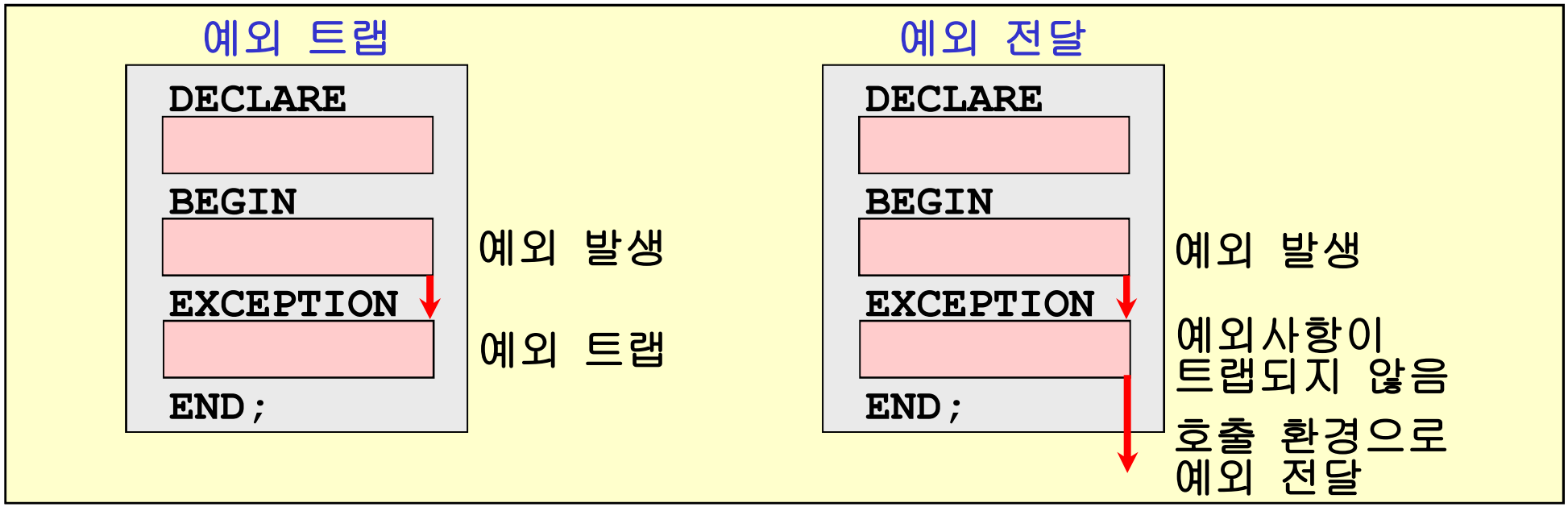
Handling Exceptions

1. 예외 처리 개요

- 예외사항은 블록 실행 중에 발생하는 PL/SQL의 식별자로 작업의 기본 본문 종료
- 예외사항이 발생하면 항상 종료하지만 예외 처리를 지정하여 최종 작업 수행 가능
- 예외사항을 발생시키는 두 가지 메소드
  - Oracle 오류가 발생하면 관련된 예외사항이 자동으로 발생
  - RAISE 문을 실행하여 사용자가 예외사항을 명시적으로 발생 시킴
- 예외 처리 방법
  - Handler로 트랩(Trap)
  - 호출 환경으로 전달

2. 예외 처리



- 예외 트랩
  - 블록의 EXCEPTION 섹션에 있는 해당 예외 Handler로 처리가 분기 됨
  - PL/SQL이 예외사항을 처리하면 PL/SQL 블록이 성공적으로 종료
- 예외 전달
  - 해당하는 예외 Handler가 없으면 PL/SQL 블록 실패
  - 예외사항은 호출 환경으로 전달

3. 예외 유형

3-1. 미리 정의한 Oracle Server 예외사항

- PL/SQL 코드에서 가장 자주 발생하는 약 20개 오류 중 하나
- 선언하지 말고 Oracle Server가 암시적으로 발생 시킴

3-2. 미리 정의하지 않은 Oracle Server 예외사항

- 기타 표준 Oracle Server 오류
- 선언 부분에서 선언하고 Oracle Server가 암시적으로 발생 시킴

3-3. 사용자가 정의한 예외사항

- 개발자가 비정상이라고 판단하는 조건
- 선언 부분에서 선언하고 명시적으로 발생 시킴

4. 예외 트랩(Trap)

4-1. 개요 (1)

```
EXCEPTION
    WHEN exception1 [OR exception2 • • •] THEN
       statement1;
       statement2;
       • • •
    [WHEN exception3 [OR exception4 • • •] THEN
       statement1;
       statement2;
       • • •]
    [WHEN OTHERS THEN
       statement1;
       statement2;
       • • •]
```

- PL/SQL 블록의 예외 처리 부분에 해당 루틴을 포함하여 오류 트랩 가능
- 각 Handler는 예외사항을 지정하는 WHEN 절 및 예외사항이 발생할 때 실행될 명령문 시퀀스로 구성

4-2. 개요 (2)

- **EXCEPTION** 키워드로 블록의 예외 처리 부분 시작
- 블록에 대해 각각 고유의 작업 집합이 있는 여러 개의 예외 **Handler** 정의
- 예외사항이 발생하면 **PL/SQL** 블록을 종료하기 전에 하나의 **Handler**만 처리
- **OTHERS** 절은 다른 모든 예외 처리 절 뒤에 위치
- **OTHERS** 절은 하나만 사용 가능
- 할당**(Assignment)**문 또는 **SQL** 문에는 예외사항을 나타낼 수 없음

4-3. 미리 정의한 예외 트랩

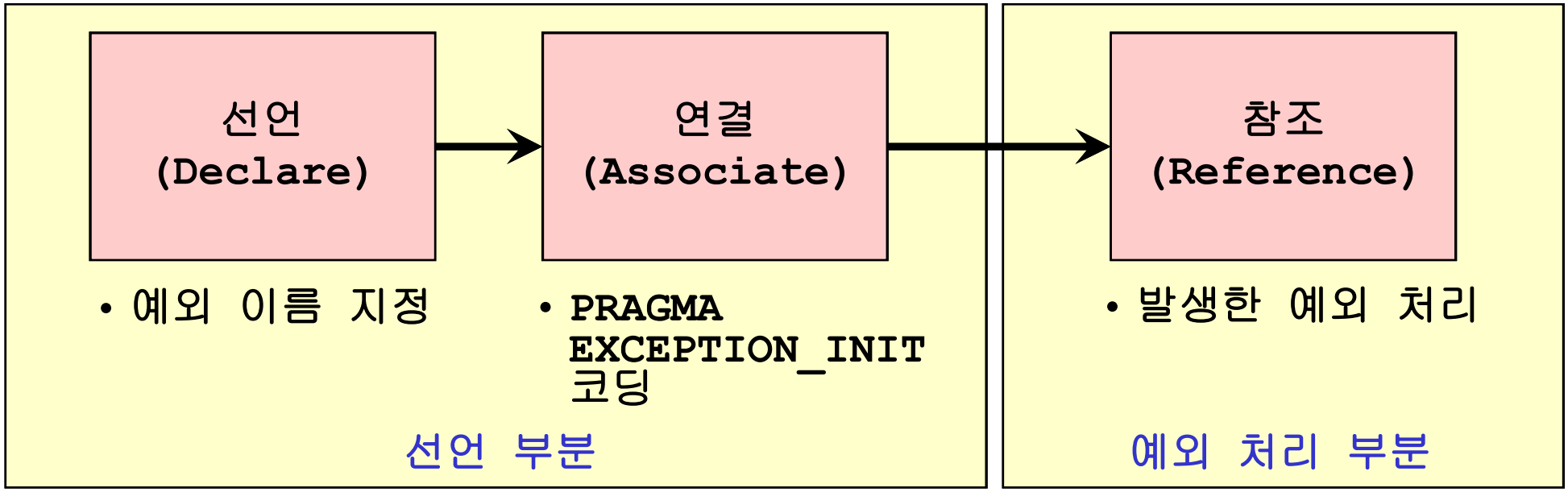
```
BEGIN
    • • •
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        statement1;
        statement2;
    WHEN TOO_MANY_ROWS THEN
        statement1;
    WHEN OTHERS THEN
        statement1;
        statement2;
        statement3;
END;
```

- 예외 처리 루틴에서 표준 이름 참조
- 미리 정의한 예외사항의 목록은 *PL/SQL User's Guide and Reference, "Error Handling"* 참조

4-3. 미리 정의한 예외 트랩

Exception Name	Error Number	Description
NO_DATA_FOUND	ORA-01403	데이터를 리턴하지 않은 SELECT 문
TOO_MANY_ROWS	ORA-01422	SELECT 문이 하나 이상의 ROW 리턴
INVALID_CORSOR	ORA-01001	잘못된 커서 연산 발생
ZERO_DEVIDE	ORA-01476	0으로 나눔
DUP_VAL_ON_INDEX	ORA-00001	중복값 삽입 시도

4-4. 미리 정의하지 않은 예외 트랩 (1)



- 미리 정의하지 않은 Oracle Server 오류는 먼저 오류를 선언하거나 OTHERS Handler를 사용하여 트랩
- 선언한 예외사항은 암시적으로 발생

4-5. 미리 정의하지 않은 예외 트랩 (2)

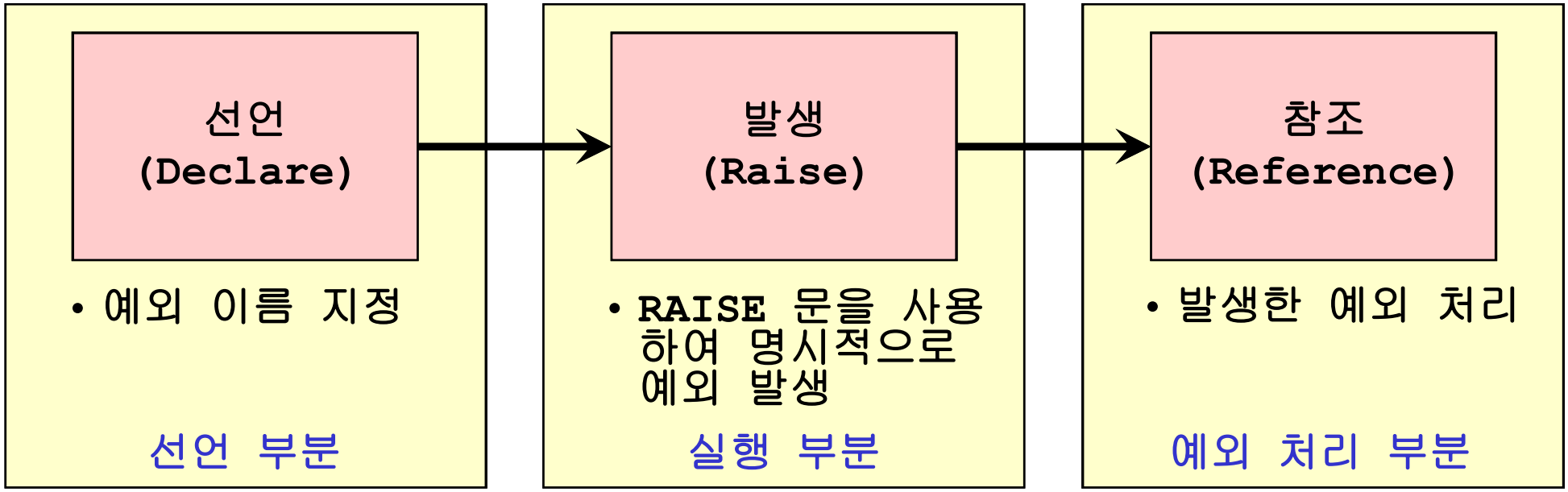
```
DECLARE
    e_emps_remaining EXCEPTION;                                    ①
    PRAGMA EXCEPTION_INIT (e_emps_remaining, -2292);               ②
    v_deptno  dept.deptno%TYPE := &p_deptno;
BEGIN
    DELETE FROM dept
    WHERE       deptno = v_deptno;
    COMMIT;
EXCEPTION
    WHEN e_emps_remaining THEN                                     ③
        DBMS_OUTPUT.PUT_LINE('Cannot remove dept ' ||
              TO_CHAR(v_deptno) || '. Employees exist.');
END;
```

① 선언 부분에서 예외사항의 이름 선언

② PRAGMA EXCEPTION INIT 문을 사용하여 선언한 예외사항과 표준 Oracle Server 오류 번호를 연결

③ 선언한 예외사항을 해당하는 예외 처리 루틴에서 참조

4-6. 사용자가 정의한 예외 트랩 (1)



- PL/SQL을 사용하여 고유한 예외사항 정의 가능
- 사용자가 정의한 PL/SQL 예외사항은 PL/SQL 블록의 선언 부분에서 선언
- RAISE 문을 사용하여 명시적으로 발생 시킴

4-7. 사용자가 정의한 예외 트랩 (2)

```
DECLARE
    e_invalid_department EXCEPTION;                    ①
BEGIN
    UPDATE    dept
    SET       dname = '&department_description'
    WHERE     deptno = &department_number;
    IF SQL%NOTFOUND THEN
        RAISE e_invalid_department;                    ②
    END IF;
    COMMIT;
EXCEPTION
    WHEN e_invalid_department THEN                     ③
       DBMS_OUTPUT.PUT_LINE('Invalid department number.');
END;
```

① 선언 부분에서 사용자가 정의한 예외사항의 이름 선언
② 실행 부분에서 **RAISE** 문을 사용하여 예외사항 명시적으로 발생
③ 선언한 예외사항을 해당하는 예외 처리 루틴에서 참조

4-8. 예외 트랩 함수

```
DECLARE
    v_error_code      NUMBER;
    v_error_message   VARCHAR2(255);
BEGIN
    • • •
EXCEPTION
    • • •
    WHEN OTHERS THEN
        ROLLBACK;
        v_error_code := SQLCODE;
        v_error_message := SQLERRM;
        INSERT INTO errors
        VALUES (v_error_code, v_error_message);
END;
```

- SQLCODE : 오류 코드의 숫자 값 반환
- SQLERRM : 오류 번호와 연관된 메시지 반환

4-8. 예외 트랩 함수

- SQLCODE : 오류 코드의 숫자 값 반환

SQLCODE Value	Description
0	예외가 없음
1	사용자 정의 예외
+100	NO_DATA_FOUND 예외
negative number	다른 오라클 서버 에러 번호

4-8. 예외 트랩 함수

```
SQL> CREATE TABLE log_table
  2   (code      NUMBER(10),
  3    message   VARCHAR2(200),
  4    info      VARCHAR2(200));
```

```
DECLARE
  e_toomanyemp EXCEPTION;
  v_empsal     NUMBER(7);
  v_empcomm    NUMBER(7);
  v_errorcode  NUMBER;
  v_errortext  VARCHAR2(200);
BEGIN
  SELECT sal, comm
  INTO   v_empsal, v_empcomm
  FROM   emp
  WHERE  empno = 7654;
  IF v_empcomm > v_empsal THEN
    RAISE e_toomanyemp;
  END IF;
EXCEPTION
  WHEN e_toomanyemp THEN
    INSERT INTO log_table(info)
    VALUES ('이 사원은 보너스가 '||v_empcomm||'으로 월급여 '||
            v_empsal||' 보다 많다');
  WHEN OTHERS THEN
    v_errorcode := SQLCODE;
    v_errortext := SUBSTR(SQLERRM, 1, 200);
    INSERT INTO log_table
    VALUES (v_errorcode, v_errortext, 'Oracle error occurred');
END;
```

5. 예외 전달

```
DECLARE
    • • •
    e_no_rows       exception;
    e_integrity     exception;
    PRAGMA EXCEPTION_INIT (e_integrity, -2292);
BEGIN
    FOR c_record IN emp_cursor LOOP
        BEGIN
            SELECT  • • •
            UPDATE  • • •
            IF SQL%NOTFOUND THEN
                RAISE   e_no_rows;
            END IF;
        EXCEPTION
            WHEN e_integrity THEN  • • •
            WHEN e_no_rows   THEN  • • •
        END;

    END LOOP;
EXCEPTION
    WHEN  NO_DATA_FOUND    THEN  • • •
    WHEN  TOO_MANY_ROWS    THEN  • • •
END;
```

하위 블록에서 예외사항을 처리하거나 포함된 블록에 전달할 수 있음

6. RAISE_APPLICATION_ERROR 프로시저

```
RAISE_APPLICATION_ERROR (error_number,
                         message[, {TRUE | FALSE}]);
```

```
• • •
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR (-20201,
                        'Manager is not a valid employee.');
END;
```

- 사용자가 정의한 오류 메시지를 내장(Stored) 하위 프로그램에서 실행하는 프로시저
- 내장(Stored) 하위 프로그램 실행을 통해서만 호출
- 실행 섹션, Exception 섹션에서 사용
- 다른 Oracle Server 오류와 동일한 방식으로 사용자에게 오류 조건을 반환