14장 PL/SQL 기초

오라클사에서 SQL 언어에 절차적인 프로그래밍 언어를 가미해 만든 것이 PL/SQL에 대해서 학습하겠습니다.

학습 내용

- PL/SQL
- 변수 선언
- 제어문
- 커서

학습목표

- PL/SQL은 선언부, 실행부, 예외 처리부 3개의 블록으로 나뉩니다.
- PL/SQL에서는 스칼라 변수와 레퍼런스 변수를 사용합니다.
- 선택문으로는 IF ~ THEN ~ END IF, IF ~ THEN ~ ELSE ~ END IF, IF ~ THEN ~ ELSIF ~
- ELSE-END IF 3가지를 제공합니다.
- 반복 처리를 위해서 BASIC ~ LOOP, FOR ~ LOOP, WHILE LOOP 문을 제공합니다.
- PL/SQL에서 SQL 문의 결과로 얻어지는 로우가 여러 개일 경우 이를 처리하기 위해 커서를 사용합니다.

01. PL/SQL

- PL/SQL(Oracle's Procedural Language extension to SQL)은 오라클에서 지원하는 프로그래밍 언어의 특성을 수용하여 SQL에서는 사용할 수 없는 절차적 프로그래밍 기능을 가지고 있어 SQL의 단점을 보완합니다.

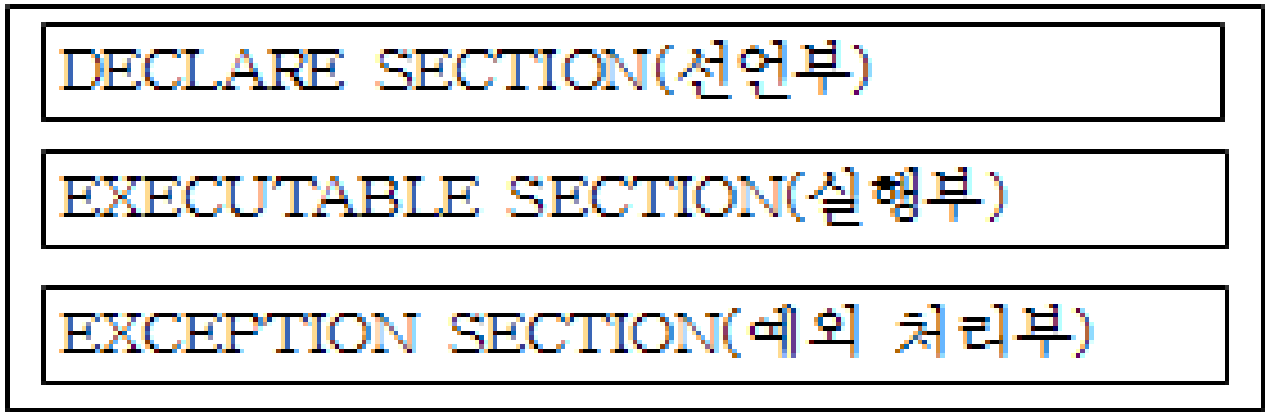
01. PL/SQL

❖ PL/SQL에서 지원되는 절차적 프로그래밍 기능은 다음과 같습니다.

. 변수, 상수 등을 선언하여 SQL과 절차형 언어에서 사용합니다.
. IF문을 사용하여 조건에 따라 문장들을 분기합니다.
. LOOP문을 사용하여 일련의 문장을 반복적으로 실행합니다.
. 커서를 사용하여 여러 행을 검색합니다.

01. PL/SQL

- 기본적인 PL/SQL 블록 구조는 선언부, 실행부, 에외처리부 이렇게 세 부분으로 구성됩니다.



01. PL/SQL

Section	설명	필수여부
DECLARE(선언부)	PL/SQL에서 사용하는 모든 변수나 상수를 선언하는 부분으로서 DECLARE로 시작합니다.	옵션
BEGIN(실행부)	절차적 형식으로 SQL 문을 실행할 수 있도록 절차적 언어의 요소인 제어문, 반복문, 함수 정의 등 로직을 기술할 수 있는 부분으로 BEGIN으로 시작합니다.	필수
EXCEPTION(예외처리부)	PL/SQL 문이 실행되는 중에 에러가 발생할 수 있는데 이를 예외 사항이라고 합니다. 이러한 예외 사항이 발생했을 때 이를 해결하기 위한 문장을 기술할 수 있는 부분으로 EXCEPTION으로 시작합니다.	옵션

01. PL/SQL

❖ PL/SQL 프로그램의 작성 요령은 다음과 같습니다.

1. PL/SQL 블록내에서는 한 문장이 종료할 때마다 세미콜론(;)을 사용합니다.
2. END뒤에 ;을 사용하여 하나의 블록이 끝났다는 것을 명시합니다.
3. PL/SQL 블록의 작성은 편집기를 통해 파일로 작성할 수도 있고, 프롬프트에서 바로 작성할 수도 있습니다.
4. SQL*PLUS환경에서는 DELCLARE나 BEGIN이라는 키워드로 PL/SQL블럭이 시작하는 것을 알 수 있습니다.
5. 단일행 주석은 --이고 여러행 주석 /* */입니다.
6. 쿼리문을 수행하기 위해서 /가 반드시 입력되어야 PL/SQL 블록은 행에 / 가 있으면 종결된 것으로 간주합니다.

01. PL/SQL

❖ 메시지 출력하기

```
set serveroutput on
begin
dbms_output.put_line('Welcome to Oracle');
end;
/
```

02. 변수선언

❖ PL/SQL 블록 내에서 변수를 사용하려면 선언부(DECLARE)에서 선언해야 합니다.

```
identifier [CONSTANT] datatype [NOT NULL]
[:= | DEFAULT expression];
```

구문	설명
identifier	변수의 이름
CONSTANT	변수의 값을 변경할 수 없도록 제약합니다.
datatype	자료형을 기술합니다.
NOT NULL	값을 반드시 포함하도록 하기 위해 변수를 제약합니다.
Expression	Literal, 다른 변수, 연산자나 함수를 포함하는 표현식

02. 변수선언

1) 스칼라

- PL/SQL에서 변수를 선언할 때 사용되는 데이터 타입은 SQL에서 사용하던 데이터 타입과 유사합니다. 숫자, 문자, 날짜, BOOLEAN 4가지로 나뉩니다.

2) 레퍼런스

- 레퍼런스 타입은 변수의 데이터 타입을 데이터베이스 기존 칼럼에 맞추어 선언하기 위해서 %TYPE Attribute를 이용합니다. 변수를 선언할 때 필요한 데이터 타입을 명시적으로 언급하는 대신 ‘TABLE이름.COLUMN이름%TYPE’ 으로 지정합니다.

02. 변수선언

❖ %TYPE은 칼럼 단위로 데이터 타입을 참조합니다.

```
declare
v_eno employee.eno%type;
v_ename employee.ename%type;
```

❖ 로우(행) 전체에 대한 데이터 타입을 참조하려면 %ROWTYPE을 사용합니다.

```
v_employee employee%rowtype;
```

02. 변수선언

❖ 변수에 값을 저장하기 위해서는 ':=' 를 사용합니다. :=의 좌측에는 변수를, 우측에는 값을 기술합니다.

```
identifier := expression;
```

```
v_eno := 7788;
v_ename := 'scott';
```

02. 변수선언

```
SELECT select_list
INTO {variable_name1[,variable_name2,..] | record_name}
FROM table_name
WHERE condition;
```

구문	설명
select_list	열의 목록이며 행 함수, 그룹 함수, 표현식을 기술할 수 있습니다.
variable_name	읽어들인 값을 저장하기 위한 스칼라 변수
record_name	읽어 들인 값을 저장하기 위한 PL/SQL RECORD 변수
Condition	PL/SQL 변수와 상수를 포함하여 열명,표현식,상수,비교 연산자로 구성되며 오직 하나의 값을 RETURN할 수 있는 조건이어야 합니다.

03. 제어문

- 기본적으로 모든 문장들은 나열된 순서대로 순차적으로 수행됩니다.
- 하지만 경우에 따라서는 문장의 흐름을 변경할 필요가 있습니다. 이때 사용하는 것이 IF 문입니다.
- IF 문은 조건을 제시해서 만족하느냐 하지 않느냐에 따라 문장을 선택적으로 수행하기 때문에 선택문이라고 합니다.
- 오라클에서는 3가지 형태의 선택문이 제공됩니다.

3.1 IF

- IF 문은 조건에 따라 어떤 명령을 선택적으로 처리하기 위해 사용합니다.

```
IF condition THEN
statements;
...
[ELSIF condition THEN
statements;
...
ELSIF condition THEN
statements;
...
ELSE
statements;
... ]
END IF;
```

3-2 LOOP 문

- 반복문은 SQL 문을 반복적으로 여러 번 실행하고자 할 때 사용합니다.

- PL/SQL에서는 다음과 같이 다양한 반복문이 사용됩니다.

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

3-2 LOOP 문

- BASIC LOOP는 가장 간단한 구조의 루프로 LOOP와 END LOOP 사이에 반복 수행할 문장을 기술됩니다.

```
LOOP
statement1;
statement2;
. . . . . .
EXIT [WHERE condition];
END LOOP
```

- 실행 상의 흐름이 END LOOP에 도달할 때마다 그와 짝을 이루는 LOOP 문으로 제어가 되돌아갑니다.

- 이러한 루프를 무한 루프라 하며, 여기서 빠져나가려면 EXIT문을 사용합니다.

- 기본 LOOP는 LOOP에 들어갈 때 조건이 이미 일치했다 할지라도 적어도 한번은 문장이 실행됩니다.

3-2 LOOP 문

❖ FOR LOOP는 반복되는 횟수가 정해진 반목문을 처리하기에 용이합니다.

```
FOR index_counter
IN [REVERSE] lower_bound..upper_bound LOOP
statement1;
statement2;
. . . . . .
END LOOP
```

구문	설명
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 지정합니다.
upper_bound	index_counter값의 범위에 대한 상단 바운드값을 지정합니

3-2 LOOP 문

- FOR LOOP 문에서 사용되는 인덱스는 정수로 자동 선언되므로 따로 선언할 필요가 없다.

- FOR LOOP 문은 LOOP을 반복할 때마다 자동적으로 1씩 증가 또는 감소합니다. REVERSE는 1씩 감소함을 의미합니다.

3-2 LOOP 문

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

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

04. 커서

- SELECT 문의 수행 결과가 여러 개의 로우로 구해지는 경우에 모든 로우에 대한 처리를 하려면 커서를 사용해야 합니다. 커서는 CURSOR, OPEN, FETCH, CLOSE 4단계 명령에 의해서 사용됩니다.

```
DECLARE
CURSOR cursor_name IS statement;
BEGIN
OPEN cursor_name;
FECTCH cur_name INTO variable_name;
CLOSE cursor_name;
END;
```

DECLARE CURSOR

- 명시적으로 CURSOR를 선언하기 위해 CURSOR문장을 사용합니다.

```
CURSOR cursor_name
IS
select_statement;
```

구문	의미
cursor_name	PL/SQL 식별자
select_statem ent	INTO절이 없는 SELECT 문장

OPEN CURSOR

- 질의를 수행하고 검색 조건을 충족하는 모든 행으로 구성된 결과 셋을 생성하기 위해 CURSOR를 OPEN합니다. CURSOR는 이제 결과 셋에서 첫 번째 행을 가리킨다.

```
OPEN cursor_name;
```

- 부서 테이블의 모든 내용을 조회하는 SELECT문과 연결된 커서 C1을 오픈합니다.

```
OPEN C1;
```

- C1을 오픈하면 검색 조건에 만족하는 모든 행으로 구성된 결과 셋이 구해지고 부서 테이블의 첫 번째 행을 가리키게 됩니다.

FETCH CURSOR

- FETCH 문은 결과 셋에서 로우 단위로 데이터를 읽어 들인다. 각 인출(FETCH) 후에 CURSOR는 결과 셋에서 다음 행으로 이동합니다.

```
FETCH cursor_name INTO {variable1[,variable2, . . . .]};
```

- FETCH 문장은 현재 행에 대한 정보를 얻어 와서 INTO 뒤에 기술한 변수에 저장한 후 다음 행으로 이동합니다. 얻어진 여러 개의 로우에 대한 결과값을 모두 처리하려면 반복문에 FETCH 문을 기술해야 합니다.

```
LOOP
FETCH C1 INTO v_dept.dno, v_dept.DNAME, v_dept.LOC;
EXIT WHEN C1%NOTFOUND;
END LOOP;
```

FETCH CURSOR

- 커서가 끝에 위치하게 되면 반복문을 탈출해야 합니다.
- 단순 LOOP는 내부에 EXIT WHEN 문장을 포함하고 있다가 EXIT WHEN 다음에 기술한 조건에 만족하면 단순 LOO을 탈출하게 됩니다.
- 반복문을 탈출할 조건으로 “C1%NOTFOUND “를 기술하였습니다.
- NOTFOUND는 커서의 상태를 알려주는 속성 중에 하나인데 커서 영역의 자료가 모두 FETCH됐다면 TRUE를 되돌립니다.
- 커서 C1영역의 자료가 모두 FETCH 되면 반복문을 탈출하게 됩니다.

커서의 상태

- FETCH 문을 설명하면서 커서의 속성 중에 NOTFOUND를 언급하였는데 오라클에서는 이외에도 다양한 커서의 속성을 통해 커서의 상태를 알려주는데 이 속성을 이용해서 커서를 제어해야 합니다.

속성	의미
%NOTFOUND	커서 영역의 자료가 모두 FETCH됐었다면 TRUE
%FOUND	커서 영역에 FETCH 되지 않은 자료가 있다면 TRUE
%ISOPEN	커서가 OPEN된 상태이면 TRUE
%ROWCOUNT	커서가 얻어 온 레코드의 개수

CLOSE CURSOR

- CLOSE문장은 CURSOR를 사용할 수 없게 하고 결과 셋의 정의를 해제합니다.
- SELECT 문장이 다 처리된 완성 후에는 CURSOR를 닫는다. 필요하다면 CURSOR를 다시 열수도 있습니다.

```
CLOSE cursor_name;
```

CURSOR와 FOR LOOP

- CURSOR FOR LOOP는 명시적 CURSOR에서 행을 처리합니다. LOOP에서 각 반복마다 CURSOR를 열고 행을 인출(FETCH)하고 모든 행이 처리되면 자동으로 CURSOR가 CLOSE되므로 사용하기가 편리합니다.

```
FOR record_name IN cursor_name LOOP
statement1;
statement2;
. . . . . .
END LOOP
```

- OPEN ~ FETCH ~ CLOSE가 없이 FOR ~ LOOP ~ END LOOP문을 사용하여 보다 간단하게 커서를 처리해 봅시다.