BY 칼럼명1, 칼럼명2, …… HAVING 검색조건2;

☞ **SQ과 PL/SQL에서 SELECT문의 차이점**

- SELECT 구문이 다르다. PL/SQL의 SELECT문은 INTO절이 있고, ORDER BY절을 사용하지 않는다.
- 결과 행이 다르다. PL/SQL의 SELECT문 결과 행은 1행이 되어야 하고, 0행이나 다수 행이 되면 예외가 발생한다.
- 결과를 출력할 때, SQL에서는 화면에 표시되나, PL/SQL 블록에서는 INTO절에 기술된 변수에 기억된다.

예) 1. SG_Scores 테이블을 참조하여 'C0901'학번의 과목수, 평균점수를 계산하여 출력하시오.
2. SG_Scores 테이블을 참조하여 'C0904' 학번의 과목수, 평균점수를 계산하여 출력하시오.

2번을 실행하게 되면 PL/SQL 블록에서 실행한 SELECT문의 결과 행이 0행이면 "ORA-01403: 데이터를 찾을 수 없습니다. ORA-06512: 줄 7에서" 오류 메시지를 출력하고, 프로그램은 종료된다.

11. PL/SQL에서 화면에 출력하기

SQL에서는 검색 결과를 화면에 표시한다. PL/SQL에서는 실행 결과나 데이터를 화면상에 표시하는 명령문이 없다. 키보드에서 데이터를 입력하고, 블록에서 화면에 표시할 때는 DMBS_OUTPUT 내장 프로시저를 사용한다.

☞ **DBMS_OUTPUT 프로시저 사용하기 위한 SQL*PLUS 명령어**

문법	SET SERVEROUTPUT ON

☞ **DBMS_OUTPUT 프로시저의 활성화와 비활성화**

문법	DBMS_OUTPUT.ENABLE; DBMS_OUTPUT.DISABLE;

예) 1. 'HELLO WORLD' 문자열을 DBMS_OUTPUT 프로시저를 비활성화하여 출력하고, 활성화한 후 출력하시오.

간단 예제) EC_Order 테이블에서 주문자ID가 'kcchoi'에 대한 거래수, 평균 거래약을 화면에 출력하시오.