

Creating Procedures



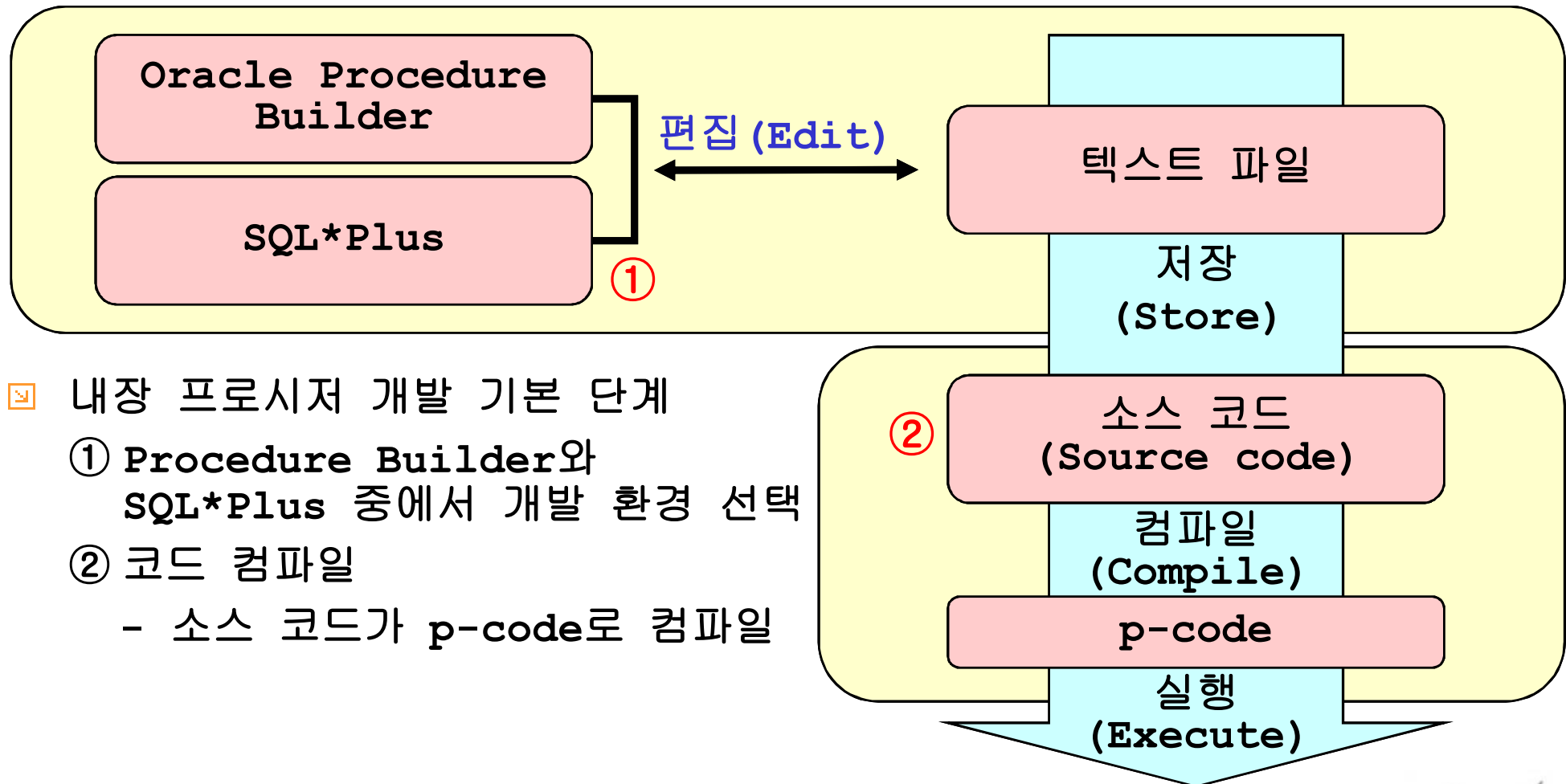
1. 내장 프로시저 개요

```
CREATE [OR REPLACE] PROCEDURE procedure_name
    (parameter1 [mode1] datatype1,
     parameter2 [mode2] datatype2,
     . . .)
IS|AS
PL/SQL Block;
```

- ☐ **내장 프로시저**는 매개변수 (Parameter) 를 사용하여 호출할 수 있는 **명명된 (named) PL/SQL 블록**
- ☐ 내장 프로시저에는 헤더 (Header), 선언 부분 (DECLARE), 실행 부분 (BEGIN), 예외 처리 부분 (EXCEPTION) 이 있음
- ☐ 재사용 및 유지 관리 기능 향상
 - 여러 응용 프로그램에서 사용 가능
 - 정의를 변경하면 프로시저에만 적용되므로 유지 관리가 용이

2. 내장 프로시저 개발 및 작성

2-1. 내장 프로시저 개발



☐ 내장 프로시저 개발 기본 단계

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

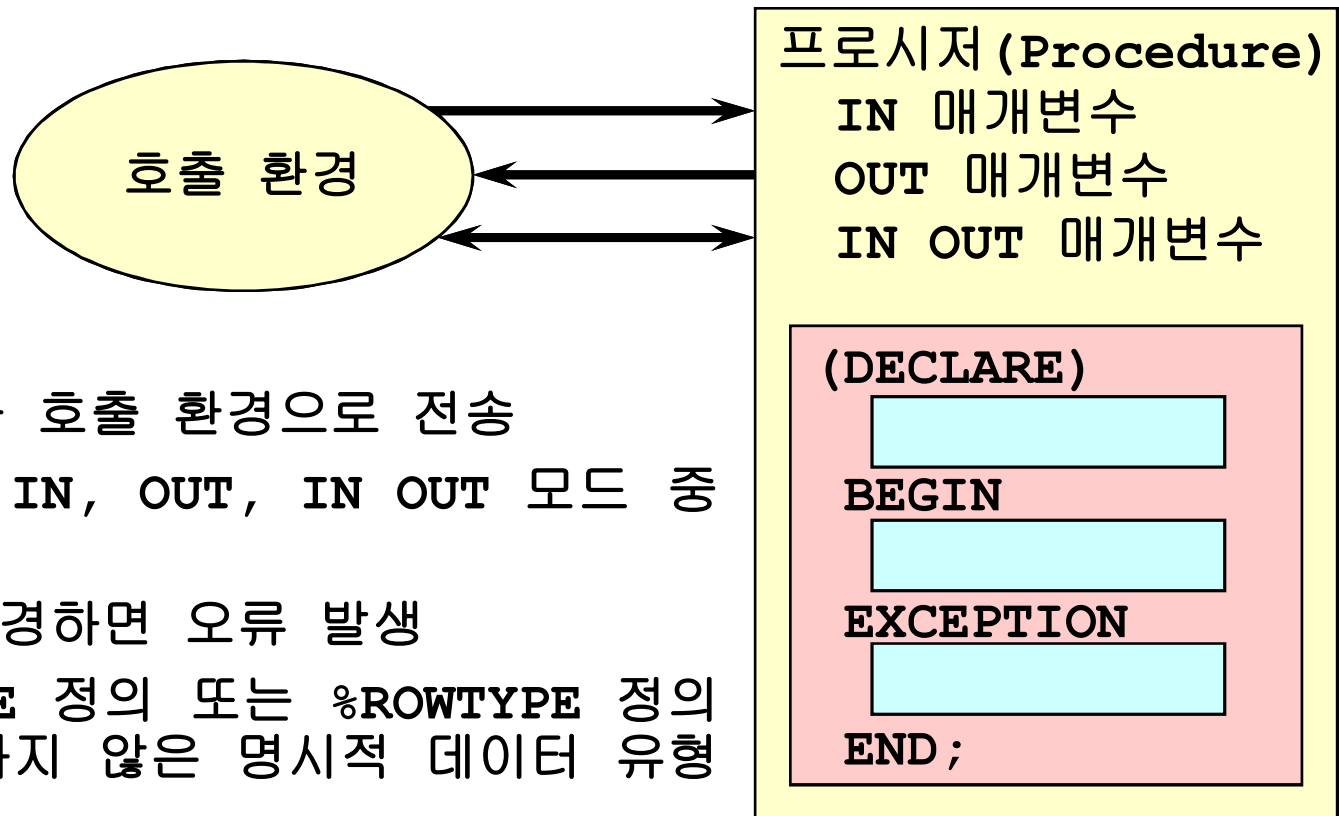
2-2. 내장 프로시저 작성

- ☐ 시스템 편집기 (SQL*Plus) 에서 CREATE PROCEDURE 문의 텍스트를 입력
- ☐ 스크립트 파일을 실행하여 프로시저 컴파일
- ☐ SQL*Plus 명령어 중 **SHOW ERRORS**를 사용하여 컴파일 오류 확인
- ☐ 성공적으로 컴파일된 프로시저는 실행 가능



3. 프로시저 매개변수 모드

3-1. 개요 (1)



- ☑ 매개변수를 통해 값을 호출 환경으로 전송
- ☑ 각 매개변수에 대해 IN, OUT, IN OUT 모드 중 하나 선택
- ☑ IN 매개변수 값을 변경하면 오류 발생
- ☑ 데이터 유형은 %TYPE 정의 또는 %ROWTYPE 정의 이거나 크기를 지정하지 않은 명시적 데이터 유형 이어야 함

mbg

3-2. 개요 (2)

IN	OUT	IN OUT
기본값 (default)	지정해야 함	지정해야 함
값을 서브 프로그램에 전달	값을 호출 환경으로 반환	값을 서브 프로그램에 전달하고 호출 환경으로 반환
형식 매개변수가 상수로 작용	초기화되지 않은 변수	초기화된 변수
실제 매개변수는 리터럴, 표현식, 상수 또는 초기화된 변수	변수여야 함	변수여야 함

3-3. IN 매개변수

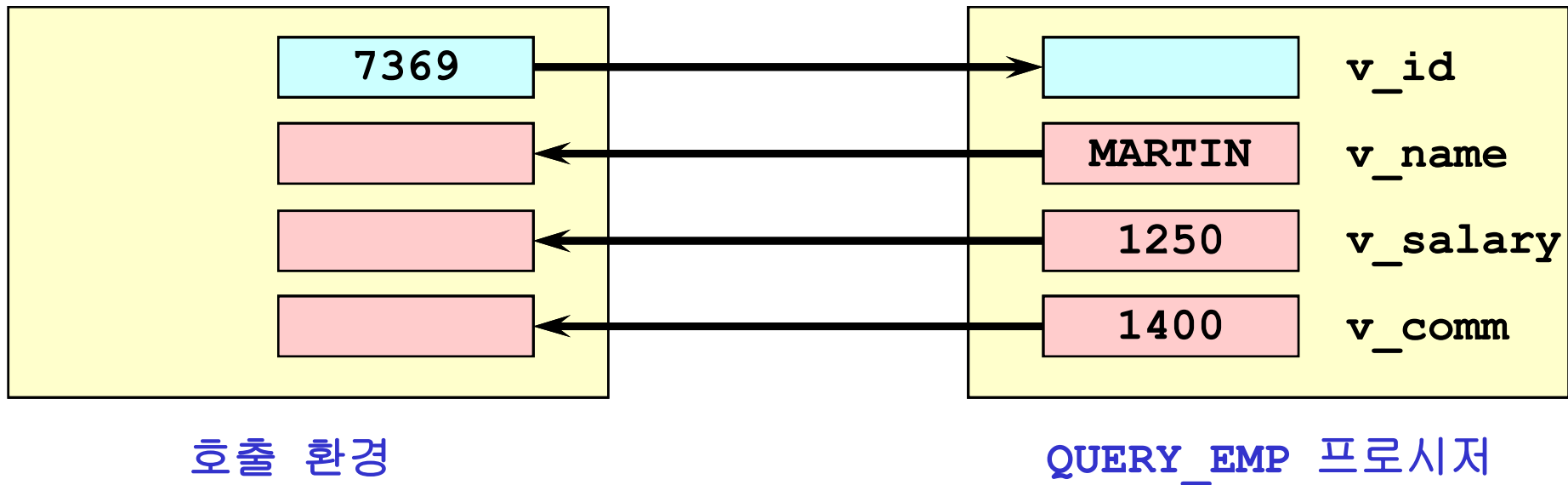


```
SQL> CREATE OR REPLACE PROCEDURE raise_salary
  2  (v_id IN emp.empno%TYPE)
  3  IS
  4  BEGIN
  5      UPDATE emp
  6      SET      sal = sal * 1.10
  7      WHERE   empno = v_id;
  8  END raise_salary;
  9  /
```

Procedure created.

```
SQL> EXECUTE raise_salary (7369)
PL/SQL procedure successfully completed.
```

3-4. OUT 매개변수 (1)



3-5. OUT 매개변수 (2)

```
SQL> CREATE OR REPLACE PROCEDURE query_emp
  2  (v_id      IN emp.empno%TYPE,
  3   v_name    OUT emp.ename%TYPE,
  4   v_salary  OUT emp.sal%TYPE,
  5   v_comm    OUT emp.comm%TYPE)
  6  IS
  7  BEGIN
  8      SELECT  ename, sal, comm
  9      INTO    v_name, v_salary, v_comm
 10      FROM    emp
 11      WHERE   empno = v_id;
 12  END query_emp;
 13  /
```

Procedure created.

3-6. OUT 매개변수 (3)

```
SQL> VARIABLE g_name          VARCHAR2 (15)
SQL> VARIABLE g_salary        NUMBER
SQL> VARIABLE g_comm          NUMBER
```

```
SQL> EXECUTE query_emp (7654, :g_name, :g_salary, g_comm)
PL/SQL procedure successfully completed.
```

```
SQL> PRINT g_name
G_NAME
-----
MARTIN
```



3-7. IN OUT 매개변수 (1)

'(053) 940-5000' ←→ '(053) 940-5000' v_phone_no

```
SQL> CREATE OR REPLACE PROCEDURE format_phone
  2  (v_phone_no IN OUT VARCHAR2)
  3  IS
  4  BEGIN
  5      v_phone_no := '(' || SUBSTR(v_phone_no, 1, 3) ||
  6                  ')' || SUBSTR(v_phone_no, 4, 3) ||
  7                  '-' || SUBSTR(v_phone_no, 7);
  8  END format_phone;
  9  /
```

Procedure created.

3-8. IN OUT 매개변수 (2)

```
SQL> VARIABLE g_phone_no    VARCHAR2(15)
```

```
SQL> BEGIN
  2      :g_phone_no := '0539400000' ;
  3  END;
PL/SQL procedure successfully completed.
```

```
SQL> EXECUTE format_phone (:g_phone_no)
PL/SQL procedure successfully completed.
```

```
SQL> PRINT g_phone_no
G_PHONE_NO
-----
(053) 940-0000
```

4. 프로시저 활용

4-1. 익명 PL/SQL 블록에서 프로시저 호출

```
DECLARE
    v_id NUMBER := 7900;
BEGIN
    raise_salary(v_id); -- 프로시저 호출
    COMMIT;
    ...
END;
```

- ☐ PL/SQL을 지원하는 모든 도구 (tool) 또는 언어 (language) 에서 프로시저 호출 가능

4-2. 내장 프로시저에서 프로시저 호출

```
SQL> CREATE OR REPLACE PROCEDURE process_emps
  2  IS
  3      CURSOR emp_cursor IS
  4          SELECT empno
  5          FROM    emp;
  6  BEGIN
  7      FOR emp_rec IN emp_cursor LOOP
  8          raise_salary(emp_rec.empno);  -- 프로시저 호출
  9      END LOOP;
 10      COMMIT;
 11  END process_emps;
 12  /
Procedure created.
```

5. 프로시저 제거

```
DROP PROCEDURE procedure_name;
```

```
SQL> DROP PROCEDURE raise_salary;  
Procedure dropped.
```

