

PL/SQL (Oracle's Procedural Language extension to SQL) = 오라클에서 SQL을 확장하여 사용하는 프로그래밍 언어. 이름과 같이 절차적 프로그래밍 언어이다.

PL/SQL을 왜 사용할까?

①

1. 대용량 데이터를 연산해야 할 때, WAS등의 서버로 전송해서 처리하려면 네트워크에 부하가 많이 걸릴 수 있다. 이때 프로시저나 함수를 사용하여 데이터를 연산하고 가공한 후에, 최종 결과만 서버에 전송하면 부담을 많이 줄일 수 있다.

2. 로직을 수정하기 위해 서버를 켜다उन 시키지 않아도 된다. 서버에서는 단순히 DB에 프로시저를 호출하여 사용하면 된다.

3. 쿼리문을 직접 노출하지 않는 만큼, SQL injection의 위험성이 줄어든다.

4. 블록 단위로 유연하게 사용할 수 있다.

②

=> 또한 SQL의 다음 단점을 해결 가능하다.

1) 변수가 없다.

2) 한번에 하나의 명령문만 사용 가능하기 때문에 트래픽이 상대적으로 증가한다.

3) 제어문이 사용 불가. (IF, LOOP)

4) 예외처리가 없다. 등등

단점도 존재한다.

1. 유지보수가 힘들다.

2. 대용량 처리가 많을 경우, DB에 부하를 줄 수 있다.

3. Git 같은 형상관리를 사용할 수 없다.

블록 : PL/SQL의 기본 단위. 선언부, 실행부, 예외처리부로 구성

- **이름부** : 블록의 명칭이 옴, 생략시 익명블록 (명칭은 함수, 프로시저 사용가능)
- **선언부** : DECLARE로 시작, 실행부와 예외처리부에서 사용할 변수, 상수, 커서 선언한다. 문장 끝에 반드시 세미콜론(;) 을 찍을 것.
- **실행부** : 실제 로직 처리하는 부분
- **예외처리부** : 로직을 처리하다가 오류가 발생하면 처리할 내용을 기술하는 부분으로 생략이 가능하다.

```
1 DECLARE
2     vi_num INTEGER := 100;
3 BEGIN
4     DBMS_OUTPUT.PUT_LINE(vi_num);
5 END;
```

자바를 해 본 사람이라면, 전반적인 메소드 작성과 비슷하게 이해 해 볼 수있다.

```
1     public void printNumber() {
2
3         int vi_num = 100;
4
5         System.out.println(vi_num);
6
7     }
```

IS와 END는 함수의 스코프이다. { 과 } 로 이해하면 되겠다. 그리고 IS는 변수를 선언한다. (여기는 익명블록을 사용하였으므로 생략)

우선 선언부에 사용할 지역변수를 선언하고 초기화한다. 그리고 실행부에서 작동하는 로직을 코드로 작성한 다음 필요한 경우 try catch로 예외처리를 한다.



PL/SQL 블록의 종류에는 익명블록, 함수, 프로시저가 있다.

PL/SQL의 구성요소

변수명 := 초기값


으로 선언하며 SQL 데이터 타입을 모두 사용할 수 있고, 따로 PL/SQL 데이터 타입을 사용할 수도 있다.

상수

상수명 CONST 데이터타입 := 상수값

변하지 않는 값은 상수로 선언한다. 예약어는 CONST이다.

* PL/SQL에서 사용할 수 있는 SQL문은 DML문이며, DDL문은 사용할 수 없다. (아예 방법이 없는것은 아니지만 일반적인 경우는 아니다)

 SELECT 절 안에서 변수에 값을 넣을때는 반드시 INTO 절을 사용하여야 한다.

```
1 DECLARE
2     vs_emp_name VARCHAR2(80);
3 BEGIN
4     SELECT emp_name
5     INTO vs_emp_name
6     FROM employees
7     WHERE employee_id = 100;
8
9     DBMS_OUTPUT.PUT_LINE('찾아온 이름은... ' || vs_emp_name);
10
11 END;
```

PL/SQL 프로시저가 성공적으로 완료되었습니다.

찾아온 이름은... Steven King

간단하게 구문 테스트를 하려면 SQL*Plus 또는 SQL DEVELOPER에서 쿼리 실행기에서 테스트 할 수 있습니다.

STORED PROCEDURE로 저장해서 사용하려면 DDL 구문을 사용하면 됩니다.

STORED PROCEDURE로 활용하는 방법은 마지막 부분에 설명하겠습니다.

결과값 보여주기 설정

실행하면

"PL/SQL 처리가 정상적으로 완료되었습니다."

라는 메시지만 나오고 출력 값이 안 나오는 경우는

```
SQL>  
SQL> SET SERVEROUTPUT ON  
SQL>
```

SET SERVEROUTPUT ON을 실행해 줍니다.

SQL DEVELOPER에서도 같은 명령어를 주고 마지막에 세미콜론(;)을 붙인 뒤 실행하면 됩니다.

SQL*Plus에서 마지막 END;를 입력한 뒤 엔터를 쳐도 코딩 상태에서 빠져나오지 않습니다.

그때는 "/"를 넣고 엔터를 치면 됩니다.

```
SQL> DECLARE  
2   msg VARCHAR2 (100) := 'Hello World!';  
3   BEGIN  
4       DBMS_OUTPUT.put_line (msg);  
5   EXCEPTION  
6       WHEN OTHERS  
7       THEN  
8           DBMS_OUTPUT.put_line(SQLERRM);  
9   END;  
10  /  
Hello World!  
  
PL/SQL 처리가 정상적으로 완료되었습니다.
```