Creating Functions

1. 내장 함수 개요

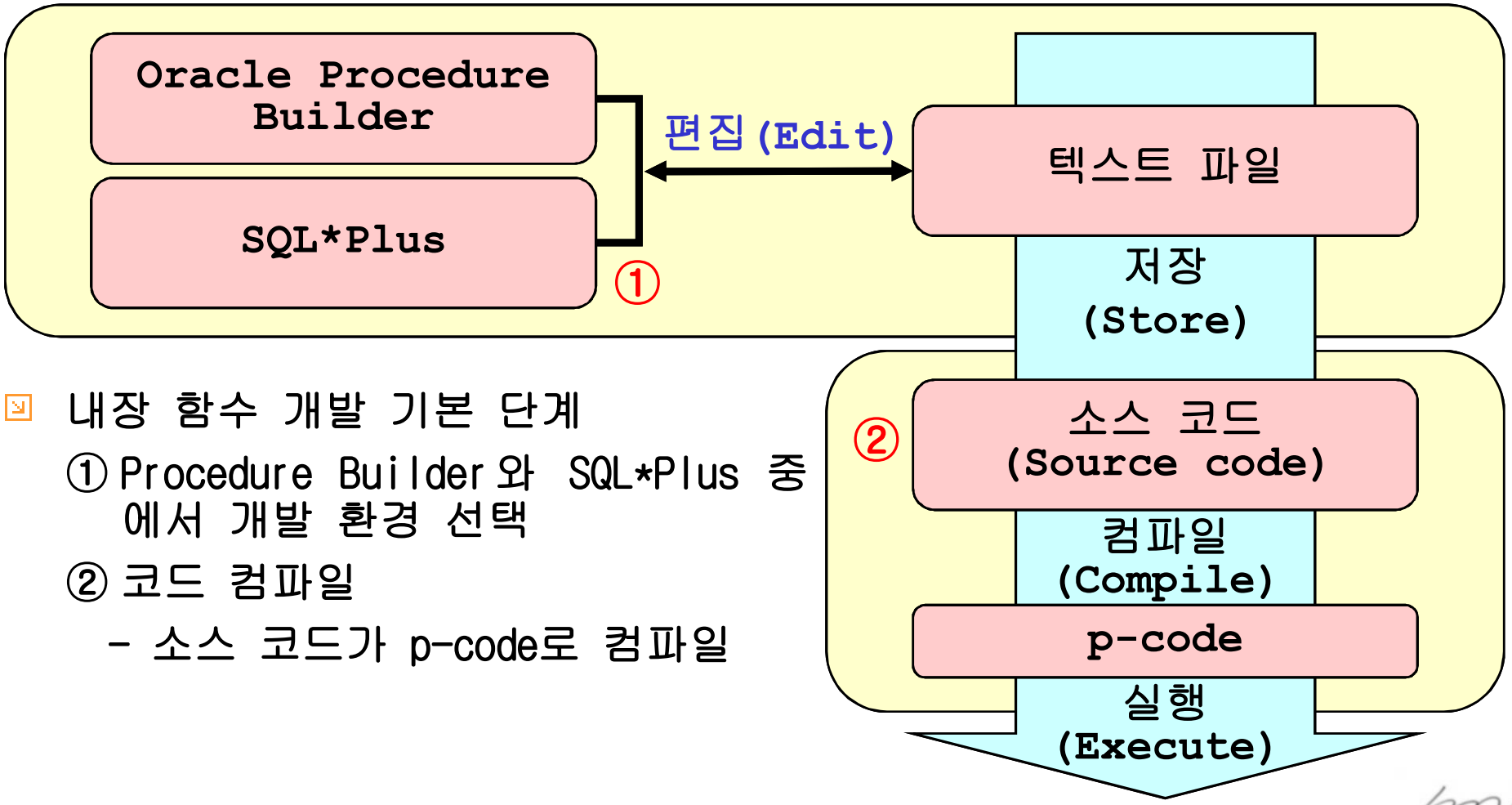
IN MODE만 존재함. (RETURN있어서 OUT필요 X)

```
CREATE [OR REPLACE] FUNCTION function_name
    (argument1 [mode1] datatype1,
     argument2 [mode2] datatype2,
     • • •)
RETURN datatype
IS|AS
PL/SQL Block;
```

- 내장 함수는 매개변수(Parameter)를 사용하여 호출할 수 있는 명명된(named) PL/SQL 블록
- 일반적으로 값을 계산할 때 사용
- 함수와 프로시저는 함수가 호출 환경으로 값을 반환해야 하는 점을 제외하면 구조가 동일
- 함수 헤더(header)에는 RETURN 절이 있어야 하며, 실행 부분에는 RETURN 문이 하나 이상 있어야 함
- 재사용 및 유지 관리 기능 향상
  - 여러 응용 프로그램에서 사용 가능
  - 정의를 변경하면 함수에만 적용되므로 유지 관리가 용이

2. 내장 함수 개발 및 작성

2-1. 내장 함수 개발

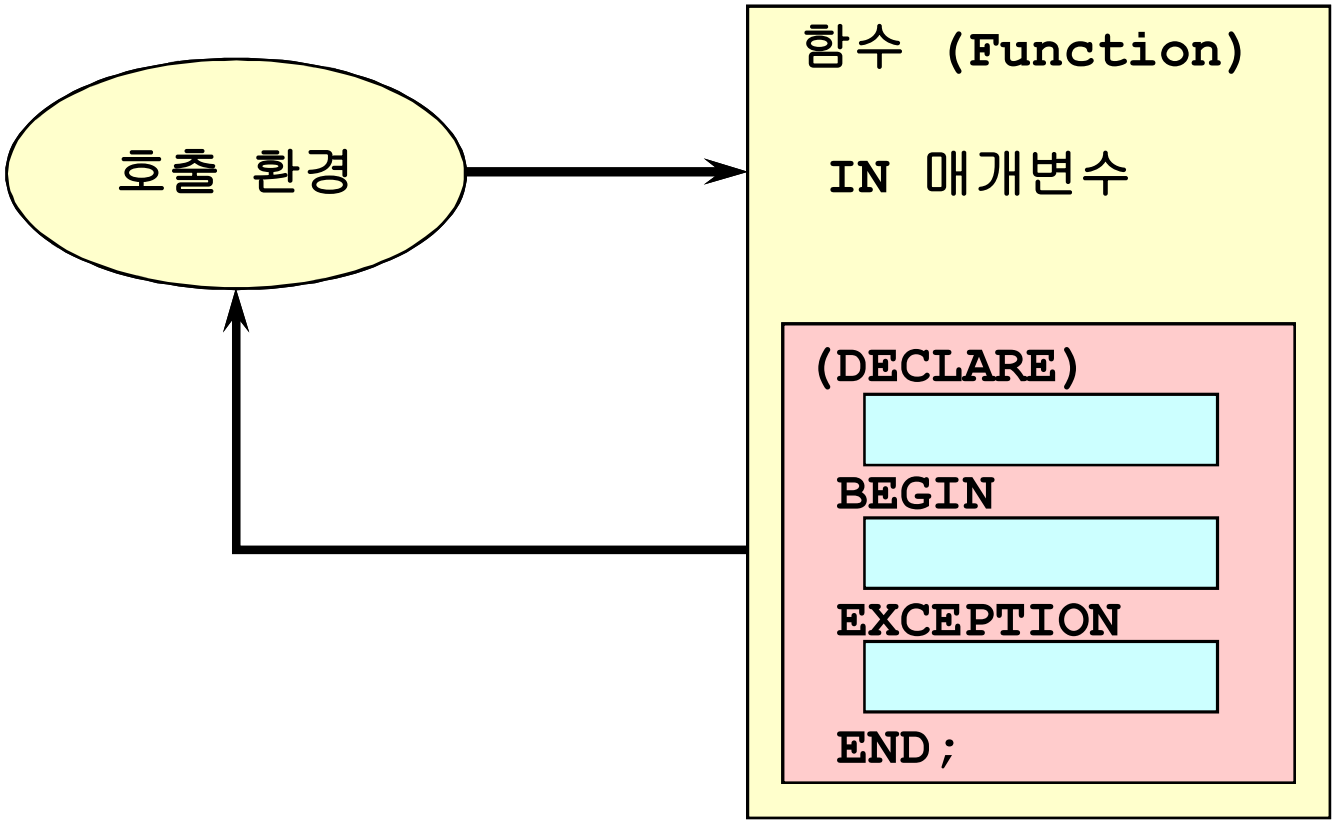


- 내장 함수 개발 기본 단계
  - ① Procedure Builder와 SQL*Plus 중에서 개발 환경 선택
  - ② 코드 컴파일
    - 소스 코드가 p-code로 컴파일

2-2. 내장 함수 작성

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

3. 함수 매개변수 모드



- 값을 계산할 때 작성, 이 값은 호출 환경으로 반드시 반환
- 함수는 호출 환경에서 전송하는 매개변수를 0개 이상 포함할 수 있지만, 값을 하나만 반환해야 하므로 함수에 OUT, IN OUT 매개변수는 선언하지 않음

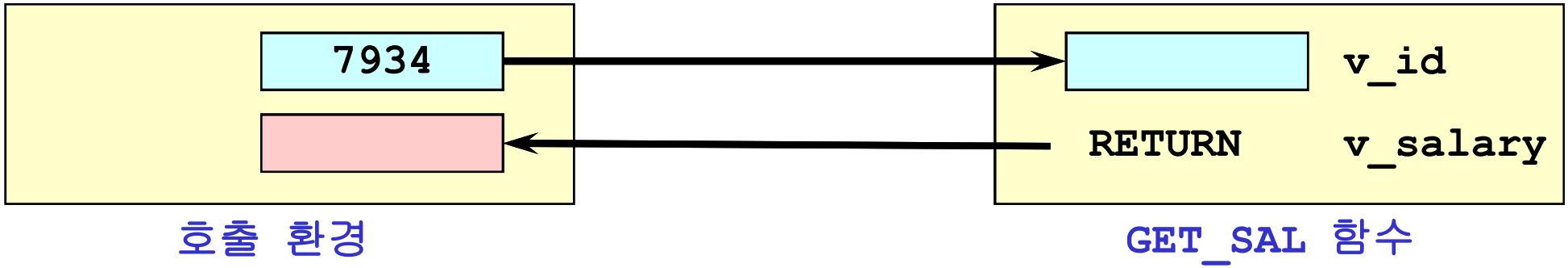
4. 함수 활용

4-1. 함수 작성

```
SQL> CREATE OR REPLACE FUNCTION get_sal
  2      (v_id IN emp.empno%TYPE)
  3       RETURN NUMBER
  4  IS
  5       v_salary emp.sal%TYPE := 0;
  6  BEGIN
  7       SELECT sal
  8       INTO   v_salary
  9       FROM   emp
 10       WHERE  empno = v_id;
 11       RETURN (v_salary);
 12  END get_sal;
 13  /
```

함수는 DBMS_OUTPUT.PUT_LINE 안 씀!
그냥 메세지도 RETURN해주면 된다.

4-2. 함수 실행




```
SQL> VARIABLE g_salary     NUMBER
```

```
SQL> EXECUTE :g_salary := get_sal (7934)
PL/SQL procedure successfully completed.
```

```
SQL> PRINT g_salary
          G_SALARY
-----------------
              1300
```

5. SQL 표현식에서 함수 사용

5-1. 내장 함수 사용의 장점

- 작업이 너무 복잡하고 다루기 힘들거나 SQL을 사용할 수 없는 경우 SQL을 확장
- 응용프로그램의 데이터 필터링과 반대로 WHERE 절에서 사용자가 정의한 함수로 데이터를 필터링하면 효과적
- 문자열 조작 가능

5-2. 내장 함수 호출 위치

- SELECT 명령의 SELECT 절
- WHERE 및 HAVING 절의 조건
- CONNECT BY, START WITH (계층형 쿼리), ORDER BY, GROUP BY 절
- INSERT 명령의 VALUES 절
- UPDATE 명령의 SET 절

5-3. 내장 함수 호출 제한사항

- 사용자가 정의한 함수는 내장 함수
- 사용자가 정의한 함수는 그룹 함수가 아닌 단일 행(Single-row) 함수
- 사용자가 정의한 함수에는 OUT, IN OUT이 아닌 IN 매개변수만 사용
- 데이터 유형은 유효한 SQL 데이터 유형인 CHAR, DATE, NUMBER
- BOOLEAN, RECORE, TABLE과 같은 PL/SQL형식의 데이터 유형 사용 불가
- INSERT, UPDATE, DELETE 명령 사용 불가
- 위의 제한사항을 위반하는 서브 프로그램은 호출할 수 없음

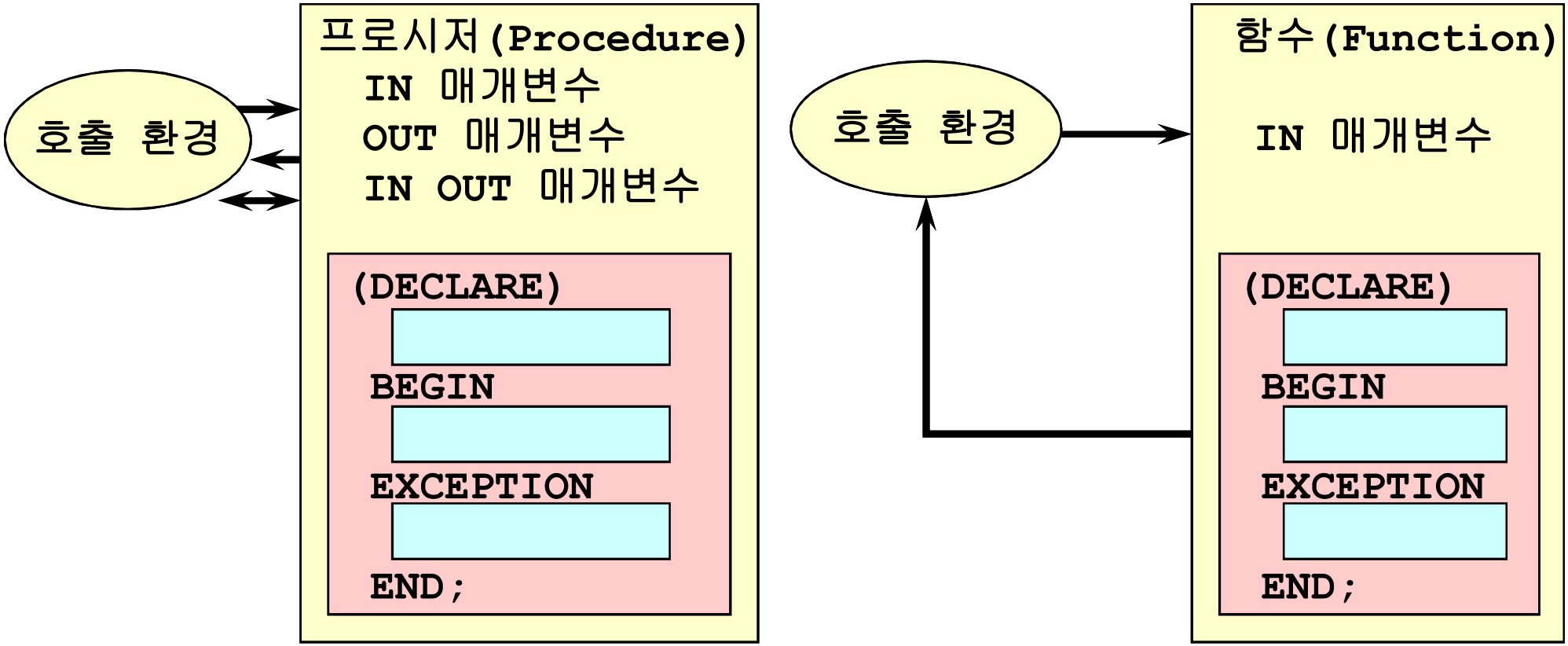
6. 함수 제거

```
DROP FUNCTION function_name;
```

```
SQL> DROP FUNCTION get_sal;
Function dropped.
```

7. 프로시저와 함수 비교

7-1. 차이점



7-1. 차이점

프로시저	함수
PL/SQL 문으로 실행	표현식의 일부로 호출
RETURN 데이터 유형을 포함하지 않음	RETURN 데이터 유형을 포함
값을 반환하지 않거나 하나 이상의 값을 반환할 수 있음	값을 하나만 반환

7-2. 장점

- **성능 향상**
  - 공유 SQL 영역을 활용하여 여러 사용자가 사용하는 구문을 재분석하지 않도록 함
  - 컴파일 중에 PL/SQL의 구문을 분석하여 실행 중에 분석되지 않도록 함
  - 명령을 묶어 데이터베이스에 대한 호출 수를 줄이고 네트워크 통신량을 감소 시킴
- **유지 관리 향상**
  - 다른 사용자를 방해하지 않고도 온라인으로 루틴을 수정
  - 루틴 하나를 수정하여 여러 응용 프로그램에 영향을 줌
  - 루틴 하나를 수정하여 여러 중복 테스트를 제거
- **데이터 보안 및 무결성 향상**
  - 보안 권한으로 권한이 없는 사용자의 데이터베이스 객체에 대한 간접 액세스 제어
  - 단일 경로를 통해 관련 테이블에 대한 작업을 한 곳으로 모아 관련 작업이 함께 수행되거나 전혀 수정되지 않도록 함