PL/SQL
(PART 1)

Teknik Informatika UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@heryandi.net)

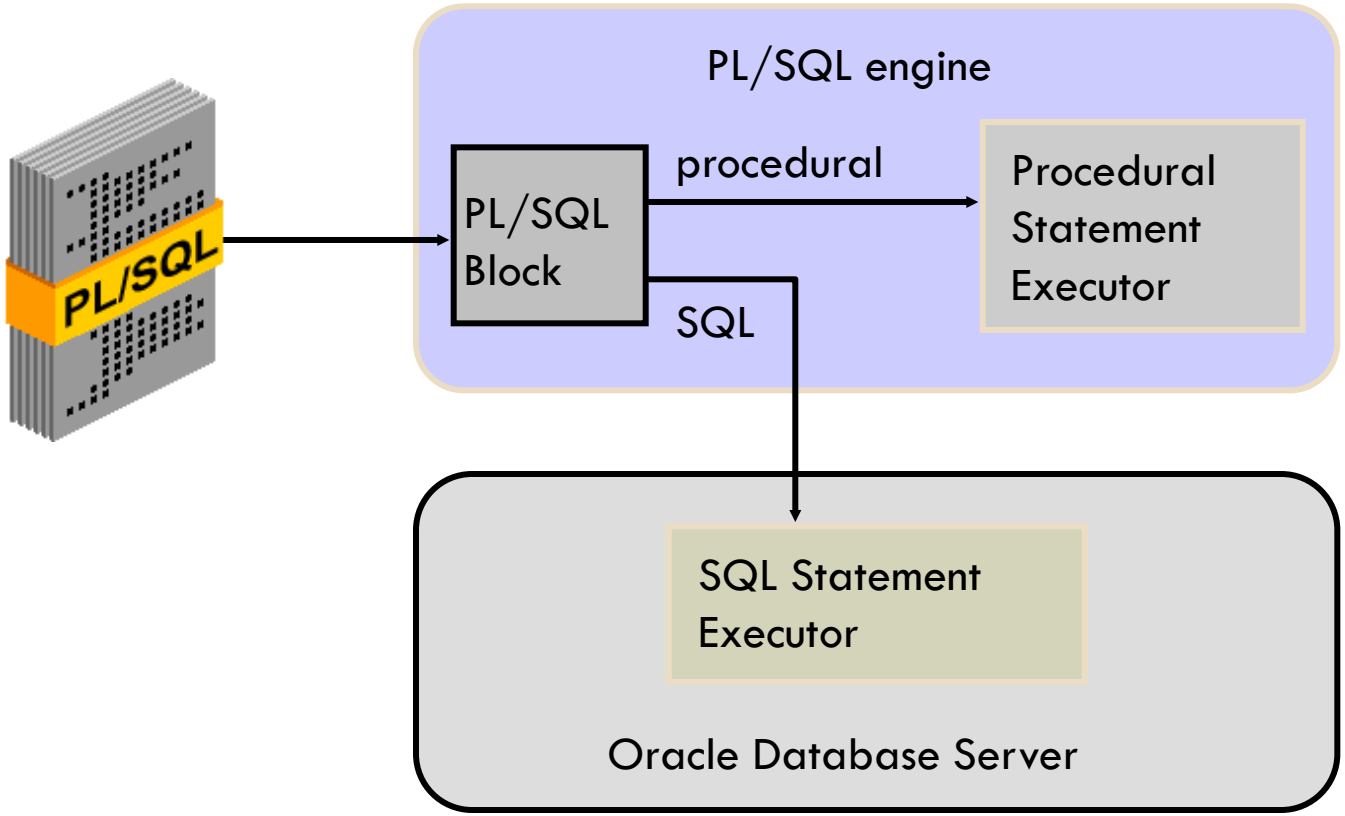
PL/SQL?

- PL/SQL:
  - PL singkatan dari Procedural Language
  - Bahasa standar untuk mengakses database relasional
  - Mengintegrasikan konstruksi prosedural dengan SQL

PL/SQL

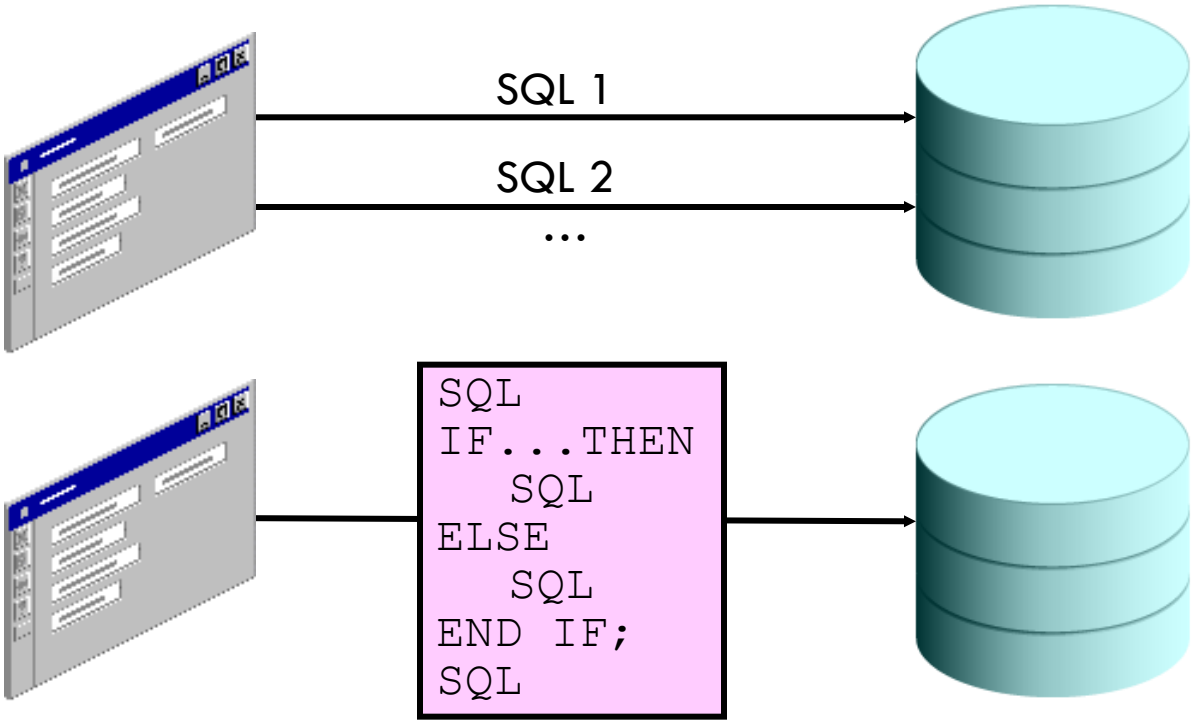
- PL/SQL:
  - Menyediakan sebuah struktur blok untuk kode yang dapat dieksekusi.
  - Menyediakan konstruksi prosedural seperti:
    - Variables, konstanta, dan type data
    - Struktur Kontrol seperti percabangan dan perulangan
    - Unit program dapat digunakan ulang. Ditulis sekali, dieksekusi banyak kali (written once and executed many times)

Lingkungan PL/SQL



Keuntungan PL/SQL

- Mengintegrasikan konstuksi prosedural dengan SQL
- Meningkatkan kinerja.

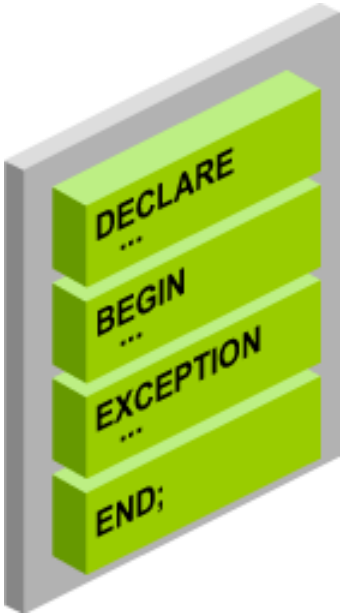


Keuntungan of PL/SQL

- Memodularisasikan pengembangan program. Sebuah pekerjaan dapat dijadikan sebuah procedure/function.
- Dapat dieksekusi di berbagai platform.
- Ada penanganan kesalahan (Exception handling)

Blok Struktur PL/SQL

- DECLARE (optional)
  - Variables, cursors, user-defined exceptions
- BEGIN (wajib)
  - SQL statements
  - PL/SQL statements
- EXCEPTION (optional)
  - Aksi ketika terjadi error
- END; (wajib)



Jenis-Jenis Blok Struktur PL/SQL

Anonymous

```
[DECLARE]

BEGIN
  --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
  --statements
  RETURN value;
[EXCEPTION]

END;
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

DEKLARASI VARIABLE PL/SQL

ORACLE DATABASE 10g

Teknik Informatika UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@heryandi.net)

Identifiers (Pengenal)

- Identifiers digunakan untuk :
  - Penamaan Variable
  - Aturan penamaan : names
    - Diawali dengan huruf
    - Dapat berisi huruf, angka, $, _, atau #.
    - Maksimal 30 karakter

Penanganan Variable dalam PL/SQL

- Variable dideklarasi dan diinisialisasi dalam bagian deklarasi.
- Digunakan dan diberi nilai baru dalam bagian executable.
- Dapat digunakan dalam parameter sub program PL/SQL.
- Digunakan untuk menyimpan data sementara hasil perhitungan sebelum dikirimkan dari function/prosedur

Deklarasi dan Inisialisasi Variable PL/SQL

ORACLE

Syntax

```
identifier [CONSTANT] datatype [NOT NULL]
     [:= | DEFAULT expr];
```

Contoh

```
DECLARE
  emp_hiredate    DATE;
  emp_deptno      NUMBER(2) NOT NULL := 10;
  location        VARCHAR2(13) := 'Atlanta';
  c_comm          CONSTANT NUMBER := 1400;
```

Deklarasi dan Inisialisasi Variable PL/SQL

1

```
SET SERVEROUTPUT ON
DECLARE
  Myname VARCHAR2(20);
BEGIN
  DBMS_OUTPUT.PUT_LINE('My name is: '||Myname);
  Myname := 'John';
  DBMS_OUTPUT.PUT_LINE('My name is: '||Myname);
END;
/
```

2

```
SET SERVEROUTPUT ON
DECLARE
  Myname VARCHAR2(20):= 'John';
BEGIN
  Myname := 'Steven';
  DBMS_OUTPUT.PUT_LINE('My name is: '||Myname);
END;
/
```

Panduan Deklarasi dan Inisialisasi Variable PL/SQL

- Ikuti aturan penamaan
- Gunakan nama yang menerangkan isi variable
- Inisialisasi variable dengan operator penugasan (:=) atau keyword DEFAULT

```
Myname VARCHAR2(20):='John';
```

```
Myname VARCHAR2(20) DEFAULT 'John';
```

- Deklarasikan satu variable per baris untuk meningkatkan keterbacaan dan memudahkan pemeliharaan

Panduan Deklarasi dan Inisialisasi Variable PL/SQL

- Hindari kesamaan nama variable sdengan nama kolom

```
DECLARE
  employee_id  NUMBER(6);
BEGIN
  SELECT    employee_id
  INTO      employee_id
  FROM      employees
  WHERE     last_name = 'Kochhar';
END;
/
```

Tipe Data

- CHAR [(maximum_length)]
- VARCHAR2 (maximum_length)
- LONG
- LONG RAW
- NUMBER [(precision, scale)]
- BINARY_INTEGER
- PLS_INTEGER
- BOOLEAN
- BINARY_FLOAT
- BINARY_DOUBLE

Tipe Data

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

Mendeklarasikan Variables

- Contoh

```
DECLARE
  emp_job          VARCHAR2(9);
  count_loop       BINARY_INTEGER := 0;
  dept_total_sal   NUMBER(9,2) := 0;
  orderdate        DATE := SYSDATE + 7;
  c_tax_rate       CONSTANT NUMBER(3,2) := 8.25;
  valid            BOOLEAN NOT NULL := TRUE;
  ...
```

Atribut %TYPE

- Atribut %TYPE
  - Digunakan untuk mendeklarasikan variable berdasarkan :
    - Pendefinisian kolom dalam database
    - Deklarasi variable lain
  - Penulisannya harus diawali dengan:
    - Nama tabel dan kolom database
    - Nama variable yang dideklarasikan
  - Keuntungan menggunakan %TYPE
    - Mencegah error karena ketidakcocokan tipe data.
    - Mengurangi penulisan tipe data variable.
    - Tidak harus mengganti deklarasi variable kalau terjadi perubahan tipe data dari tabel

Declaring Variables with the %TYPE Attribute

Syntax

```
identifier      table.column_name%TYPE;
```

Contoh

```
...
  emp_lname       employees.last_name%TYPE;
  balance         NUMBER(7,2);
  min_balance     balance%TYPE := 1000;
...
```

MENULIS STATEMENT PL/SQL

Teknik Informatika UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@heryandi.net)

Menulis PL/SQL


```
BEGIN
   dbms_output.put_line('Baris 1');
   dbms_output.put_line('Baris 2');
END;
/
```

PL/SQL procedure successfully completed.

Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Menulis PL/SQL

- Gunakan “SET SERVEROUTPUT ON” untuk menampilkan hasil put_line

```
SET SERVEROUTPUT ON
BEGIN
   dbms_output.put_line('Baris 1');
   dbms_output.put_line('Baris 2');
END;
/
```

```
Baris 1
Baris 2
PL/SQL procedure successfully completed.
```

Menulis Variable


```
DECLARE
   angka integer;
BEGIN
   angka:=5;
   dbms_output.put_line('angka sekarang '||angka);
   angka:=75;
   dbms_output.put_line('angka sekarang '||angka);
END;
/
```

```
angka sekarang 5
angka sekarang 75
PL/SQL procedure successfully completed.
```

Menulis Rumus (Aritmatika)


```
DECLARE
   jari_jari integer;
   phi   number := 22/7;
   keliling number;
BEGIN
   jari_jari:=10;
   keliling:= 2 * phi * jari_jari;
   dbms_output.put_line('Jari-jari : '||jari_jari);
   dbms_output.put_line('Keliling  : '||keliling);
END;
/
```

```
Jari-jari : 10
Keliling : 62.857142857142857142857142857142857142857142
PL/SQL procedure successfully completed.
```

Memberi Komentar

- Awali dengan 2 tanda minus untuk mengomentari 1 baris.
- Memberikan komentar untuk banyak baris bisa digunakan dengan awalan /* dan diakhiri dengan */.

- Contoh

```
DECLARE
...
annual_sal NUMBER (9,2);
BEGIN    -- Begin the executable section

/* Compute the annual salary based on the
   monthly salary input from the user */
annual_sal := monthly_sal * 12;
END;     -- This is the end of the block
/
```

Function SQL dalam PL/SQL

- Kebanyakan function dapat digunakan dalam PL/SQL kecuali:
  - DECODE
  - Group functions

Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Menulis Function


```
DECLARE
   nama varchar(40) :='Universitas Komputer Indonesia';
   panjang int;
   kapital nama%type;
BEGIN
   panjang:=length(nama);
   dbms_output.put_line('Asli : '||nama||' dengan panjang '||panjang);
   kapital:=upper(nama);
   dbms_output.put_line('Kapital : '||kapital);
   nama:=lower(nama);
   dbms_output.put_line('Huruf Kecil : '||nama);
END;
/
```

```
Asli : Universitas Komputer Indonesia dengan panjang 30
Kapital : UNIVERSITAS KOMPUTER INDONESIA
Huruf Kecil : universitas komputer indonesia
PL/SQL procedure successfully completed.
```

Operator dalam PL/SQL

- Logika
- Aritmatika
- Concatenation
- Tanda kurung untuk mengatur urutan operasi

} Sama dengan di SQL

- Operator Pangkat (**)

Operator dalam PL/SQL

- Examples
  - Increment the counter for a loop.

```
loop_count := loop_count + 1;
```

  - Set the value of a Boolean flag.

```
good_sal := sal BETWEEN 50000 AND 150000;
```

  - Validate whether an employee number contains a value.

```
valid  := (empno IS NOT NULL);
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

INTERAKSI DENGAN SERVER ORACLE

ORACLE DATABASE 10g

Teknik Informatika UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@heryandi.net)

Pernyataan SQL dalam PL/SQL

- Mengambil data dari database dengan `SELECT`.
- Melakukan DML.
- Mengatur Transaksi (`COMMIT`, `ROLLBACK`, atau `SAVEPOINT`).

Pernyataan SQL dalam PL/SQL

- Mengambil data dari database dengan statement SELECT.
- Syntax:

```
SELECT   select_list
INTO     {variable_name[, variable_name]}
FROM     table
[WHERE   condition];
```

Pernyataan SQL dalam PL/SQL

- Dibutuhkan clause `INTO`.
- Query harus hanya mereturnkan 1 baris. Untuk banyak baris gunakan Cursor.

- Contoh

```
DECLARE
 namapeg VARCHAR2(50);
BEGIN
 SELECT nama INTO namapeg
 FROM emp WHERE emp_id=100;
 DBMS_OUTPUT.PUT_LINE('Nama Pegawai : '||namapeg);
END;
/
```

```
Nama Pegawai : Steven King
PL/SQL procedure successfully completed.
```

Mengambil Data dalam PL/SQL

- Mengambil data lebih dari 1 kolom ke dalam INTO
- Contoh :

```
DECLARE
 namapeg VARCHAR2(50);
 gajipeg NUMBER;
BEGIN
 SELECT nama,gaji INTO namapeg,gajipeg
 FROM emp WHERE emp_id=100;
 DBMS_OUTPUT.PUT_LINE('Nama Pegawai : '||namapeg);
 DBMS_OUTPUT.PUT_LINE('Gaji Pegawai : '||gajipeg);
END;
/
```

```
Nama Pegawai : Steven King
Gaji Pegawai : 24000
PL/SQL procedure successfully completed.
```

Mengambil Data dalam PL/SQL

- Mengambil hasil function aggregate ke variable menggunakan INTO.
- Contoh :

```
DECLARE
   tot_gaji INTEGER;
BEGIN
     SELECT sum(gaji) INTO tot_gaji
       FROM emp;
     dbms_output.put_line('Total Gaji : '||tot_gaji);
END;
```

```
Total Gaji : 748955
PL/SQL procedure successfully completed.
```

Aturan Penamaan

- Gunakan aturan penamaan untuk menghindari ambigu dalam WHERE.
- Hindari penggunaan nama kolom dari database sebagai nama variable.
- Nama variable lokal dan parameter lebih diutamakan dari nama tabel.
- Nama kolom dari tabel database lebih diutamakan dari nama variable.

Aturan Penamaan (contoh salah)

ORACLE

```
DECLARE
  hire_date       employees.hire_date%TYPE;
  sysdate         hire_date%TYPE;
  employee_id     employees.employee_id%TYPE := 176;
BEGIN
  SELECT      hire_date, sysdate
  INTO        hire_date, sysdate
  FROM        employees
  WHERE       employee_id = employee_id;
END;
/
```

```
DECLARE
*
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
```

Aturan Penamaan (contoh benar)

```
DECLARE
  vhire_date      employees.hire_date%TYPE;
  vsysdate        hire_date%TYPE;
  vemployee_id    employees.employee_id%TYPE := 176;
BEGIN
  SELECT      hire_date, sysdate
  INTO        vhire_date, vsysdate
  FROM        employees
  WHERE       employee_id = vemployee_id;
END;
/
```

PERCABANGAN

ORACLE DATABASE 10g

Teknik Informatika UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@heryandi.net)

Jenis Percabangan

- Ada 2 jenis percabangan yang dapat dilakukan
  - IF THEN ELSE
    - IF THEN
    - IF THEN ELSE
    - IF THEN ELSEIF THEN ELSE
  - CASE

Percabangan IF THEN


```
IF kondisi THEN
   statement-statement;
END IF;
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Percabangan IF THEN ELSE


```
IF Kondisi THEN
  statement;
ELSE
  statement;

END IF;
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Percabangan IF THEN ELSEIF THEN ELSE


```
IF kondisi THEN
    statement;
ELSIF kondisi-elseif THEN
 statement;
ELSE
 statement;
END IF;
```

Contoh Percabangan IF THEN ELSE


```
DECLARE
   nilai integer := 75;
   indeks varchar(1);
BEGIN
     if nilai>=80 then
        indeks:='A';
     elsif nilai>=68 then
        indeks:='B';
     elsif nilai>=56 then
         indeks:='C';
     elsif nilai>=45 then
        indeks:='D';
     else
        indeks:='E';
     end if;
     dbms_output.put_line('Index : '||indeks);
END;
/
```

```
Index : B
PL/SQL procedure successfully completed.
```

Percabangan Dengan CASE


```
CASE  [ kolom|ekspresi]
  WHEN kondisi_1 THEN hasil_1
  WHEN kondisi_2 THEN hasil_2
  ...
  WHEN kondisi_n THEN kondisi_n
  ELSE hasil_else
END;
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Percabangan Dengan CASE


```
DECLARE
   indeks varchar(1);
   keterangan varchar(20);
BEGIN
     indeks:='B';
     keterangan:= CASE indeks
                    WHEN 'A' THEN 'BAIK SEKALI'
                    WHEN 'B' THEN 'BAIK'
                    WHEN 'C' THEN 'CUKUP'
                    WHEN 'D' THEN 'KURANG'
                    WHEN 'E' THEN 'KURANG SEKALI'
                  END;
     dbms_output.put_line('Indeks : '||indeks);
     dbms_output.put_line('Keterangan : '||keterangan);
END;
/
```

Indeks : B
Keterangan : BAIK

PERULANGAN

ORACLE DATABASE 10g

Teknik Informatika UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@heryandi.net)

Perulangan

- Perulangan bisa dilakukan dengan :
  - LOOP dan EXIT
  - FOR
  - WHILE

Perulangan dengan LOOP dan EXIT

- Perulangan dilakukan sampai kondisi_keluar bernilai TRUE.

```
LOOP
     [statement]
     EXIT WHEN kondisi_keluar;
     [statement]
END LOOP;
```

Perulangan dengan LOOP dan EXIT


```
DECLARE
    i integer;
BEGIN
     i:=1;
     LOOP
         dbms_output.put_line(i);
         EXIT WHEN i=5;
        i:=i+1;
     END LOOP;
END;
/
```

```
1
2
3
4
5
PL/SQL procedure successfully completed.
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Perulangan dengan FOR

- Perulangan dilakukan sampai counter mencapai batas atas (tanpa REVERSE) atau sampai counter mencapai batas bawah (dengan REVERSE)
- Nama_Counter tidak usah dideklarasikan

```
FOR nama_counter IN [REVERSE] batas_bawah .. batas_atas
LOOP
     statement
END LOOP;
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Perulangan dengan FOR


```
BEGIN
      dbms_output.put_line('PERULANGAN NAIK');
      FOR i IN 1..5 LOOP
          dbms_output.put_line(i);
      END LOOP;
END;
/
```

```
PERULANGAN NAIK
1
2
3
4
5
PL/SQL procedure successfully completed.
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Perulangan dengan FOR


```
BEGIN
      dbms_output.put_line('PERULANGAN TURUN');
      FOR i IN REVERSE 95..99 LOOP
          dbms_output.put_line(i);
      END LOOP;
END;
/
```

```
PERULANGAN TURUN
99
98
97
96
95
PL/SQL procedure successfully completed.
```

Perulangan dengan WHILE

- Perulangan dilakukan sampai kondisi di WHILE bernilai TRUE

```
WHILE kondisi while LOOP
      statement;
END LOOP;
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Perulangan dengan WHILE


```
DECLARE
    i integer;
BEGIN
     i:=1;
     WHILE i<=10 LOOP
         dbms_output.put_line(i);
         i:=i+3;
     END LOOP;
END;
/
```

```
1
4
7
10
PL/SQL procedure successfully completed.
```


Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)