

PENGENALAN PL/SQL

Pokok Bahasa/Materi

- ▶ Pengertian RDBMS
- ▶ Pengenalan PL/SQL



Pendahuluan

- ❑ Seperti yang diketahui bahwa dalam model relasional, suatu database dibangun dengan banyak tabel. Sebuah tabel terdiri dari kolom dan dilengkapi dengan baris yang merupakan data atau konten. Namun, dalam mengolah suatu database yang terintegrasi diperlukannya suatu konsep untuk menghubungkan tabel yang satu dengan yang lainnya melalui kunci yang dimiliki. Itulah yang disebut dengan RDBMS (Relational Database Management System).
- ❑ Konsep RDBMS merupakan sistem yang mendukung adanya hubungan atau relationship antar tabel pada suatu database. Setiap tabel memiliki kunci yang disebut dengan *primary key* untuk dihubungkan ke tabel berikutnya yang memiliki *foreign key*.



-
- ❑ RDBMS sudah banyak digunakan oleh berbagai vendor sejak tahun 1970-an. Seiring dengan berkembangnya keunggulan RDBMS, banyak perusahaan yang awalnya menggunakan model hirarki dan jaringan beralih ke model RDBMS. Sebab, model ini mudah untuk digunakan dan dipahami. Awalnya model ini hanya digunakan oleh perusahaan besar, namun kini sudah banyak jenis database yang menerapkan model RDBMS didalamnya, seperti Microsoft Access, MySQL, SQL Server, Oracle, PostgreSQL, OpenOffice Base dan FoxBase.



► Perbedaan RDBMS dan DBMS

RDBMS adalah arsitektur database yang tabel-tabelnya mempunyai hubungan atau relationship satu sama lain. Hubungan disini menggunakan key pada masing-masing tabel. Sedangkan kebalikannya, DBMS tidak harus membutuhkan hubungan antar tabel di dalamnya. Keduanya merupakan metode yang diterima secara umum untuk membangun arsitektur sebuah database.

Contoh :

DBMS – *File System, XML*

RDBMS – *SQL Server, Oracle*



❑ Bagian Dasar dalam SQL

- SQL adalah bahasa. Oracle SQL adalah superset dari American National Standards Institute (ANSI) dan standar Organisasi Standar Internasional (ISO) SQL92 pada kesesuaian entry level.
- PL / SQL adalah ekstensi bahasa prosedural Oracle ke SQL. Ini memungkinkan Anda untuk menghubungkan beberapa perintah SQL melalui bahasa prosedural.
- Oracle SQL Developer adalah alat perangkat lunak sisi klien grafis yang digunakan untuk menyambung ke server dari jarak jauh.



PENGENALAN PL/SQL



Pendahuluan

- ❑ PL/SQL (Procedural Language/Structure Query Language)
 - Adalah suatu blok yang berisi skrip-skrip bahasa prosedural.
- ❑ PL/SQL merupakan bahasa pemrograman prosedural
- ❑ PL/SQL dapat meningkatkan kinerja database
- ❑ PL/SQL merupakan bahasa pemrograman yang menggabungkan bahasa procedural, seperti pernyataan percabangan (IF-THEN-ELSE), pengulangan (LOOP) dan deklarasi variable. PL/SQL dikembangkan oleh Oracle untuk pembuatan Fungsi, Database Trigger, dan Stored Procedure.



Tipe Data

Tipe Data dasar :

☐ Numerik

- NUMBER, BINARY_INTEGER, DEC, DOUBLE PRECISION, INTEGER, INT, NUMERIC, REAL, SMALLINT

☐ Karakter

- VARCHAR2, CHAR, LONG

☐ DATE

☐ BOOLEAN

☐ ROWID

Tipe Data tambahan :

☐ RECORD

☐ ARRAY



Variabel

- Adalah sebuah perubah yang digunakan untuk menampung sebuah nilai di memori komputer.

Contoh Variabel

DECLARE

 X integer;

 Alamat varchar2(40);

 No_induk char(8);

.....

BEGIN

 X := 12;

 Alamat := 'Gelatik Dalam 391, Bandung';

 No_induk := 'DOG29549';

END;



Konstanta

- Digunakan untuk menyimpan sebuah nilai di memori komputer.
- Nilai yang disimpan bersifat tetap (konstan)

Contoh :

DECLARE

pi CONSTANT real := 3.14;

lebar CONSTANT integer := 100;



Komentar

- Digunakan untuk memudahkan proses maintenance
- Jenis komentar :
 - `/* ... */` : untuk beberapa baris komentar
 - `-- ...` : untuk satu baris komentar

Contoh :

`/* Ini adalah komentar Oracle */`

`-- Ini juga komentar Oracle`



Struktur Blok PL/SQL

- **Terdapat tiga bagian :**
 - Bagian pendeklarasian tipe data (opsional)
 - Bagian penulisan perintah
 - Bagian eksepsi (opsional)



Bentuk Umum Struktur Umum PL/SQL

DECLARE

variabel tipe_data;

konstanta CONSTANT tipe_data := nilai;

...

BEGIN

statement_1;

statement_2;

EXCEPTION

WHEN nama_eksepsi THEN statement_untuk_mengatasi_error;

...

END;



Contoh Paling Sederhana

```
BEGIN
```

```
  DBMS_OUTPUT.PUT_LINE('Belajar Oracle');
```

```
END;
```

Catatan:

Untuk melihat hasil, setting terlebih dahulu variabel sistem
SERVEROUTPUT dengan menuliskan :

```
SET SERVEROUTPUT ON
```



Contoh Lain

```
SET SERVEROUTPUT ON
DECLARE
    teks VARCHAR2 (20);
BEGIN
    teks := 'Belajar Oracle';
    DBMS_OUTPUT.PUT_LINE(teks);
END;
/
```



Contoh dengan eksepsi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X Integer;
```

```
BEGIN
```

```
    X := 'Belajar Oracle';
```

```
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(X) );
```

```
EXCEPTION
```

```
    WHEN VALUE_ERROR THEN
```

```
        DBMS_OUTPUT.PUT_LINE ('Kesalahan pada pengisian nilai');
```

```
END;
```

```
/
```



OPERATOR



Pendahuluan

- Terminologi

$Z := 3 + 6$

Maka :

Z	disebut variabel
:=	disebut operator assignment
3 dan 6	disebut operand
3 + 6	disebut ekspresi
+	disebut operator aritmatika
Z:=3+6	disebut statemen aritmatika



Jenis Operator

- **Operator Logika**
 - Operator NOT ,AND , OR
- **Operator Aritmatika**
 - Operator + , - , * , / , MOD
- **Operator Relasional**
 - Operator < , <= , > , >=
- **Operator Persamaan**
 - Operator = , <>
- **Operator Penggabungan**
 - Operator ||



Contoh Penggunaan Operator +

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X INTEGER;
```

```
    Y NUMBER;
```

```
BEGIN
```

```
    X := 2 + 3;
```

```
    Y := 2.45 + 3.14;
```

```
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(X) );
```

```
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(Y) );
```

```
END;
```

```
/
```



Contoh Penggunaan Operator -

SET SERVEROUTPUT ON

DECLARE

 X INTEGER;

 Y NUMBER;

BEGIN

 X := 5 - 3;

 Y := 6.88 - 2.73;

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(X));

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(Y));

END;

/



Contoh Penggunaan Operator *

SET SERVEROUTPUT ON

DECLARE

 X INTEGER;

 Y NUMBER;

BEGIN

 X := 5 * 2;

 Y := 6.13 - 2;

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(X));

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(Y));

END;

/



Contoh Penggunaan Operator /

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X NUMBER;
```

```
BEGIN
```

```
    X := 10 / 3;
```

```
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(X) );
```

```
END;
```

```
/
```



Contoh Penggunaan Operator MOD

SET SERVEROUTPUT ON

DECLARE

 X INTEGER;

BEGIN

 X := 10 MOD 3;

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(X));

END;

/



Contoh Penggunaan Operator Relasional

```
SET SERVEROUTPUT ON
DECLARE
    X INTEGER;
BEGIN
    X := 10;
    IF (X <= 5) THEN
        DBMS_OUTPUT.PUT_LINE('TRUE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('FALSE');
    END IF;
END;
/
```



Contoh Penggunaan Operator Persamaan

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X INTEGER;
```

```
BEGIN
```

```
    X := 10;
```

```
    IF (X = 10) THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Sepuluh');
```

```
    ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('Bukan Sepuluh');
```

```
    END IF;
```

```
END;
```

```
/
```



Contoh Penggunaan Operator Penggabungan

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X NUMBER;
```

```
    VARCHAR2 (25);
```

```
BEGIN
```

```
    X := 15.2 + 4.63 ;
```

```
    S := 'Nilai X sama dengan ';
```

```
    DBMS_OUTPUT.PUT_LINE(S || TO_CHAR(X) );
```

```
END;
```

```
/
```



ARRAY



Pendahuluan

▪ Definisi

- Tipe data bentukan yang dapat menyimpan sekumpulan nilai dari tipe data yang sama dan dikemas dalam bentuk larik.
- Nilai dari elemen-elemen array dapat diacu atau diakses melalui indeksinya, perlu diperhatikan bahwa indeks array harus dari tipe data yang mempunyai keterurutan, seperti halnya tipe integer.



Membuat Tipe Array

- Bentuk Umum :

```
TYPE nama_tipe IS  
    TABLE OF tipe_data  
    INDEX BY BINARY_INTEGER;
```

Contoh :

```
DECLARE  
TYPE array_ku IS  
    TABLE OF CHAR(5)  
    INDEX BY BINARY_INTEGER;  
X array_ku;
```



Mengisikan Nilai pada Elemen Array

- Contoh 1 :

```
BEGIN
```

```
    X(1) := 'A';
```

```
    X(2) := 'B';
```

```
END;
```

- Contoh 2 :

```
BEGIN
```


```
    X(1) := 10;
```

```
    X(2) := 20;
```

```
END;
```



Contoh 1

```
SET SERVEROUTPUT ON
DECLARE
    TYPE LARIK IS
        TABLE OF NUMBER
        INDEX BY BINARY_INTEGER;
    A LARIK;
    I INTEGER;
BEGIN
    FOR I IN 1..5 LOOP
        A(I) := I * 10;
    END LOOP;
    FOR I IN 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE('Nilai elemen larik ke-' || TO_CHAR(I) || ' = '
        || TO_CHAR(A(I)));
    END LOOP;
END;/  

```

Contoh 2

```
SET SERVEROUTPUT ON
DECLARE
    TYPE SISWA IS
        TABLE OF VARCHAR2(25)
        INDEX BY BINARY_INTEGER;
    NAMA SISWA;
    I INTEGER;
BEGIN
    NAMA(1) := 'Arista Destriana';
    NAMA(2) := 'Yandri Gunawan';
    NAMA(3) := 'Herry Wahyudinata';
    NAMA(4) := 'Budi Raharjo';
    NAMA(5) := 'Noni Sutrisna';
    FOR I IN 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE('Nama siswa ke-' || TO_CHAR(I) || ' : ' || NAMA(I));
    END LOOP;
END;
/
```



EKSEPSI



Pendahuluan

- ▶ Blok Eksepsi (*exception block*) adalah blok yang digunakan untuk menjebak error yang mungkin terjadi di dalam blok PL/SQL
- ▶ Eksepsi (*exception*) merupakan jenis-jenis error yang menyebabkan terhentinya program secara tidak normal



Tujuan dan Jenis

- Tujuan Blok Eksepsi adalah menangani error-error yang telah didefinisikan menjadi sebuah eksepsi sehingga meskipun terdapat error, error tersebut tidak akan ditampilkan melainkan dilempar ke bagian eksepsi.
- Jenis Eksepsi :
 - Pre-defined exception
 - User-defined exception



Bentuk Umum Eksepsi

EXCEPTION

WHEN eksepsi_pertama THEN
 statemen_utmengatasi_eksepsi_pertama;

WHEN eksepsi_kedua THEN
 statemen_utmengatasi_eksepsi_kedua;

...

WHEN OTHERS THEN
 statemen_utmengatasi_eksepsi_lainnya;

END;



Pre-defined Exception

- Adalah sebuah eksepsi yang telah didefinisikan atau sudah tersedia dalam Oracle, sehingga dapat langsung menggunakan tanpa harus membuatnya terlebih dahulu



Contoh 1, Tanpa blok eksepsi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X INTEGER;
```

```
    Y NUMBER;
```

```
BEGIN
```

```
    X := 0;
```

```
    Y := 1 / X;
```

```
    DBMS_OUTPUT.PUT_LINE('Nilai Y = ' || TO_CHAR(Y));
```

```
END;
```

```
/
```

Hasil yang muncul dilayar:

ORA : divisor is equal to zero



Contoh 1, Dengan blok eksepsi

```
SET SERVEROUTPUT ON
DECLARE
    X INTEGER;
    Y NUMBER;
BEGIN
    X := 0;
    Y := 1 / X;
    DBMS_OUTPUT.PUT_LINE('Nilai Y = ' || TO_CHAR(Y));
EXCEPTION
    WHEN ZERO_DEVIDE THEN
        DBMS_OUTPUT.PUT_LINE('Terjadi kesalahan karena
        terdapat ' || 'pembagian dengan 0 (NOL) ');
END;
/
```



Contoh 2, Tanpa blok eksepsi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X NUMBER;
```

```
BEGIN
```

```
    X := 'Budi Raharjo';
```

```
    DBMS_OUTPUT.PUT_LINE('Nilai X = ' || TO_CHAR(X) );
```

```
END;
```

/ Hasil yang muncul dilayar :

ORA-06502: PL/SQL: numeric or value error



Contoh 2, Dengan blok eksepsi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X NUMBER;
```

```
BEGIN
```

```
    X := 'Budi Raharjo';
```

```
    DBMS_OUTPUT.PUT_LINE('Nilai X = ' || TO_CHAR(X) );
```

```
EXCEPTION
```

```
    WHEN VALUE_ERROR THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Pengisian nilai tidak sesuai  
dengan ' || 'tipe variabel');
```

```
END;
```

```
/
```



Nama Eksepsi	Kode Error	Keterangan
CURSOR_ALREADY_OPEN	ORA-6511	Cursor masih dalam keadaan terbuka
DUP_VALUE_ON_INDEX	ORA-0001	Duplikasi constraint
INVALID_CURSOR	ORA-1001	Membuka cursor yang salah
INVALID_NUMBER	ORA-1722	Melakukan operasi numerik pada tipe non-numerik
LOGIN_DENIED	ORA-1017	Username/password salah
NO_DATA_FOUND	ORA-1403	Tidak terdapat data pada query
NOT_LOGGED_ON	ORA-1012	Melakukan operasi database ketika tidak terhubung (connect)
PROGRAM_ERROR	ORA-6501	Kesalahan internal
ROWTYPE_MISMATCH	ORA-6504	Host variabel dan cursor mempunyai tipe yang tidak sama
STORAGE_ERROR	ORA-6500	Kesalahan internal
TYMEOUT_ON_RESOURCE	ORA-0051	Terjadi timeout
TOO_MANY_ROWS	ORA-1422	Statemen SELECT INTO yang mengembalikan lebih dari satu baris
TRANSACTION_BACKED_OUT	ORA-006	Rollback transaksi untuk didealokasikan
VALUE_ERROR	ORA-6502	Konversi atau truncation salah
ZERO_DEVIDE	ORA-1476	Pembagian dengan nol



User-defined Exception

Untuk membuat sebuah eksepsi, diperlukan :

- Variabel bertipe EXCEPTION
- Dihubungkan dengan tipe PRAGMA EXCEPTION_INIT
- Kode error harus negatif (-)
- Bentuk Umum :

DECLARE

nama_eksepsi EXCEPTION;

PRAGMA EXCEPTION_INIT (nama_eksepsi, kode_error);



Contoh tanpa eksepsi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    X ROWID;
```

```
BEGIN
```

```
    SELECT ROWID INTO X FROM ALL_VIEWS;
```

```
END;
```

```
/
```

Hasil yang tampil dilayar :

ORA-01445: Cannot select ROWID from view of more than one table



Contoh dengan eksepsi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    eksepsiku EXCEPTION;
```

```
    PRAGMA EXCEPTION_INIT(eksepsiku, );
```

```
    X ROWID;
```

```
BEGIN
```

```
    SELECT ROWID INTO X FROM ALL_VIEWS;
```

```
EXCEPTION
```

```
    WHEN eksepsiku THEN
```

```
        DBMS_OUTPUT.PUT_LINE('KESALAHAN :Tidak dapat ‘  
        || ‘menampilkan ROWID dari beberapa tabel atau view’);
```

```
END;
```

```
/
```



PENGULANGAN



PENDAHULUAN

- Tiga macam struktur pengulangan :
 - Struktur simple LOOP
 - Struktur WHILE-LOOP
 - Struktur FOR-LOOP



Contoh struktur simple loop

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
  J INTEGER
```

```
BEGIN
```

```
  J := 0;
```

```
  LOOP
```

```
    J := J + 1;
```

```
    DBMS_OUTPUT.PUT_LINE('Saya belajar PL/SQL');
```

```
    EXIT WHEN J = 10;
```

```
  END LOOP;
```

```
END;
```

```
/
```



Struktur WHILE-LOOP

- Bentuk umum :
WHILE kondisi LOOP
 statemen_1;
 ...
END LOOP;



Contoh struktur WHILE-LOOP

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
  J INTEGER
```

```
BEGIN
```

```
  J := 1;
```

```
  WHILE (J <= 10) LOOP
```

```
    DBMS_OUTPUT.PUT_LINE('Saya belajar PL/SQL');
```

```
    J := J + 1;
```

```
  END LOOP;
```

```
END;
```

```
/
```



Struktur FOR-LOOP

- **Bentuk umum :**

```
FOR variabel IN batas_minimal ..  
    batas_maksimal LOOP  
    statemen_l;  
    ...  
END LOOP;
```



Contoh Struktur FOR-LOOP

SET SERVEROUTPUT ON

DECLARE

 I INTEGER;

BEGIN

 FOR I IN LOOP

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(I));

 END LOOP;

END;

/



Contoh Struktur FOR-LOOP (2)

SET SERVEROUTPUT ON

DECLARE

 I INTEGER;

BEGIN

 FOR I IN REVERSE LOOP

 DBMS_OUTPUT.PUT_LINE(TO_CHAR(I));

 END LOOP;

END;

/



PERCABANGAN



Pendahuluan

Percabangan dalam PL/SQL Oracle :

- Struktur satu kondisi
- Struktur dua kondisi
- Struktur tiga atau lebih kondisi



Struktur satu kondisi

- Jika kondisi tidak terpenuhi atau bernilai FALSE, badan percabangan tidak akan pernah dimasuki

- Bentuk umum :

```
IF kondisi THEN  
    statemen_1;  
    ...  
END IF;
```



Contoh Struktur satu kondisi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    BIL INTEGER := 80;
```

```
BEGIN
```

```
    IF MOD(BIL, 2) = 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(BIL) || ' '
ADALAH BILANGAN GENAP');
```

```
    END IF;
```

```
END;
```

```
/
```



Struktur dua kondisi

- Bentuk umum :

IF kondisi THEN

 statemen_1;

 ...

ELSE

 statemen_2;

 ...

END IF;



Contoh struktur dua kondisi

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    BIL INTEGER := 3;
```

```
BEGIN
```

```
    IF MOD(BIL, 2) = 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(BIL) || ' ADALAH  
        BILANGAN GENAP');
```

```
    ELSE
```

```
        BILANGAN GANJIL');
```

```
    END IF;
```

```
END;
```

```
/
```



Struktur tiga kondisi atau lebih

- Bentuk umum :

```
IF kondisi_1 THEN  
    statemen_1;
```

```
    ...
```

```
ELSIF kondisi_2 THEN  
    statemen_2;
```

```
    ...
```

```
ELSE  
    statemen_3;
```

```
    ...
```

```
END IF;
```



Contoh struktur tiga kondisi atau lebih

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    BIL INTEGER;
```

```
BEGIN
```

```
    BIL := -25;
```

```
    IF BIL > 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(BIL) ||  
        ' adalah bilangan positif');
```

```
    ELSIF BIL < 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(BIL) ||  
        ' adalah bilangan negatif');
```

```
    END IF;
```

```
END;
```

```
/
```



PROCEDURE



Pendahuluan

- ▶ Adalah sebuah blok PL/SQL yang dapat berdiri sendiri serta dikompilasi untuk selanjutnya masuk ke dalam skema database.
- ▶ Nama procedure yang dibuat kemudian menjadi objek dengan tipe procedure. Procedure akan dieksekusi pada saat pemanggilan setelah sebelumnya dibuat terlebih dahulu



Membuat Procedure

- Bentuk umum :

```
CREATE OR REPLACE PROCEDURE
  nama_procedure
  (parameter_1 tipe_data, ... ) AS
  variabel_1 tipe_data;
  ...
BEGIN
  statemen_1;
  ...
END;
```

- Bentuk umum

```
CREATE OR RE
  nama_proc
  (parameter
  variabel_1 t
  ...
  BEGIN
    statem
    ...
  END;
```



Membuat Procedure (2)

- CREATE digunakan untuk membuat procedure baru
- REPLACE digunakan untuk mengganti isi procedure yang telah dibuat sebelumnya
- Parameter dan variable/konstanta bersifat opsional
- Bentuk umum perintah untuk mengeksekusi sebuah procedure :

EXECUTE nm_procedure(paremeter_1,...);



Contoh Procedure Tanpa Parameter

```
SET SERVEROUTPUT ON
CREATE OR REPLACE PROCEDURE hitung_luas_segitiga AS
    alas NUMBER(5);
    tinggi NUMBER(5);
    luas NUMBER(10);
BEGIN
    alas := 3;
    tinggi := 6;
    luas := (alas * tinggi) / 2;
    DBMS_OUTPUT.PUT_LINE('LUAS = ' || luas);
END;
/

.....
EXECUTE hitung_luas_segitiga
```



Contoh Procedure dengan Parameter

- Dengan Parameter Masukan

```
CREATE OR REPLACE PROCEDURE
    tambah_dua ( a IN INTEGER) AS
        hasil INTEGER(5);
BEGIN
    hasil := a + 2;
    DBMS_OUTPUT.PUT_LINE ('Hasil akhir = ' || hasil);
END;
/

.....
EXECUTE tambah_dua(4);
```



Contoh Procedure dengan Parameter (2)

- Dengan Parameter Keluaran

```
CREATE OR REPLACE PROCEDURE
```

```
    tambah_10 ( bil IN INTEGER, X OUT INTEGER) AS  
BEGIN
```

```
    X := bil + 10;
```

```
END;
```

```
/
```



Contoh Procedure dengan Parameter (2)

- Dengan Parameter Keluaran

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    hasil INTEGER;
```

```
BEGIN
```

```
    tambah_10(5, hasil);
```

```
    DBMS_OUTPUT.PUT_LINE('Hasilnya = ' || TO_CHAR(hasil));
```

```
END;
```

```
/
```



Contoh Procedure dengan Parameter (3)

- Dengan Parameter Masukan/Keluaran

```
CREATE OR REPLACE PROCEDURE
    tambah_10 ( X IN OUT INTEGER) AS
BEGIN
    X := X + 10;
END;
/
```



Contoh Procedure dengan Parameter (3)

- Dengan Parameter Masukan/Keluaran

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    Y INTEGER;
```

```
BEGIN
```

```
    Y := 15;
```

```
    tambah_10(Y);
```

```
    DBMS_OUTPUT.PUT_LINE('Hasilnya = ' || TO_CHAR(Y));
```

```
END;
```

```
/
```



Contoh Procedure Di Dalam Procedure

```
CREATE OR REPLACE PROCEDURE cetak_angka(X IN  
    INTEGER) AS  
    J INTEGER;  
BEGIN  
    FOR J IN 1..X LOOP  
        DBMS_OUTPUT.PUT_LINE(TO_CHAR(J));  
    END LOOP;  
END;  
/
```



Contoh Procedure Di Dalam Procedure (2)

```
CREATE OR REPLACE PROCEDURE panggil_proc AS  
BEGIN  
    cetak_angka(10);  
END;  
/  
  
.....  
SET SERVEROUTPUT ON  
EXECUTE panggil_proc
```



FUNCTION



Pendahuluan

- Function adalah suatu blok PL/SQL yang memiliki konsep sama dengan procedure, hanya saja pada function terdapat pengembalian nilai (return value)
- Karena function dapat mengembalikan sebuah nilai, function dapat diakses seperti layaknya sebuah variabel biasa



Membuat Function

- **Bentuk Umum :**

```
CREATE OR REPLACE FUNCTION nama_function  
(parameter_1, ...)
```

```
RETURN tipe_data AS  
    variabel_1 tipe_data;
```

```
    ...
```

```
BEGIN  
    statemen_1;
```

```
    ...
```

```
RETURN nilai_yang_dikembalikan;  
END;
```



Membuat Function (2)

- Statemen RETURN tipe_data diatas menunjukkan bahwa function akan mengembalikan nilai dengan tipe data tertentu.
- Statemen RETURN nilai_yang_dikembalikan berfungsi untuk mengembalikan nilai yang telah diproses dalam function.



Contoh Function Tanpa Parameter

```
CREATE OR REPLACE FUNCTION tulis_teks  
  RETURN VARCHAR2 AS  
  S VARCHAR2(20)  
BEGIN  
  S := 'HALLO SEMUA';  
  RETURN S;  
END;  
/
```



Contoh Function Tanpa Parameter (2)

SET SERVEROUTPUT ON

DECLARE

 X VARCHAR2(20);

BEGIN

 X := tulis_teks;

 DBMS_OUTPUT.PUT_LINE(X);

END;

/



Contoh Function Dengan Parameter

CREATE OR REPLACE FUNCTION

 pangkat (bil INTEGER, n INTEGER)

RETURN INTEGER AS

 HASIL INTEGER(10);

 I INTEGER;

BEGIN

 HASIL := 1;

 FOR I IN 1..n LOOP

 HASIL := HASIL * bil;

 END LOOP;

 RETURN HASIL;

END;

/



Contoh Function Dengan Parameter (2)

SET SERVEROUTPUT ON

DECLARE

 H INTEGER;

BEGIN

 H := pangkat(2, 3);

 DBMS_OUTPUT.PUT_LINE('Hasil = ' || TO_CHAR(H));

END;

/



Contoh Function Dalam Function

```
CREATE OR REPLACE FUNCTION kuadrat (X NUMBER)
RETURN NUMBER AS
    HASIL NUMBER(10);
BEGIN
    HASIL := X * X;
    RETURN HASIL;
END;
/
```



Contoh Function Dalam Function (2)

```
CREATE OR REPLACE FUNCTION determinan(a NUMBER, b  
    NUMBER, c NUMBER)  
RETURN NUMBER AS  
    D NUMBER(10);  
BEGIN  
    D := kuadrat(b) - (4 * a * c);  
    RETURN D;  
END;  
/
```



Contoh Function Dalam Function (3)

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    D NUMBER(10);
```

```
BEGIN
```

```
    D := determinan(1, 1, -6);
```

```
    DBMS_OUTPUT.PUT_LINE('Nilai determinan = ' ||  
    TO_CHAR(D));
```

```
END;
```

```
/
```



Terima Kasih

