Pembuatan Procedure

Pertemuan 10

Tujuan umum

- Memahami perbedaan antara PL/SQL block yang bernama (sub program yaitu procedure dan function) dengan yang tidak bernama (anonymous block).
- Memahami keuntungan penggunaan sub program
- Mengetahui environment dimana sub program dapat dipanggil

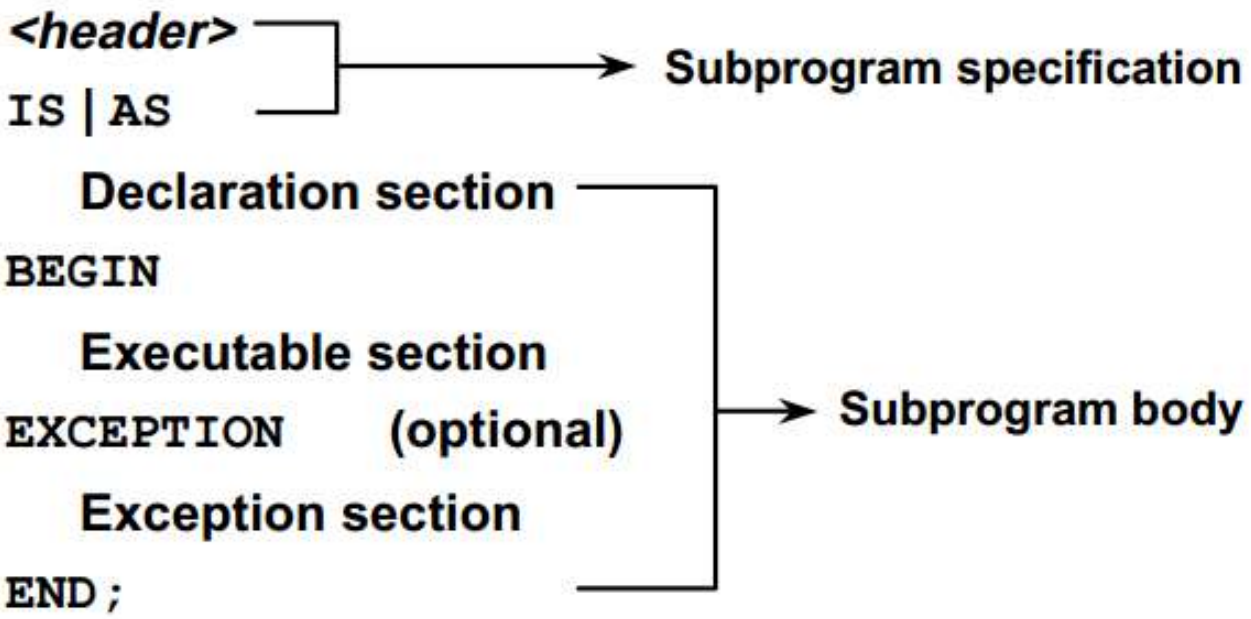
Tujuan khusus

- Memahami pembuatan PL/SQL block dan PL/SQL sub program
- Memahami pembuatan procedure
- Dapat membedakan antara parameter formal dengan parameter aktual
- Dapat membuat procedure dengan parameter
- Memanggil procedure di lingkungan yang lain
- Dapat menangani exception dalam procedure
- Dapat menghapus procedure

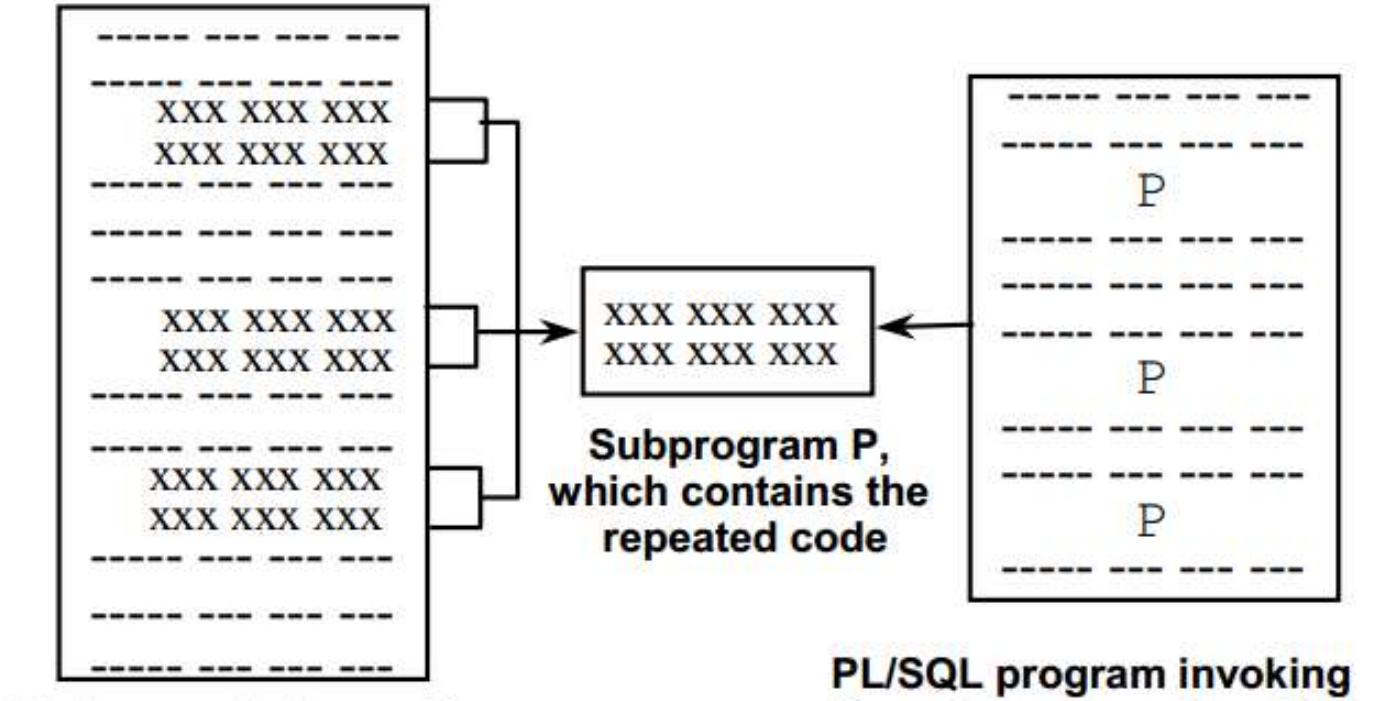
Review : subprogram

- PL/SQL block yang memiliki nama yang dapat memiliki parameter dan yang dapat dipanggil dari lingkungan tertentu.
- Ada 2 tipe:
  - Procedure yang membentuk suatu tindakan
  - Function yang menghitung suatu nilai
- Ditulis berdasarkan struktur blok PL/SQL
- Menyediakan modularitas, reusability, extensibility, dan maintanibility
- Menyediakan maintenance secara mudah, meningkatkan sekuritas dan integritas data, meningkatkan performansi dan meningkatan kejelasan code program.

Blok struktur dari PL/SQL subprogram



PL/SQL subprogram



Apa itu procedure?

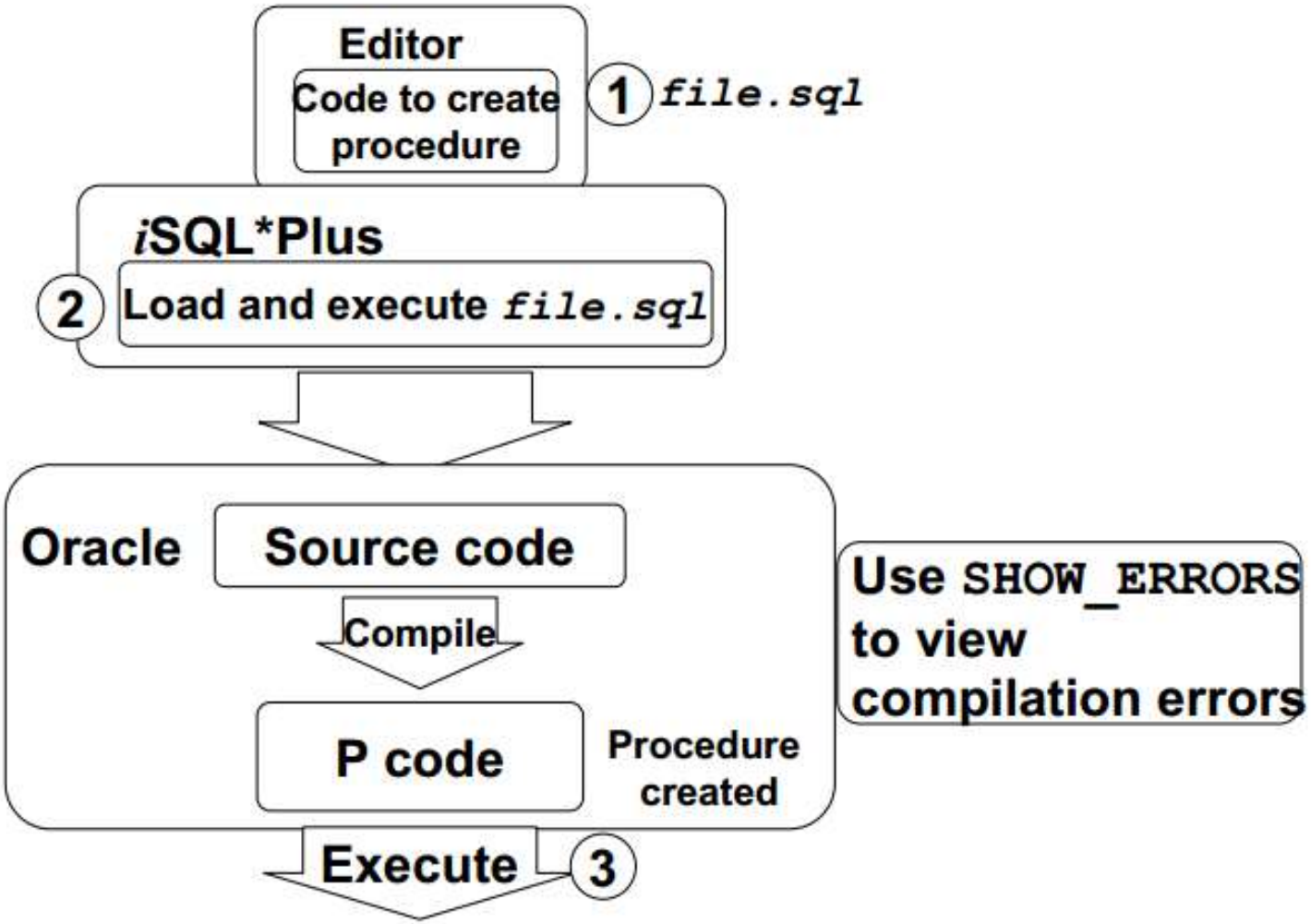
- Procedure adalah tipe subprogram yang membentuk sebuah aksi
- Procedure dapat disimpan dalam database karena merupakan salah satu dari obyek database

Pembuatan procedure

- Sintaks:

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

Pengembangan Procedure



Parameter formal vs. parameter aktual

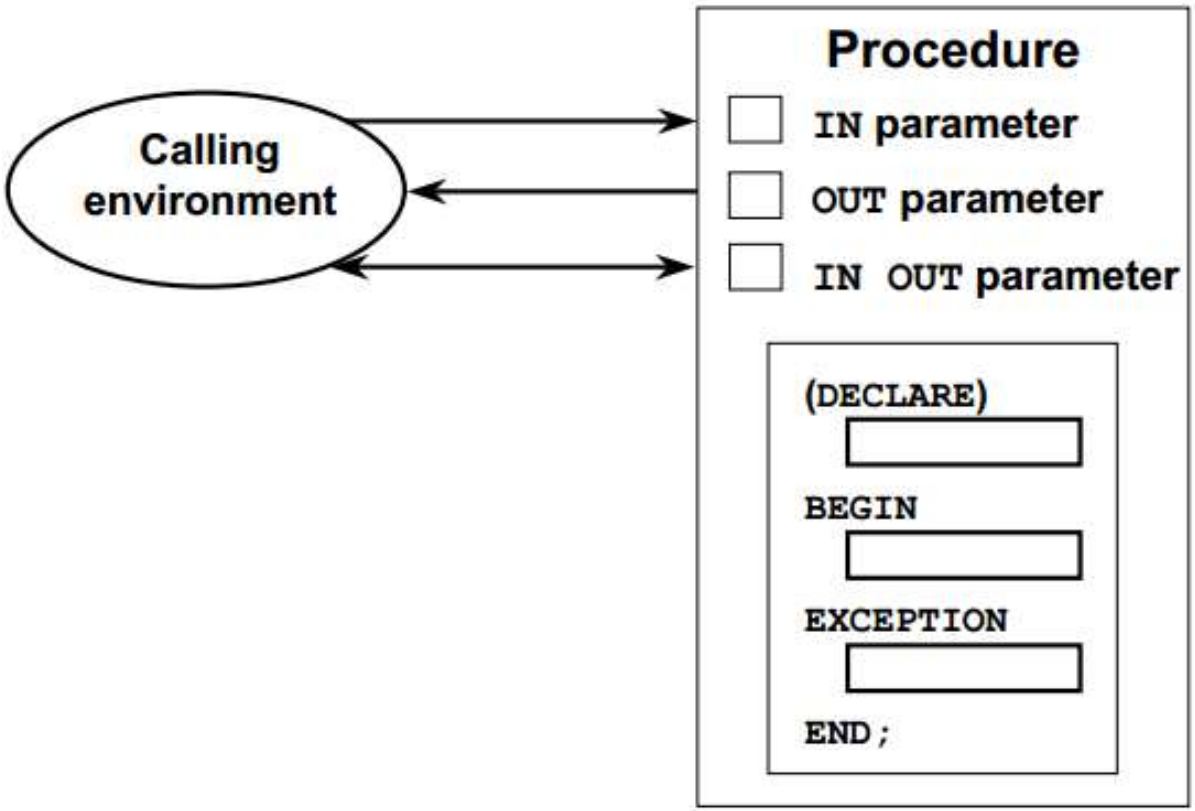
- Parameter formal: variabel dideklarasikan pada list parameter yang ada di bagian spesikasi subprogram

```
CREATE PROCEDURE raise_sal(p_id NUMBER, p_amount NUMBER)
...
END raise_sal;
```

- Parameter aktual: variabel atau ekspresi diberikan sebagai daftar parameter untuk memanggil subprogram

```
raise_sal(v_id, 2000)
```

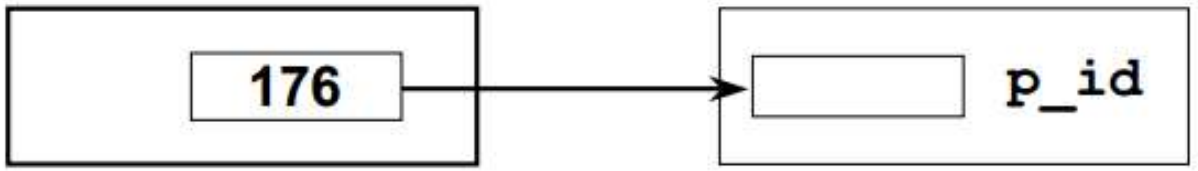
Model Parameter



Pembuatan procedure dengan parameter

- IN
  - Nilai dilewatkan ke dalam subprogram
- OUT
  - Nilai dikembalikan ke environment yang memanggil
- IN OUT
  - Nilai dilewatkan ke dalam subprogram dan dikembalikan lagi ke environment yang memanggil

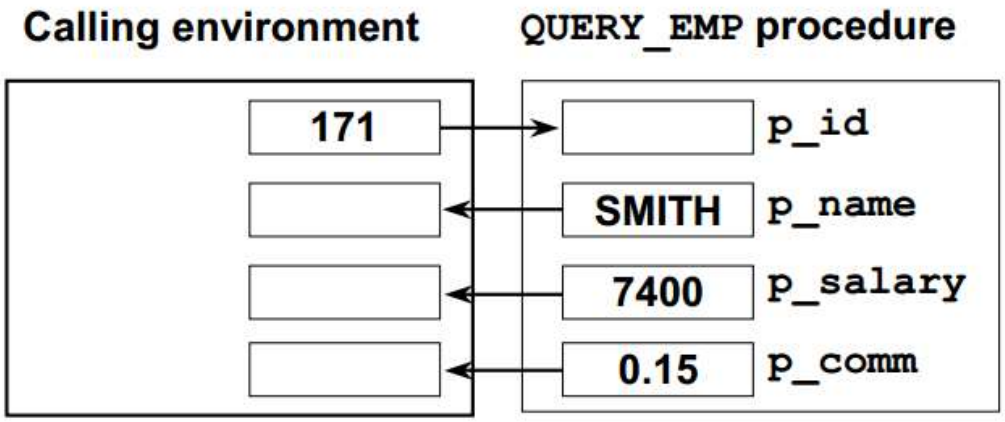
Contoh penggunaan parameter IN




```
CREATE OR REPLACE PROCEDURE raise_salary
  (p_id IN employees.employee_id%TYPE)
IS
BEGIN
  UPDATE employees
  SET    salary = salary * 1.10
  WHERE  employee_id = p_id;
END raise_salary;
/
```

Procedure created.

Contoh penggunaan parameter OUT




```
CREATE OR REPLACE PROCEDURE query_emp
   (p_id       IN   employees.employee_id%TYPE,
    p_name     OUT  employees.last_name%TYPE,
    p_salary   OUT  employees.salary%TYPE,
    p_comm     OUT  employees.commission_pct%TYPE)
IS
BEGIN
  SELECT   last_name, salary, commission_pct
   INTO     p_name, p_salary, p_comm
   FROM     employees
   WHERE    employee_id = p_id;
END query_emp;
/
```

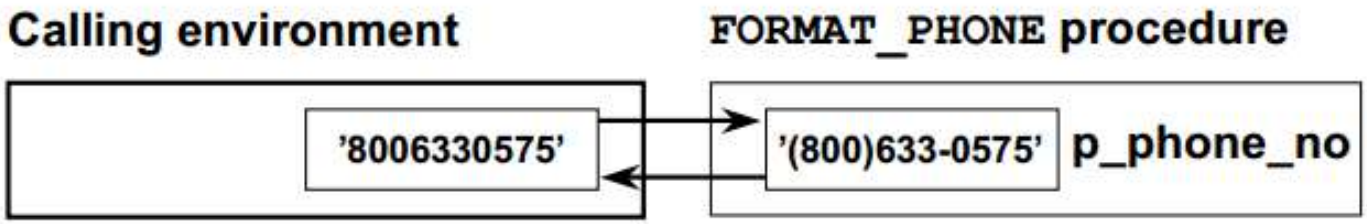
Menampilkan hasil dari parameter OUT

```
VARIABLE g_name      VARCHAR2(25)
VARIABLE g_sal       NUMBER
VARIABLE g_comm      NUMBER

EXECUTE query_emp(171, :g_name, :g_sal, :g_comm)

PRINT g_name
```

Contoh penggunaan parameter IN OUT




```
CREATE OR REPLACE PROCEDURE format_phone
   (p_phone_no IN OUT VARCHAR2)
IS
BEGIN
  p_phone_no := '(' || SUBSTR(p_phone_no,1,3) ||
                ')' || SUBSTR(p_phone_no,4,3) ||
                '-' || SUBSTR(p_phone_no,7);
END format_phone;
/
```

Menampilkan hasil dari parameter IN OUT

```
VARIABLE g_phone_no VARCHAR2(15)
BEGIN
  :g_phone_no := '8006330575';
END;
/
PRINT g_phone_no
EXECUTE format_phone (:g_phone_no)
PRINT g_phone_no
```

PL/SQL procedure successfully completed.

G_PHONE_NO
8006330575

PL/SQL procedure successfully completed.

G_PHONE_NO
(800)633-0575

Metode untuk melewatkan parameter

- Posisi
  - Menuliskan daftar parameter dengan posisi yang sama dengan parameter formal
- Nama
  - Menuliskan daftar parameter yang berasosisasi dengan setiap parameter formal yang berkorespondensi
- Kombinasi
  - Gabungan antara metode posisi dan metode nama

Penggunaan DEFAULT dalam parameter

- Digunakan untuk menginisialisasi parameter IN ke dalam nilai DEFAULT.

```
CREATE OR REPLACE PROCEDURE add_dept
  (p_name  IN departments.department_name%TYPE
                              DEFAULT 'unknown',
   p_loc   IN departments.location_id%TYPE
                              DEFAULT 1700)
IS
BEGIN
  INSERT INTO departments(department_id,
              department_name, location_id)
  VALUES (departments_seq.NEXTVAL, p_name, p_loc);
END add_dept;
/
```

```
Procedure created.
```

Cara melewatkan parameter dengan berbagai macam metode

```
BEGIN
  add_dept;
  add_dept ('TRAINING', 2500);
  add_dept ( p_loc => 2400, p_name =>'EDUCATION');
  add_dept ( p_loc => 1200) ;
END;
/
SELECT department_id, department_name, location_id
FROM departments;
```

Subprogram di dalam sebuah Procedure

```
CREATE OR REPLACE PROCEDURE leave_emp2
   (p_id  IN  employees.employee_id%TYPE)
IS
   PROCEDURE log_exec
   IS
   BEGIN
     INSERT INTO log_table (user_id, log_date)
     VALUES (USER, SYSDATE);
   END log_exec;
BEGIN
   DELETE FROM employees
   WHERE employee_id = p_id;
   log_exec;
END leave_emp2;
/
```

Memanggil Procedure dari anonymous PL/SQL block

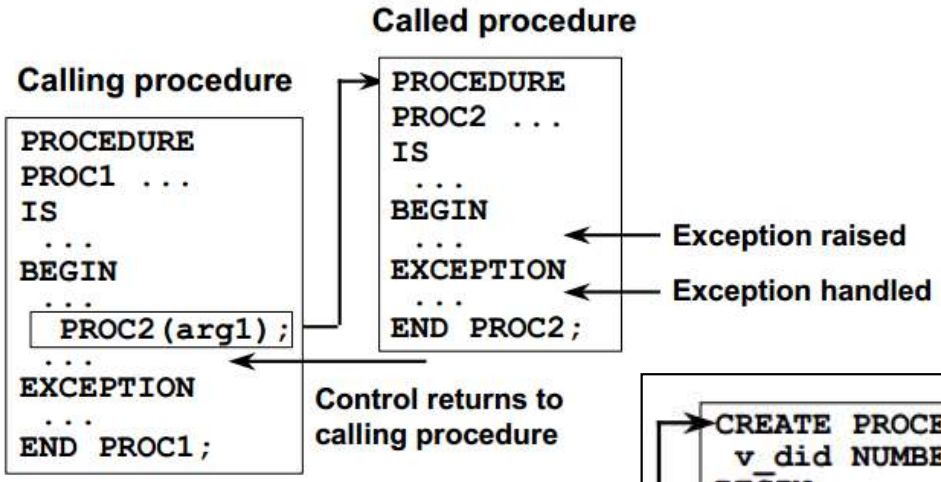
```
DECLARE
  v_id NUMBER := 163;
BEGIN
  raise_salary(v_id);    --invoke procedure
  COMMIT;
...
END;
```

Memanggil Procedure dari Procedure yang lain

process_emps.sql

```
CREATE OR REPLACE PROCEDURE process_emps
IS
   CURSOR emp_cursor IS
    SELECT employee_id
    FROM   employees;
BEGIN
   FOR emp_rec IN emp_cursor
   LOOP
    raise_salary(emp_rec.employee_id);
   END LOOP;
   COMMIT;
END process_emps;
/
```

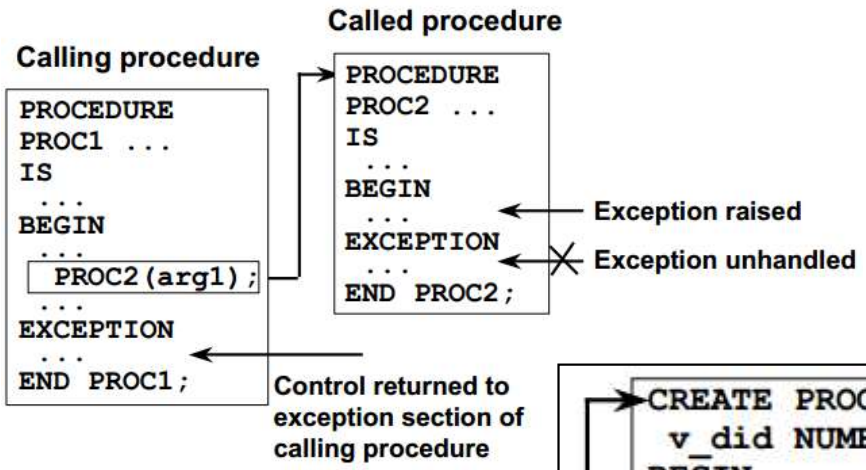
Meng-handle Exception




```
CREATE PROCEDURE p2_ins_dept(p_locid NUMBER) IS
 v_did NUMBER(4);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Procedure p2_ins_dept started');
 INSERT INTO departments VALUES (5, 'Dept 5', 145, p_locid);
 SELECT department_id INTO v_did FROM employees
  WHERE employee_id = 999;
END;
```

```
CREATE PROCEDURE p1_ins_loc(p_lid NUMBER, p_city VARCHAR2)
IS
 v_city VARCHAR2(30); v_dname VARCHAR2(30);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Main Procedure p1_ins_loc');
 INSERT INTO locations (location_id, city) VALUES (p_lid, p_city);
 SELECT city INTO v_city FROM locations WHERE location_id = p_lid;
 DBMS_OUTPUT.PUT_LINE('Inserted city '||v_city);
 DBMS_OUTPUT.PUT_LINE('Invoking the procedure p2_ins_dept ...');
 p2_ins_dept(p_lid);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
   DBMS_OUTPUT.PUT_LINE('No such dept/loc for any employee');
END;
```

Exception yang tidak ter-handle




```
CREATE PROCEDURE p2_noexcep(p_locid NUMBER) IS
 v_did NUMBER(4);
BEGIN
 DBMS_OUTPUT.PUT_LINE('Procedure p2_noexcep started');
 INSERT INTO departments VALUES (6, 'Dept 6', 145, p_locid);
 SELECT department_id INTO v_did FROM employees
  WHERE employee_id = 999;
END;
```

```
CREATE PROCEDURE p1_noexcep(p_lid NUMBER, p_city VARCHAR2)
IS
 v_city VARCHAR2(30); v_dname VARCHAR2(30);
BEGIN
 DBMS_OUTPUT.PUT_LINE(' Main Procedure p1_noexcep');
 INSERT INTO locations (location_id, city) VALUES (p_lid, p_city);
 SELECT city INTO v_city FROM locations WHERE location_id = p_lid;
 DBMS_OUTPUT.PUT_LINE('Inserted new city '||v_city);
 DBMS_OUTPUT.PUT_LINE('Invoking the procedure p2_noexcep ...');
 p2_noexcep(p_lid);
END;
```

Menghapus Procedure

- Sintaks:

```
DROP PROCEDURE procedure_name
```

- Contoh:

```
DROP PROCEDURE raise_salary;
```

Ringkasan

- Subprogram adalah PL/SQL block yang memiliki nama, memiliki parameter dan bisa dipanggil
- Procedure adalah subprogram yang melakukan suatu aksi/tindakan
- Telah dipelajari cara pembuatan procedure dan tipe parameter dalam procedure (IN, OUT, IN OUT).
- Telah dipelajari penanganan exception di dalam procedure