ORACLE ACADEMY

PL/SQL
(PART 2)

ORACLE DATABASE 10g

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

CURSOR

ORACLE DATABASE 10g

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

Cursor

- Ada dua jenis cursor
  - Implicit Cursor

    Cursor yang dideklarasikan dan dikelola oleh PL/SQL setiap ada eksekusi DML atau SELECT PL/SQL.
  - Explicit Cursor

    Cursor yang dideklarasikan dan dikelola oleh programmer.
- Server oracle secara implicit akan membuka sebuah cursor untuk setiap proses SQL statement yang tidak dideklarasikan dalam deklarasi cursor.
- Dalam PL/SQL, anda dapat memanggil cursor implicit paling terakhir dengan menggunakan cursor bernama **SQL**.

Cursor

- Beberapa fungsi dari cursor adalah
  - Melakukan proses terhadap setiap baris yang diambil dari SELECT yang menghasilkan banyak baris.
  - Memeriksa kondisi setelah DML dieksekusi. Anda dapat melihat berapa data yang telah didelete dll.

Cursor Implicit

- Cursor jenis ini dideklarasikan dan dikelola oleh oracle.
- Cursor jenis ini tidak diberinama, tetapi biasa dipanggil dengan **SQL**.
- Cursor SQL mempunyai atribut. Atribut tersebut bisa digunakan dalam percabangan atau kondisi exit dari suatu perulangan.
- Atribut cursor SQL adalah :
  - SQL%FOUND : Bernilai true jika eksekusi sebelumnya mereturnkan minimal 1 baris
  - SQL%NOTFOUND : Berlawanan dengan SQL%FOUND
  - SQL%ROWCOUNT : Memberikan nilai angka yang berisi banyak data yang terpengaruhi (affected row) oleh sql sebelumnya.

Contoh Cursor Implicit (1)


```
BEGIN
   DELETE FROM emp where gaji>10000;
   IF(SQL%NOTFOUND) THEN
        dbms_output.put_line('SQL%NOTFOUND : TRUE');
   ELSE
        dbms_output.put_line('SQL%NOTFOUND : FALSE');
   END IF;
   IF(SQL%FOUND) THEN
        dbms_output.put_line('SQL%FOUND : TRUE');
   ELSE
        dbms_output.put_line('SQL%FOUND : FALSE');
   END IF;
   dbms_output.put_line('SQL%ROWCOUNT : '|| SQL%ROWCOUNT);
   ROLLBACK; -- Mengembalikan ke kondisi asal, sekedar test
END;
```

SQL%NOTFOUND : FALSE
SQL%FOUND : TRUE
SQL%ROWCOUNT : 15

Contoh Cursor Implicit (2)


```
BEGIN
   DELETE FROM emp where gaji>30000;
   IF(SQL%NOTFOUND) THEN
        dbms_output.put_line('SQL%NOTFOUND : TRUE');
   ELSE
        dbms_output.put_line('SQL%NOTFOUND : FALSE');
   END IF;
   IF(SQL%FOUND) THEN
        dbms_output.put_line('SQL%FOUND : TRUE');
   ELSE
        dbms_output.put_line('SQL%FOUND : FALSE');
   END IF;
   dbms_output.put_line('SQL%ROWCOUNT : '|| SQL%ROWCOUNT);
   ROLLBACK; -- Mengembalikan ke kondisi asal, sekedar test
END;
```

SQL%NOTFOUND : TRUE
SQL%FOUND : FALSE
SQL%ROWCOUNT : 0

Contoh Cursor Implicit (3)


```
DECLARE
   vemp_id emp.emp_id%TYPE;
BEGIN
   SELECT emp_id INTO vemp_id FROM emp WHERE gaji>20000;
   IF(SQL%FOUND) THEN
        dbms_output.put_line('SQL%FOUND : TRUE');
   ELSE
        dbms_output.put_line('SQL%FOUND : FALSE');
   END IF;
   dbms_output.put_line('SQL%ROWCOUNT : '|| SQL%ROWCOUNT);
END;
```

SQL%FOUND : TRUE
SQL%ROWCOUNT : 1

Contoh Cursor Implicit (4)


```
DECLARE
   vemp_id emp.emp_id%TYPE;
BEGIN
   SELECT emp_id INTO vemp_id FROM emp WHERE gaji>10000;
   IF(SQL%FOUND) THEN
        dbms_output.put_line('SQL%FOUND : TRUE');
   ELSE
        dbms_output.put_line('SQL%FOUND : FALSE');
   END IF;
   dbms_output.put_line('SQL%ROWCOUNT : '|| SQL%ROWCOUNT);
END;
```

```
DECLARE
*
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 4
```


Cursor Explicit

- Cursor jenis ini dideklarasikan dan dikelola secara manual oleh programmer.
- Langkah-langkah penggunaan cursor jenis ini adalah :
  - Deklarasikan cursor dalam block DECLARE
  - OPEN cursor
  - FETCH baris dari CURSOR
  - CLOSE cursor

Atribut Cursor Explicit

- Atribut yang dimiliki cursor explicit mirip dengan cursor implicit. Atributnya adalah
  - %FOUND : Bernilai true jika fetch terakhir mengembalikan sebuah baris.
  - %NOTFOUND : Berlawanan dengan SQL%FOUND
  - %ROWCOUNT : Memberikan nilai berupa angka yang berisi banyak data yang telah di-fetch.
  - %ISOPEN : Bernilai true jika cursor telah dibuka dan belum ditutup.

DECLARE Cursor Explicit

Sintak:

```
CURSOR nama_cursor IS
     pernyataan_select;
```

Contoh

```
DECLARE
  CURSOR emp_cursor IS
     SELECT emp_id, gaji FROM emp
     WHERE gaji>10000;
```

```
DECLARE
  vgaji NUMBER:= 15000;
  CURSOR emp_cursor IS
     SELECT emp_id, gaji FROM emp
     WHERE gaji >= vgaji;
...
```

OPEN dan CLOSE Cursor Explicit


```
DECLARE
  CURSOR emp_cursor IS
     SELECT emp_id, gaji FROM emp
     WHERE gaji>10000;
BEGIN
  OPEN emp_cursor;
  ...
  ...
  CLOSE emp_cursor;
END;
```

- OPEN cursor berarti mengeksekusi statement SELECT yang terkait, dan memindahkankan pointer ke baris pertama.
- CLOSE cursor berarti menutup cursor.

FETCH baris dari Cursor Explicit


```
DECLARE
  CURSOR emp_cursor IS
     SELECT nama, gaji FROM emp
     WHERE gaji>15000;
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
     OPEN emp_cursor;
     FETCH emp_cursor INTO vnama, vgaji;
     dbms_output.put_line(vnama || ' dengan gaji ' || vgaji);
     CLOSE emp_cursor;
END;
```

Hasil Eksekusi

Steven King dengan gaji 24000

- FETCH pada contoh di atas hanya mengambil data pertama. Ini dikarenakan fetch tidak dilakukan dalam sebuah perulangan, jadi hanya mengambil data sebanyak 1 kali.

FETCH baris dari Cursor Explicit


```
DECLARE
  CURSOR emp_cursor IS
     SELECT nama, gaji FROM emp
     WHERE gaji>15000;
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
     OPEN emp_cursor;
     LOOP
       FETCH emp_cursor INTO vnama, vgaji;
       EXIT WHEN emp_cursor%NOTFOUND;
       dbms_output.put_line(vnama || ' dengan gaji ' || vgaji);
     END LOOP;
     CLOSE emp_cursor;
END;
```

Hasil Eksekusi

```
Steven King dengan gaji 24000
Neena Kochhar dengan gaji 17000
Lex De Haan dengan gaji 17000
```

Mengambil Seluruh Kolom dari FETCH


```
DECLARE
  CURSOR emp_cursor IS
     SELECT nama, gaji FROM emp
     WHERE gaji>15000;
  vemp emp_cursor%ROWTYPE;
BEGIN
     OPEN emp_cursor;
     LOOP
       FETCH emp_cursor INTO vemp;
       EXIT WHEN emp_cursor%NOTFOUND;
       dbms_output.put_line(emp_cursor%ROWCOUNT ||'.'||
                            vemp.nama || ' dengan gaji ' ||
                            vemp.gaji);
     END LOOP;
     CLOSE emp_cursor;
END;
```

Hasil Eksekusi

1.Steven King dengan gaji 24000
2.Neena Kochhar dengan gaji 17000
3.Lex De Haan dengan gaji 17000

Keterangan :

1. Deklarasi **emp_cursor** sebagai sebuah cursor dengan sql : **`SELECT nama, gaji FROM emp WHERE gaji>15000`**
2. Deklarasi **vemp emp_cursor%ROWTYPE** berarti deklarasi suatu variable yang bertipe record(baris) yang dihasilkan dari cursor emp_cursor (fieldnya nama dan gaji).
3. **FETCH emp_cursor INTO vemp** berarti mengisikan baris hasil fetch ke variable record vemp.
4. Pada put_line ada emp_cursor%ROWCOUNT yang akan menampilkan nomor record.
5. Pada put_line ada vemp.nama yang berarti field nama dari record vemp, dan vemp.gaji yang berarti field gaji dari record vemp.

Perulangan LOOP untuk Cursor


```
DECLARE
  CURSOR emp_cursor IS
     SELECT nama, gaji FROM emp
     WHERE gaji>15000;
BEGIN
     FOR vemp IN emp_cursor
     LOOP
       dbms_output.put_line(vemp.nama || ' dengan gaji ' ||
vemp.gaji);
     END LOOP;
END;
```

Hasil Eksekusi

```
Steven King dengan gaji 24000
Neena Kochhar dengan gaji 17000
Lex De Haan dengan gaji 17000
```

- Tidak usah mendeklarasikan variable untuk menampung data dari fetch (karena tidak ada fetch).
- Variable penampung (vemp) data tidak usah dideklarasikan.
- Tidak usah OPEN atau CLOSE cursor.

Perulangan LOOP untuk Cursor SubQuery

ORACLE 18

```
BEGIN
  FOR vemp IN (SELECT nama, gaji
               FROM emp WHERE gaji>15000)
  LOOP
    dbms_output.put_line(vemp.nama || ' gaji ' || vemp.gaji);
  END LOOP;
END;
```

Steven King gaji 24000
Neena Kochhar gaji 17000
Lex De Haan gaji 17000

- Cursor diganti dengan sql SELECT secara langsung.

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

Keterangan :

1. Deklarasi **emp_cursor** sebagai sebuah cursor dengan sql : **`SELECT nama, gaji FROM emp WHERE gaji>15000`**
2. Deklarasi **vemp emp_cursor%ROWTYPE** berarti deklarasi suatu variable yang bertipe record(baris) yang dihasilkan dari cursor emp_cursor (fieldnya nama dan gaji).
3. **FETCH emp_cursor INTO vemp** berarti mengisikan baris hasil fetch ke variable record vemp.
4. Pada put_line ada vemp.nama yang berarti field nama dari record vemp, dan vemp.gaji yang berarti field gaji dari record vemp.

Cursor dengan Parameter


```
DECLARE
  CURSOR emp_cursor(vgaji NUMBER) IS
     SELECT nama, gaji FROM emp
     WHERE gaji>vgaji;
  vemp emp_cursor%ROWTYPE;
BEGIN
     OPEN emp_cursor(20000);
     LOOP
       FETCH emp_cursor INTO vemp;
       EXIT WHEN emp_cursor%NOTFOUND;
       dbms_output.put_line(emp_cursor%ROWCOUNT ||'.'||
                            vemp.nama || ' dengan gaji ' ||
                            vemp.gaji);
     END LOOP;
     CLOSE emp_cursor;
END;
```

Hasil Eksekusi

1.Steven King dengan gaji 24000

Cursor dengan Parameter


```
DECLARE
  CURSOR emp_cursor(vgaji NUMBER) IS
     SELECT nama, gaji FROM emp
     WHERE gaji>vgaji;
  vemp emp_cursor%ROWTYPE;
BEGIN
     OPEN emp_cursor(15000);
     LOOP
       FETCH emp_cursor INTO vemp;
       EXIT WHEN emp_cursor%NOTFOUND;
       dbms_output.put_line(emp_cursor%ROWCOUNT ||'.'||
                            vemp.nama || ' dengan gaji ' ||
                            vemp.gaji);
     END LOOP;
     CLOSE emp_cursor;
END;
```

Hasil Eksekusi

1.Steven King dengan gaji 24000
2.Neena Kochhar dengan gaji 17000
3.Lex De Haan dengan gaji 17000

Latihan Menggunakan Cursor

- Tabel Menu : Nama menu yang tersedia
- Tabel Bahan : Bahan masakan yang tersedia
- Tabel Kebutuhan_Masak : Kebutuhan penggunaan bahan masakan untuk membuat menu.

Latihan Menggunakan Cursor

Contoh Data:

MENU	
IdMenu	NamaMenu
1	Telur Ceplok
2	Telur Cornet
3	Telur Dadar
4	Omelet

BAHAN		
IdBahan	Nama	Tersedia
1	Telur	50
2	Cornet	2000
3	Daun Bawang	2000
4	Mie	1500
5	Minyak Goreng	3000

Kebutuhan_Masak		
IdMenu	IdBahan	Dibutuhkan
1	1	1
1	5	100
2	1	1
2	2	200
2	5	100
3	1	1
3	3	100
4	1	2
4	3	100
4	4	200
4	5	200

Latihan Menggunakan Cursor

- Buatlah sebuah PL/SQL yang akan mengupdate data tabel bahan pada kolom Tersedia, jika suatu menu dibuat.
- Contoh : Jika ada pembuatan menu “Telor Kornet”, maka akan mengurangi persediaan Telur sebanyak 1(biji), Cornet 200(gram), Minyak goreng 100(ml). Banyaknya pengunaan bahan dapat di-select dari tabel Kebutuhan_masak.

PENANGANAN EXCEPTION

ORACLE DATABASE 10g

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

Contoh Exception


```
DECLARE
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
   SELECT nama,gaji INTO vnama, vgaji FROM emp
       WHERE gaji>20000;
   dbms_output.put_line(vnama || ' - ' || vgaji);
END;
```

Steven King - 24000

Tidak muncul error karena pegawai yang bergaji diatas 20000 ditemukan hanya 1 pegawai.

Contoh Exception

Gaji yang dicari diganti dengan 50000. Data pasti tidak ditemukan.

```
DECLARE
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
   SELECT nama,gaji INTO vnama, vgaji FROM emp
       WHERE gaji>50000;
   dbms_output.put_line(vnama || ' - ' || vgaji);
END;
```

```
DECLARE
*

ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 5
```


Contoh Exception

Gaji yang dicari diganti dengan 10000. Data pasti ditemukan, tetapi lebih dari 1 (melanggar aturan SELECT INTO).

```
DECLARE
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
   SELECT nama,gaji INTO vnama, vgaji FROM emp
       WHERE gaji>10000;
   dbms_output.put_line(vnama || ' - ' || vgaji);
END;
```

```
DECLARE
*
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 5
```


Contoh Exception


```
DECLARE
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
   SELECT nama,gaji INTO vnama, vgaji FROM emp
       WHERE gaji>50000;
   dbms_output.put_line(vnama || ' - ' || vgaji);
   EXCEPTION
     WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Data tidak ditemukan');
     WHEN TOO_MANY_ROWS THEN
        dbms_output.put_line('Data ditemukan lebih dari 1');
     WHEN OTHERS THEN
        dbms_output.put_line('Ada Error '||SQLERRM);
 END;
```

```
Data tidak ditemukan
PL/SQL procedure successfully completed.
```

Contoh Exception


```
DECLARE
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
   SELECT nama,gaji INTO vnama, vgaji FROM emp
       WHERE gaji>10000;
   dbms_output.put_line(vnama || ' - ' || vgaji);
   EXCEPTION
     WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Data tidak ditemukan');
     WHEN TOO_MANY_ROWS THEN
        dbms_output.put_line('Data ditemukan lebih dari 1');
     WHEN OTHERS THEN
        dbms_output.put_line('Ada Error '||SQLERRM);
 END;
```

```
Data ditemukan lebih dari 1
PL/SQL procedure successfully completed.
```


Contoh Exception


```
DECLARE
  vnama emp.nama%TYPE;
  vgaji emp.gaji%TYPE;
BEGIN
   SELECT nama,gaji INTO vnama, vgaji FROM emp
       WHERE gaji>'abc10000';
   dbms_output.put_line(vnama || ' - ' || vgaji);
   EXCEPTION
     WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Data tidak ditemukan');
     WHEN TOO_MANY_ROWS THEN
        dbms_output.put_line('Data ditemukan lebih dari 1');
     WHEN OTHERS THEN
        dbms_output.put_line('Ada Error '||SQLERRM);
 END;
```

```
Ada Error ORA-01722: invalid number
PL/SQL procedure successfully completed.
```


Exception dari Oracle

- Ada exception yang terdefinisikan oleh Oracle. Beberapa diantaranya :
  - NO_DATA_FOUND
  - TOO_MANY_ROWS
  - ZERO_DIVIDE
  - INVALID_NUMBER
  - LOGIN_DENIED
  - NOT_LOGGED_ON
  - VALUE_ERROR

Keterangan :

1. NO_DATA_FOUND : Terjadi ketika ada SELECT INTO tidak menemukan 1 pun baris data.
2. TOO_MANY_ROWS : Terjadi ketika ada SELECT INTO menemukan baris yang sesuai lebih dari 1.
3. ZERO_DIVIDE : Terjadi ketika ada pembagian dengan angka nol.
4. INVALID_NUMBER : Terjadi kesalahan konversi dari string ke number
5. LOGIN_DENIED : Terjadi ketika login salah user name atau password
6. NOT_LOGGED_ON : Terjadi ketika belum melakukan login.
7. VALUE_ERROR : Terjadi ketika terjadi error aritmatika, atau konversi.

Menangkap Exception

- Syntax:

```
EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
        . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement2;
        . . .]
  [WHEN OTHERS THEN
    statement3;
        . . .]
```

- Exception OTHERS digunakan untuk menangkap exception yang tidak didefinisikan pada WHEN sebelumnya. Dalam percabangan, OTHERS mirip dengan ELSE.

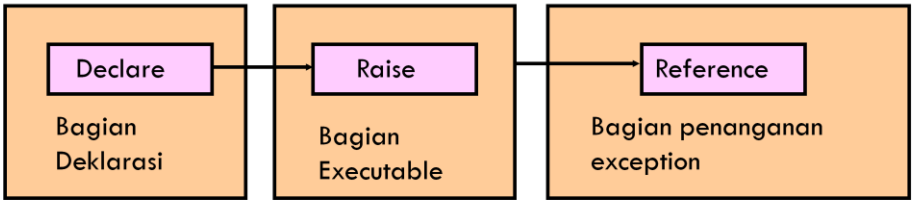
Function Menangkap Exception

- SQLCODE: Mengembalikan angka yang berisi kode error
- SQLERRM: Mengembalikan pesan yang berhubungan dengan nomor error

```
DECLARE
  vnama emp.nama%TYPE;
BEGIN
   SELECT nama INTO vnama FROM emp WHERE gaji>'abc10000';
   dbms_output.put_line(vnama);
   EXCEPTION
      WHEN OTHERS THEN
        dbms_output.put_line('Error '||SQLCODE||': ' ||SQLERRM);
END;
```

Error -1722: ORA-01722: invalid number

Exception Buatan User



Nama Exception

Bangkitkan exception dengan perintah RAISE

Penanganan exception yang dibangkitkan

Exception Buatan User


```
DECLARE
  pegawai_tidak_ada EXCEPTION;
BEGIN
   DELETE FROM emp WHERE emp_id=1234;
   IF SQL%NOTFOUND THEN
        RAISE pegawai_tidak_ada;
   END IF;
   dbms_output.put_line('Data telah dihapus');
   ROLLBACK;
   EXCEPTION
       WHEN pegawai_tidak_ada THEN
            dbms_output.put_line('Pegawai tidak ditemukan');
 END;
```

Pegawai tidak ditemukan

Exception Buatan User


```
DECLARE
  jk char(1) :='X';
  err_kelamin EXCEPTION;
BEGIN
   IF jk NOT IN ('L','P') THEN
        RAISE err_kelamin;
   END IF;
   dbms_output.put_line('Jenis Kelamin Valid');
   EXCEPTION
       WHEN err_kelamin THEN
            dbms_output.put_line('Jenis kelamin salah');
 END;
```

Jenis kelamin salah

Procedure RAISE_APPLICATION_ERROR

- Sintak:

```
raise_application_error (kode_error, pesan_error);
```

- Procedure ini dapat digunakan untuk menampilkan pesan kesalahan yang dibuat oleh user (user defined error message) dari dalam sub program.
- Procedure ini dapat mereportkan kesalahan pada aplikasi anda untuk menghindari kesalahan yang tidak tertangani/dikenal oleh aplikasi.
- Kode_Error berisi kode error yang akan digunakan. Kode yang boleh digunakan oleh user adalah -20000 s/d -20999.
- Pesan error merupakan pesan error yang ingin ditampilkan. Panjang pesan maksimal 2048 karakter.
- Procedure ini boleh digunakan dalam blok executable atau dalam blok exception

Procedure RAISE_APPLICATION_ERROR


```
DECLARE
  jk char(1) :='X';
BEGIN
   IF jk NOT IN ('L','P') THEN
        RAISE_APPLICATION_ERROR(-20000,'Jenis Kelamin
Invalid');
   END IF;
   dbms_output.put_line('Jenis Kelamin Valid');
 END;
```

```
DECLARE
*
```

ERROR at line 1:
ORA-20000: Jenis Kelamin Invalid
ORA-06512: at line 6

Procedure RAISE_APPLICATION_ERROR


```
DECLARE
  jk char(1) :='X';
  err_kelamin EXCEPTION;
BEGIN
   IF jk NOT IN ('L','P') THEN
        RAISE err_kelamin;
   END IF;
   dbms_output.put_line('Jenis Kelamin Valid');
   EXCEPTION
     WHEN err_kelamin THEN
       RAISE_APPLICATION_ERROR(-20000,'Jenis Kelamin Invalid');
END;
```

```
DECLARE
*

ERROR at line 1:
ORA-20000: Jenis Kelamin Invalid
ORA-06512: at line 6
```


STORED PROCEDURE

ORACLE DATABASE 10g

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

Definisi Stored Procedure

- Stored procedure adalah sebuah blok PL/SQL yang diberi nama dan kadang memiliki parameter.
- Sintak

```
CREATE [OR REPLACE] PROCEDURE nama_procedure
 [(argument1 tipedata1,
   argument2 tipedata2,
  . . .)]
IS|AS
badan_procedure;
```


Contoh Stored Procedure


```
CREATE OR REPLACE
PROCEDURE Tambah_Dept(vdept_id dept.department_id%TYPE,
                      vnama dept.nama%TYPE)
IS
BEGIN
      IF vdept_id IS NULL THEN
         RAISE_APPLICATION_ERROR(-20000,'Department ID tidak
boleh kosong');
      END IF;
      IF vnama IS NULL THEN
         RAISE_APPLICATION_ERROR(-20001,'Nama Departemen tidak
boleh kosong');
      END IF;
      INSERT INTO dept VALUES(vdept_id,TRIM(vnama));
      COMMIT;
END;
```

Eksekusi Stored Procedure


```
EXEC Tambah_Dept(501,'Dept 501');
```

PL/SQL procedure successfully completed.

```
SELECT * FROM dept WHERE department_id=501
```

DEPARTMENT_ID	NAMA
501	Dept 501

Eksekusi Stored Procedure


```
EXEC Tambah_Dept(NULL,'Dept 501')
```

```
BEGIN Tambah_Dept(NULL,'Dept 501'); END;

*
```

ERROR at line 1:
ORA-20000: Department ID tidak boleh kosong
ORA-06512: at "DB97025.TAMBAH_DEPT", line 6
ORA-06512: at line 1

Eksekusi Stored Procedure


```
EXEC Tambah_Dept(502,NULL)
```

```
BEGIN Tambah_Dept(502,NULL); END;

*

ERROR at line 1:
ORA-20001: Nama Departemen tidak boleh kosong
ORA-06512: at "DB97025.TAMBAH_DEPT", line 9
ORA-06512: at line 1
```


Contoh Stored Procedure (2)

- Menghapus data di tabel Dept disertai dengan penghapusan semua pegawai di department yang dihapus

```
CREATE OR REPLACE
PROCEDURE Hapus_Dept(vdept_id dept.department_id%TYPE)
IS
BEGIN
   DELETE FROM dept WHERE department_id=vdept_id;
   IF SQL%ROWCOUNT=0 THEN
     RAISE_APPLICATION_ERROR(-20000,'Department ID tidak ada');
   ELSE
       DELETE FROM emp WHERE department_id=vdept_id;
   END IF;
   COMMIT;
END;
```


Eksekusi Stored Procedure (2)


```
EXEC Hapus_Dept(505);
```

```
BEGIN hapus_dept(505); END;

*
```

ERROR at line 1:
ORA-20000: Department ID tidak ada
ORA-06512: at "DB97025.HAPUS_DEPT", line 6
ORA-06512: at line 1

Eksekusi Stored Procedure (2)


```
SELECT (SELECT COUNT(*) FROM dept) RecordDept,
       (SELECT COUNT(*) FROM emp) RecordEmp FROM DUAL
```

RECORDDEPT	RECORDEMP
27	108

```
EXEC Hapus_Dept(110);
```

PL/SQL procedure successfully completed.

```
SELECT (SELECT COUNT(*) FROM dept) RecordDept,
       (SELECT COUNT(*) FROM emp) RecordEmp FROM DUAL
```

RECORDDEPT	RECORDEMP
26	107

Keterangan :

1. Query 1 menampilkan banyak record di tabel Dept dan Emp sebelum dihapus
2. Query 2 melakukan eksekusi procedure Hapus_Dept(101)
3. Query 3 menampilkan banyak record di tabel Dept dan Emp setelah dihapus.

Dari Query 1 dan Query 3 dapat dilihat perbedaannya.

STORED FUNCTION

ORACLE DATABASE 10g

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

Definisi Stored Function

- Stored Function adalah suatu blok PL/SQL yang diberi nama dan mempunyai parameter masukan dan menghasilkan sebuah nilai kembalian (return value).
- Sintak

```
CREATE [OR REPLACE] FUNCTION nama_function
 [(argument1 tipedata1,
  argument2 tipedata2, . . .)]
RETURN tipedata_return
IS|AS
badan_function;
```


Contoh Function

- Function menghitung Nilai Akhir

```
CREATE OR REPLACE
FUNCTION Hitung_NA(quiz NUMBER, uts NUMBER, uas NUMBER)
    RETURN NUMBER
IS
BEGIN
     RETURN 0.2*quiz+0.3*uts+0.5*uas;
END;
```


Eksekusi Stored Function

- Mencari nilai akhir dengan
  - Quis : 70
  - UTS : 90
  - UAS : 70

```
SELECT Hitung_NA(70,90,70) NilaiAkhir FROM DUAL
```

NILAIAKHIR
76

Contoh Function (2)

- Function penentuan Index nilai dari nilai akhir

```
CREATE OR REPLACE
FUNCTION Hitung_Index(nilai number) RETURN char IS
   indeks char(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;
    RETURN indeks;
END;
```

Eksekusi Stored Function (2)

- Mencari index dari nilai akhir 78

```
SELECT Hitung_Index(78) Indeks FROM DUAL
```

INDEKS
B

- Mencari index nilai dari nilai
  - Quis : 40
  - UTS : 80
  - UAS : 70

```
SELECT Hitung_Index(Hitung_NA(40,80,70)) Indeks FROM DUAL
```

INDEKS
C

Contoh Function (3)

- Function untuk menghilangkan spasi yang tidak perlu dari suatu string. Spasi yang tidak perlu adalah spasi di awal dan di akhir, serta spasi ganda di antara kata.

```
CREATE OR REPLACE
FUNCTION Remove_Space(st varchar) RETURN varchar IS
   temp VARCHAR(255);
BEGIN
   temp:=TRIM(st);
   LOOP
          temp:=REPLACE(temp,'  ',' ');
          EXIT WHEN INSTR(temp,'  ')=0;
   END LOOP;
   RETURN temp;
END;
```


Eksekusi Stored Function

- Menghilangkan spasi dari string ‘ AB CD E ‘

```
SELECT Length(' AB   CD   E    ') Asli,
    Length(Remove_Space(' AB   CD   E    ')) SetelahRemove
FROM DUAL
```

ASLI	SETELAHREMOVE
16	7