# 목차

I. 데이터베이스 이해

II. 데이터베이스를 구성하는 객체 이해

III. SQL 기본

IV. SQL 함수

V. 그룹 쿼리와 집합 연산자

VI. 조인(Join)과 서브쿼리(SubQuery)

VII.PL/SQL

# 목차

## VII. PL/SQL


1. PL/SQL 기본 구조
2. PL/SQL 구성 요소
3. PL/SQL 제어문
4. PL/SQL 사용자 정의 함수
5. 프로시저

# 1. PL/SQL 기본 구조

- ✓ PL/SQL ( Oracle's Procedural Language extension to SQL)
- ✓ PL/SQL 은 집합적 언어와 절차적 언어의 특징을 모두 가지고 있다.
- ✓ SQL의 집합적 특징과 프로그래밍 언어처럼 변수에 값을 할당하고 예외처리도 할 수 있으며 특정 기능을 처리하는 함수나 프로시저를 만들 수 있는 기능을 제공한다.
- ✓ 일반 프로그래밍 언어와 다른점은 모든 코드가 DB 내부에 만들어지며 처리됨으로써 수행 속도와 성능 측면에서 큰 장점이 있다.

## -블록 (PL/SQL 소스 프로그램의 기본 단위)

① 이름부 : 블록의 명칭이 오는데, 생략할 때는 익명 블록이 된다.
② 선언부 : DECLARE로 시작되며, 실행부와 에외 처리부에서 사용할 각종 변수, 상수, 커서 등을 선언한다.
문장의 마지막은 세미콜론(;) 을 넣어야 하며 사용할 변수나 상수가 없다면 생략 가능.
① 실행부 : 실제로 로직을 처리하는 부분. (일반 SQL문 , 조건문, 반복문 등)이 올 수 있고, 이러한 문장을 사용해 비즈니스 로직을 구현한다.
(문장 중 DDL 문은 사용할 수 없고 DML 문만 사용가능 문장의 끝은 세미콜론을 붙인다)

# 2. PL/SQL 구성 요소

## – 변수

✓ 변수는 다른 프로그래밍 언어에서 사용하는 변수와 개념이 같으며 선언부에서 변수 선언을 하고 실행부에서 사용한다.

```
변수명 데이터타입 := 초깃값;
vi_ex NUMBER := 100;
```

✓ 변수는 선언과 동시에 초깃값을 할당할 수 있는데, 초깃값을 할당하지 않으면 데이터 타입에 상관없이 그 변수의 초깃값은 NULL이 된다.

## – 상수

✓ 상수는 CONSTANT 란 키워드를 붙어야 하며 선언할 때 반드시 초기화 해야한다.

```
상수명 데이터타입 := 상수값;
vi_ex CONSTANT NUMBER := 3.14;
```

# 2. PL/SQL 구성 요소

## – 연산자

✓ PL/SQL 블록에서는 모든 SQL 연산자를 사용할 수 있으며 우선순위는 다음과 같다.

| 연산자 | | 설명 |
|---|---|---|
| 1 | ** | 제곱 연산자 |
| 2 | +, - | 양수, 음수 식별 연산자 |
| 3 | *, / | 곱셈, 나눗셈 |
| 4 | +, -, \|\| | 덧셈, 뺄셈, 문자열 연결 연산자 |
| 5 | =, <, >, <=, >=, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN | 비교 연산자 |
| 6 | NOT | 논리연산자 |
| 7 | AND | 논리연산자 |
| 8 | OR | 논리 연산자 |

# 2. PL/SQL 구성 요소

## - DML 문

✓ PL/SQL 블록 상에서 사용하는 변수, 상수, 연산자는 부차적인 용도이고
PL/SQL작성 목적은 테이블 상에 있는 데이터를 가공해서 특정 로직에 따라
처리하는 것이며, 따라서 주로 사용되는 것은 SQL문이다.
SQL문 중 DDL은 PL/SQL상에서 직접 쓰지 않고 DML문만 사용한다.

**변수명 테이블명.컬럼명%TYPE; 예제**

```
DECLARE
  vs_emp_name employees.emp_name%TYPE;
  vs_dep_name departments.department_name%TYPE;
BEGIN
  SELECT a.emp_name, b.department_name
    INTO vs_emp_name, vs_dep_name
    FROM employees a,
         departments b
   WHERE a.department_id = b.department_id
     AND a.employee_id = 100;
  DBMS_OUTPUT.PUT_LINE( vs_emp_name || ' - ' || vs_dep_name);
END;
```

# 3. PL/SQL 제어문

✓ PL/SQL 다른 프로그래밍 언어에서 제공하는 다양한 처리문들을 포함하는데 이들을 통칭해 PL/SQL 제어문이라고 한다. 제어문에는 특정 조건에 맞을 때 처리하는 조건문, 반복처리 시 사용하는 반복문, GOTO나 NULL과 같은 순차적 제어문이 있다.

## -IF 문 (특정 조건에 따라 처리하는 것을 조건문이라 한다)

**조건이 1개일 경우**

```
IF 조건 THEN
   조건 처리 ;
END IF;
```

**조건이 2개일 경우**

```
IF 조건 THEN
   조건 처리 2;
ELSE
   조건 처리2
END IF;
```

**조건이 2개일 경우**

```
IF 조건1 THEN
   조건 처리 1;
ELSIF 조건2 THEN
   조건 처리 2;
   …
ELSE
   조건 처리;
END IF;
```

# 3. PL/SQL CASE문

✓ PL/SQL 다른 프로그래밍 언어에서 제공하는 다양한 처리문들을 포함하는데 이들을 통칭해 PL/SQL 제어문이라고 한다. 제어문에는 특정 조건에 맞을 때 처리하는 조건문, 반복처리 시 사용하는 반복문, GOTO나 NULL과 같은 순차적 제어문이 있다.

## -CASE 문

**유형 1**

```
CASE  표현식
      WHEN 결과1 THEN
           처리문1;
      WHEN 결과2 THEN
           처리문2;
      ……..
      ELSE
           기타처리문;
END CASE;
```

**유형 2**

```
CASE WHEN 표현식1 THEN
                처리문1;
     WHEN 표현식2 THEN
                처리문2;
      …..
      ELSE
            기타 처리문;
END CASE;
```

# 3. PL/SQL CASE문

✓ PL/SQL 다른 프로그래밍 언어에서 제공하는 다양한 처리문들을 포함하는데 이들을 통칭해 PL/SQL 제어문이라고 한다. 제어문에는 특정 조건에 맞을 때 처리하는 조건문, 반복처리 시 사용하는 반복문, GOTO나 NULL과 같은 순차적 제어문이 있다.

## -LOOP 문

**유형 1**

```
LOOP
   처리문;
   EXIT [WHEN 조건];
END LOOP;
```

✓ 반복문은 특정 조건을 만족할 때만 루프를 돌며 로직을 처리하는데 LOOP문은 특정조건 없이 실행되며 EXIT를 사용해 루프를 빠져나간다.

# 3. PL/SQL CASE문

✓ PL/SQL 다른 프로그래밍 언어에서 제공하는 다양한 처리문들을 포함하는데 이들을 통칭해 PL/SQL 제어문이라고 한다. 제어문에는 특정 조건에 맞을 때 처리하는 조건문, 반복처리 시 사용하는 반복문, GOTO나 NULL과 같은 순차적 제어문이 있다.

## -WHILE 문

**유형 1**

```
WHILE 조건
LOOP
   처리문;
END LOOP;
```

✓ WHILE 문은 LOOP문과 비슷하지만 조건에 만족 할때만 루프를 돌면서 로직을 처리한다.

# 3. PL/SQL CASE문

✓ PL/SQL 다른 프로그래밍 언어에서 제공하는 다양한 처리문들을 포함하는데 이들을 통칭해 PL/SQL 제어문이라고 한다. 제어문에는 특정 조건에 맞을 때 처리하는 조건문, 반복처리 시 사용하는 반복문, GOTO나 NULL과 같은 순차적 제어문이 있다.

## -FOR 문

**유형 1**

```
FOR 인덱스 IN [REVERSE] 초깃값..최종값
LOOP
   처리문;
END LOOP;
```

✓ 인덱스는 초깃값에서 시작해 최종값까지 루프를 돌며 1씩 증가되는데, 인덱스는 참조는 가능하지만 변경할 수는 없고 참조도 오직 루프 안에서만 가능하다.

✓ REVERSE 를 명시하면 순서가 거꾸로 된다. 즉 최종값부터 시작해 최솟값에 이르기까지 감소하면서 루프를 돈다.

# 3. PL/SQL CASE문

✓ PL/SQL 다른 프로그래밍 언어에서 제공하는 다양한 처리문들을 포함하는데 이들을 통칭해 PL/SQL 제어문이라고 한다. 제어문에는 특정 조건에 맞을 때 처리하는 조건문, 반복처리 시 사용하는 반복문, GOTO나 NULL과 같은 순차적 제어문이 있다.

## -CONTINUE문

✓ CONTINUE 문은 FOR 나 WHILE 같은 반복문은 아니지만, 반복문 내에서 특정 조건에 부합 할 때 처리 로직을 건너뛰고 상단의 루프 조건으로 건너가 루프를 계속 수행할 때 사용한다.
EXIT 는 루프를 완전히 빠져 나오는데 반해, CONTINUE는 제어 범위가 조건절로 넘어간다.

## -GOTO문

✓ PL/SQL 코드 상에서 GOTO문을 만나면 GOTO 문이 지정하는 라벨로 제어가 넘어간다.

## -NULL문

✓ NULL문은 아무것도 처리하지 않는 문장이다.

# 4. PL/SQL 사용자 정의 함수

✓ 익명 블록은 한 번 사용하고 나면 없어져 버리는 휘발성 블록이지만,
서브 프로그램인 함수나 프로시저는 컴파일을 거쳐 데이터베이스 내에 저장되어 재사용이 가능하다.

## –함수 생성

```
CREATE OR REPLACE FUNCTION 함수 이름 (매개변수1, 매개변수2, ....)
RETURN 데이터타입;
IS [AS]
  변수, 상수 등 선언
BEGIN
  실행부

  RETURN 반환값;
[EXCEPTION 예외 처리부]
END [함수 이름];
```

# 4. PL/SQL 사용자 정의 함수

① CREATE OR REPLACE FUNCTION : 구문을 사용해 함수를 생성한다.
② 매개변수 : 함수로 전달되는 매개변수로, "매개변수명 데이터 타입" 형태로 명시한다.(매개변수는 생략가능)
③ RETURN 데이터 타입 : 함수가 반환할 데이터 타입을 지정한다.
④ RETURN 반환값 : 매개변수를 받아 특정 연산을 수행한 후 반환할 값을 명시한다.

## -함수 호출

**<매개변수가 없는 함수 호출>**
**함수명 혹은 함수명()**

**<매개변수가 있는 함수 호출>**
**함수명(매개변수1, 매개변수2, ….)**

✓ SQL함수와 마찬가지로PL/SQL 함수는 SELECT 문에서 사용할 수 있다.

# 5. 프로시저

- ✓ 함수와 프로시저의 차이는 함수는 클라이언트에서 실행되며 리턴값이 필수 이지만 프로시저는 서버에서 실행되며 리턴값이 없이 또는 여러 개가 가능하다.
- ✓ 테이블에서 데이터를 추출해 입맛에 맞게 조작하고 그 결과를 다른 테이블에 다시 저장하거나 갱신하는 일련의 처리를 할 때 주로 프로시저를 사용한다.
- ✓ 함수나 프로시저는 모두 DB에 저장된 객체이므로 프로시저를 스토어드(Stored, 저장된) 프로시저라고 부르기도 한다.(함수도 스토어드 함수라고도 한다.)

## -프로시저 생성

```
CREATE OR REPLACE PROCEDURE 프로지서 이름
    ( 매개변수명 1 [IN | OUT | IN OUT ] 데이터타입 [:= 디폴트 값],
      매개변수명 2 [IN | OUT | IN OUT ] 데이터타입 [:= 디폴트 값],
       ……
     )
IS [AS]
   변수, 상수 등 선언
BEGIN
   실행부
[EXCEPTION 예외 처리부]
END [프로시저 이름];
```

# 5. 프로시저

## -프로시저 실행

- ✓ 함수는 반환 값을 받으므로 실행할 때 '호출' 이라고 명명하지만 프로시저는 '실행'한다고 표현한다.
- ✓ 프로시저는 함수처럼 SELECT 절에는 사용할 수 없다.

**프로시저 실행 1**

```
EXEC 혹은 EXECUTE 프로시저명(매개변수1 값, 매개변수2 값, ...);
```

**프로시저 실행 2**

```
EXEC 혹은 EXECUTE 프로시저명(매개변수1 => 매개변수1 값,
                              매개변수2 => 매개변수2 값, ...);
```

- ✓ 프로시저의 매개변수가 많으면 실행할 때 매개변수 값의 개수나 순서를 혼동할 소지가 높기 때문에 이런 경우에 실행2 방법으로 값을 매핑해 실행한다.

# 5. 프로시저

## -매개변수 디폴트 값 설정

✓ 프로시저를 실행할 때는 반드시 매개변수의 개수에 맞춰 값을 전달해 실행하여야 하는데 디폴트 값을 설정하면 실행할 때 해당 매개변수를 전달 하지 않더라도 오류가 나지 않는다.

**(예제) 생성**

```
CREATE OR REPLACE PROCEDURE my_new_job_proc
( p_job_id IN JOBS.JOB_ID%TYPE ,
  p_job_title IN JOBS.JOB_TITLE%TYPE ,
  p_min_sal IN JOBS.JOBS.MIN_SALARY%TYPE:=10 ,       --디폴트값 설정
  p_max_sal IN JOBS.JOBS.MAX_SALARY%TYPE:=100)      --디폴트값 설정
IS
..
..
```

**예제 실행**

```
EXECUTE my_new_job_proc ('SM_JOB1', 'Sample JOB1');
```

# 5. 프로시저

## -OUT, IN OUT 매개변수

✓ 프로시저의 값 반환 방법

**(예제) 생성**

```
CREATE OR REPLACE PROCEDURE my_new_job_proc
( p_job_id IN JOBS.JOB_ID%TYPE ,
  p_job_title IN JOBS.JOB_TITLE%TYPE ,
  p_min_sal IN JOBS.JOBS.MIN_SALARY%TYPE:=10 ,       --디폴트값 설정
  p_max_sal IN JOBS.JOBS.MAX_SALARY%TYPE:=100)       --디폴트값 설정
IS
..
..
```

**예제 실행**

```
EXECUTE my_new_job_proc ('SM_JOB1', 'Sample JOB1');
```

# 감사합니다!

이 문서는 나눔글꼴로 작성되었습니다. 설치하기