

Creating Packages

1. 패키지(Package) 개요

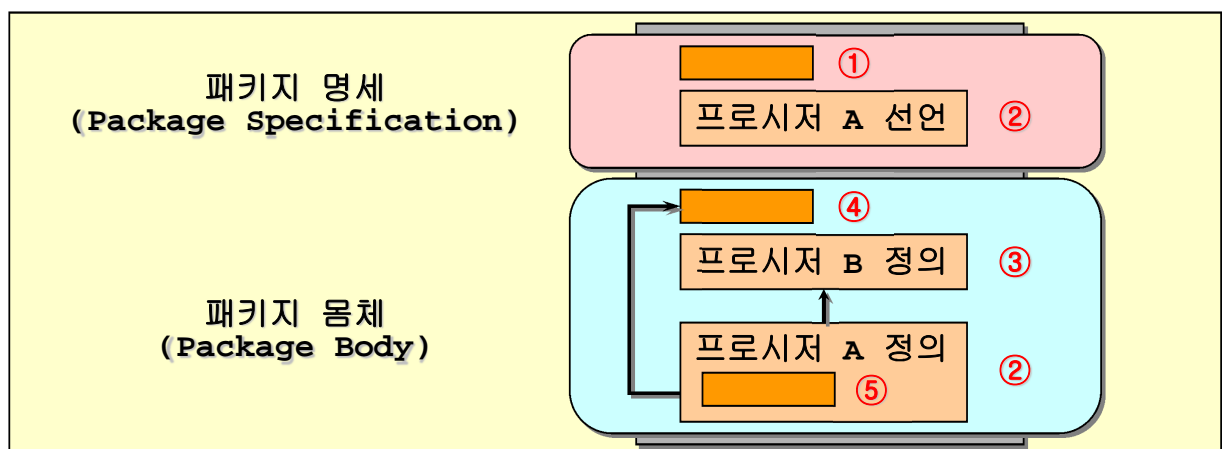
1-1. 패키지 개념

- ☑ 논리적으로 관련된 PL/SQL 유형, 항목, 서브 프로그램을 그룹화
- ☑ 패키지에는 데이터베이스에 별도로 저장된 명세(Specification), 몸체(Body)가 있음
 - 명세는 응용 프로그램에 대한 인터페이스이며 사용 가능한 유형, 변수, 상수, 예외사항, 커서, 서브 프로그램을 선언
 - 몸체는 커서 및 서브 프로그램을 완전히 정의하고 명세 부분을 구현
- ☑ 호출 또는 중첩하거나 매개변수를 지급할 수 없음
- ☑ 한 번에 여러 객체를 메모리로 읽어 들일 수 있음

1-2. 패키지 장점

- ☐ 모듈화(Modularity)
 - 관련 생성자를 캡슐화
- ☐ 간단한 응용 프로그램 설계
 - 명세 및 몸체를 따로 코딩 및 컴파일
- ☐ 정보 숨김(Hiding)
 - 전용(Private) 생성자는 숨겨지며 액세스할 수 없음
 - 몸체의 모든 코딩은 숨겨짐
- ☐ 추가 기능
 - 변수 및 커서의 지속성
- ☐ 성능 향상
 - 패키지를 처음 참조할 때 전체 패키지가 메모리에 로드(road)
 - 모든 사용자를 위한 복사본은 메모리에 하나만 존재
 - 종속성 계층이 단순
- ☐ 오버로드
 - 동일한 이름의 여러 서브 프로그램

1-3. 패키지 구성 요소 (1)



- ① 공용(Public) 변수
- ② 공용(Public) 프로시저
- ③ 전용(Private) 프로시저
- ④ 전역(Global) 변수
- ⑤ 지역(Local) 변수

1-4. 패키지 구성 요소 (2)

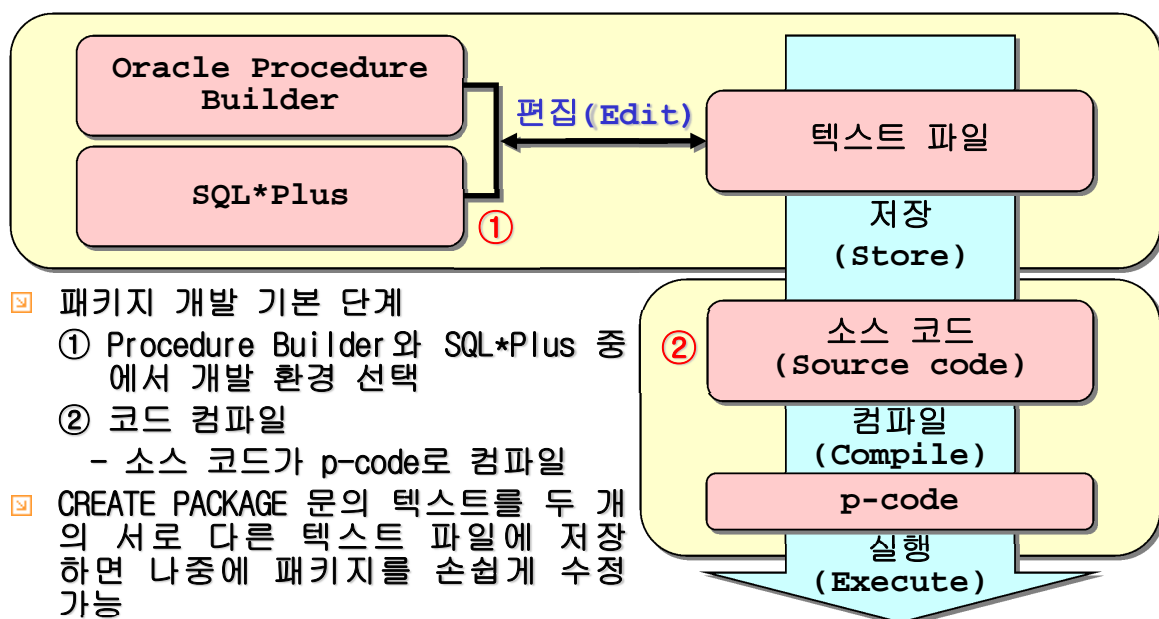
☐ 생성자(Construct)의 범위

범 위	설 명	패키지 내 배치
공 용 (Public)	모든 Oracle Server에서 참조 가능	패키지 명세에서 선언 패키지 몸체에서 정의
전 용 (Private)	동일한 패키지의 일부인 다른 생성자만 참조 가능	패키지 몸체에서 선언 및 정의

☐ 생성자(Construct)의 가시성(Visibility)

가시성	설 명
지 역 (Local)	다른 서브 프로그램에서 정의하고 외부 사용자가 볼 수 없는 변수 또는 서브 프로그램
전 역 (Global)	패키지 외부에서 참조 및 변경 가능하며 외부 사용자가 볼 수 있는 변수 또는 서브 프로그램

2. 패키지 개발



☐ 패키지 개발 기본 단계

- ① Procedure Builder와 SQL*Plus 중
에서 개발 환경 선택
- ② 코드 컴파일
 - 소스 코드가 p-code로 컴파일

☐ CREATE PACKAGE 문의 텍스트를 두 개 의 서로 다른 텍스트 파일에 저장 하면 나중에 패키지를 손쉽게 수정 가능

- ☐ 패키지 명세는 패키지 몸체 없이 존재 가능
- ☐ 패키지 몸체는 패키지 명세 없이 존재 불가능

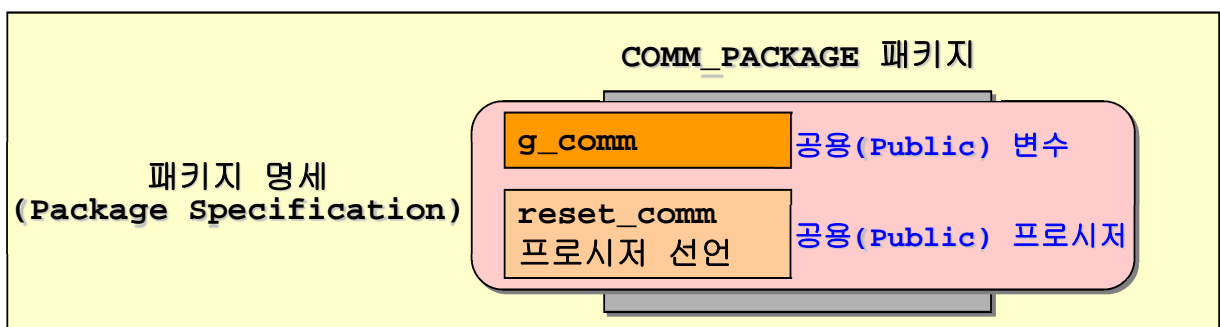
3. 패키지 명세(Specification)

3-1. 개요

```
CREATE [OR REPLACE] PACKAGE package_name
IS | AS
    public type and item declarations
    subprogram specifications
END package_name;
```

- ☐ 패키지를 작성하려면 패키지 명세에서 모든 공용(Public) 생성자 선언
- ☐ 패키지 명세가 이미 있는 경우에는 REPLACE 옵션 지정
- ☐ 필요한 경우 선언 부분에 있는 변수를 상수 값 또는 수식으로 초기화
- ☐ 변수를 초기화하지 않으면 암시적으로 NULL로 초기화

3-2. 공용(Public) 생성자 선언



- ☐ 패키지 명세에서 공용 변수, 공용 프로시저 및 공용 함수 선언
- ☐ 공용 프로시저 또는 함수는 동일한 패키지 또는 패키지 외부의 다른 생성자에 의해 반복적으로 호출될 수 있는 루틴

3-3. 패키지 명세 작성

```
SQL> CREATE OR REPLACE PACKAGE comm_package
2  IS
3      g_comm NUMBER := 10; -- 10으로 초기화
3      PROCEDURE reset_comm
4          (v_comm IN NUMBER);
5  END comm_package;
```

3-4. 공용 변수 및 공용 프로시저 선언

```
SQL> EXECUTE comm_package.g_comm := 5
SQL> EXECUTE comm_package.reset_comm(8)
```

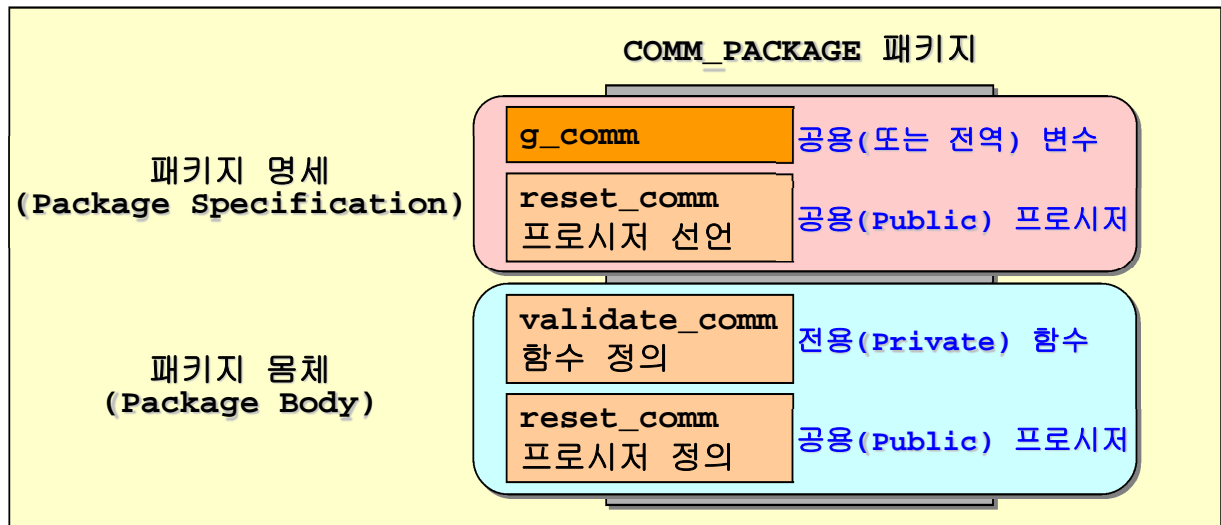
4. 패키지 몸체(Body)

4-1. 개요

```
CREATE [OR REPLACE] PACKAGE BODY package_name
IS | AS
    private type and item declarations
    subprogram bodies
END package_name;
```

- ☐ 패키지를 작성하려면 패키지 명세에서 모든 공용(Public) 및 전용(Private) 생성자 정의
- ☐ 패키지 몸체가 이미 있는 경우에는 REPLACE 옵션 지정
- ☐ 패키지 몸체에서 서브 프로그램이 정의되는 순서 중요
 - 다른 변수 또는 서브 프로그램이 변수를 참조하기 전에 변수 선언
 - 다른 서브 프로그램에서 전용(Private) 서브 프로그램을 호출하기 전에 이를 선언 또는 정의
 - 일반적으로 먼저 패키지 몸체에서 전용(Private) 변수 및 서브 프로그램을 모두 정의한 다음 마지막으로 공용(Public) 서브 프로그램 정의

4-2. 공용(Public) 및 전용(Private) 생성자



- ☐ 공용 프로시저 및 함수를 모듈화하고 코드를 확실하게 구별할 수 있도록 전용 프로시저 또는 함수를 정의할 수 있음

4-3. 패키지 몸체 작성 (1)

```
SQL> CREATE OR REPLACE PACKAGE BODY comm_package
2  IS
3      FUNCTION validate_comm
4          (v_comm IN NUMBER)
5          RETURN BOOLEAN
6  IS
7      v_max_comm  NUMBER;
8  BEGIN
9      SELECT max(comm)
10     INTO   v_max_comm
11     FROM   emp;
12     IF     v_comm > v_max_comm THEN
13         RETURN(FALSE);
14     ELSE
15         RETURN(TRUE);
16     END IF;
17 END validate_comm;
```

4-4. 패키지 몸체 작성 (2)

```

18     PROCEDURE reset_comm
19         (v_comm IN NUMBER)
20     IS
21         v_valid BOOLEAN;
22     BEGIN
23         v_valid := validate_comm(v_comm);
24         IF v_valid = TRUE THEN
25             g_comm := v_comm;
26         ELSE
27             RAISE_APPLICATION_ERROR
28                 (-20210, 'Invalid commission');
29         END IF;
30     END reset_comm;
31 END comm_package;

```

SQL 실무

5. 패키지 활용

5-1. 패키지 생성자 호출 (1)

```

CREATE OR REPLACE PACKAGE BODY comm_package
    . . .
    PROCEDURE reset_comm
        (v_comm IN NUMBER)
    IS
        v_valid BOOLEAN;
    BEGIN
        v_valid := validate_comm(v_comm);
        IF v_valid = TRUE THEN
            g_comm := v_comm;
        ELSE
            RAISE_APPLICATION_ERROR(-20210, 'Invalid comm');
        END IF;
    END reset_comm;
END comm_package;

```

- ☞ 데이터베이스에 패키지가 저장되면 생성자의 전용 또는 공용 여부에 따라 패키지 내부 또는 외부에서 패키지 생성자 호출 가능

SQL 실무

5-2. 패키지 생성자 호출 (2)

```
SQL> EXECUTE comm_package.reset_comm(1500)
```

- SQL*Plus에서 패키지 프로시저 호출

```
SQL> EXECUTE scott.comm_package.reset_comm(1500)
```

- 다른 스키마에 있는 패키지 프로시저 호출

```
SQL> EXECUTE comm_package.reset_comm@ny(1500)
```

- 원격 데이터베이스에 있는 패키지 프로시저 호출

5-3. 패키지 몸체 없는 선언

```
CREATE OR REPLACE PACKAGE global_consts
IS
    mile_2_kilo    CONSTANT NUMBER := 1.6093;
    kilo_2_mile    CONSTANT NUMBER := 0.6214;
    yard_2_meter   CONSTANT NUMBER := 0.9144;
    meter_2_yard   CONSTANT NUMBER := 1.0936;
END global_consts;
/
```

```
SQL> SET SERVEROUTPUT ON
SQL> EXECUTE DBMS_OUTPUT.PUT_LINE('20 miles = ' || -
    2          20*global_consts.mile_2_kilo || ' km')
```


5-4. 독립형 프로시저에서 공용 변수 참조

```
CREATE OR REPLACE PROCEDURE meter_to_yard
  (p_meter  IN  NUMBER,
   p_yard   OUT NUMBER)
IS
BEGIN
  p_yard := p_meter * global_consts.meter_2_yard;
END meter_to_yard;
```

```
SQL> VARIABLE yard NUMBER
SQL> EXECUTE meter_to_yard (1, :yard)
SQL> PRINT yard
```

6. 패키지 삭제

- ❑ 패키지 명세 및 몸체 제거

```
DROP PACKAGE package_name
```

- ❑ 패키지 몸체 제거

```
DROP PACKAGE BODY package_name
```