MODUL VI
PL / SQL

Tujuan Praktikum :

1. Memahami Stuktur Program PL/SQL.
2. Memahami sintaks-sintaks dasar PL/SQL.
3. Menggunakan PL/SQL dalam SQL*Plus.
4. Memahami statemen Kondisional (IF..Then..Else).
5. Memahami statemen Pengulangan(LOOP).
6. Memahami Penanganan Error (Exception).

I. Definisi PL/SQL

PL/SQL (*Procedural Languange/Structure Query Languange*) merupakan sebuah bahasa yang menggabungkan antara kekuatan dan fleksibilitas dari SQL / *non prosedural* dengan bentuk-bentuk prosedural (*procedural construct).*

Pada PL/SQL kamu dapat menuliskan perintah-perintah, seperti halnya pendeklarasian variabel, statement kondisional (IF..Then..Else), pengulangan (LOOP), dsb. **PL/SQL tidak mendukung Data Definition Language(DDL)** seperti CREATE TABLE, ALTER TABLE atau DROP TABLE, selain itu juga tidak mendukung Data Control Language(DCL) seperti GRANT atau REVOKE.

Beberapa karakteristik dari PL/SQL[5], antara lain :

1. User friendly
2. Bahasa standard dan portabel dalam pengembangan Oracle
3. Merupakan bahasa bawaan dari Oracle yang berarti PL/SQL dapat dieksekusi di host maupun di client
4. Integritas yang tinggi

II. Struktur Blok PL/SQL

Setiap unit PL/SQL terdiri dari satu blok atau lebih.
Struktur umum untuk membuat sebuah blok PL/SQL di Oracle adalah[6] :

```
DECLARE
     Bagian Deklarasi
BEGIN
     Bagian Eksekusi
EXCEPTION
     Bagian Exception
END;
```

Blok tersebut dapat berdiri sendiri atau berada dalam blok lain, yang disebut dengan **subblok**.

```
DECLARE
     Bagian Deklarasi
BEGIN
     DECLARE
       Bagian Deklarasi
     BEGIN
       Bagian Eksekusi
     EXCEPTION
              Bagian Exception
       END;
EXCEPTION
     Bagian Exception
END;
```

Seperti terlihat diatas, blok PL/SQL lengkap terdiri dari 3 bagian yaitu :

1. Declaration Section

Digunakan untuk mendefinisikan / mendeklarasikan variable, konstanta, cursor, dan seluruh exception yang didefinisikan oleh user yang akan digunakan pada bagian eksekusi[7]. Penulisan blok ini dimulai dengan menulis **DECLARE.**

```
DECLARE
   v_first_name    VARCHAR2(35);
   v_last_name    VARCHAR2(35);
   v_counter    NUMBER := 0;
```

2. Executable Section

Digunakan untuk mengeksekusi / menjalankan blok perintah PL/SQL, seperti pengulangan, percabangan, perintah SQL dan perintah CURSOR[7]. Berisi statement SQL untuk memanipulasi data pada basis data, dan statement PL/SQL untuk memanipulasi data dalam blok.

```
BEGIN
     SELECT   first_name,   last_name
     INTO   v_first_name,   v_last_name
     FROM   student
     WHERE   student_id = 123;
     DBMS_OUTPUT.PUT_LINE     ('Student     name:     '||v_first_name||'
'||v_last_name);
END;
```

3. Exception Section

Merupakan bagian yang akan diaktifkan bila terjadi kesalahan atau pengecualian pada saat menjalankan program PL/SQL (executable statement)[7]. Exception Section terdiri dari predefined dan user-defined. Sebagai contoh exception predefined NO_DATA_FOUND akan diaktifkan bila perintah DML SQL tidak menemukan data dalam clausa Where.

```
EXCEPTION
      WHEN   NO_DATA_FOUND   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no student with '||'student id 123');
END;
```

Penggunaan PL/SQL :

Contoh 1

```
BEGIN
     dbms_output.put_line('Hello World');
END;
/                ← tanda ini ('/') wajib ada setelah('END;'), untuk
mengeksekusi PL/SQL
```

Untuk menampilkan output dari blok PL/SQL, maka terlebih dahulu **menset SERVEROUTPUT menjadi ON**. Hal ini cukup dilakukan sekali saja, saat membuka SQLplus.

```
SET SERVEROUTPUT ON;
```

Hasil Contoh 1:

```
Hello World
```

Blok PL/SQL selalu **diawali** dengan **kata kunci BEGIN** untuk memulai execution section **dan diakhiri dengan END**, dimana keduanya **bersifat wajib** (executable section harus ada pada sebuah blok). Sedangkan **declare section** dan **exception section bersifat opsional**. Contoh 1 merupakan contoh paling sederhana.

Berikut merupakan blok PL/SQL dengan DECLARATION, EXECUTABLE, dan EXCEPTION:

Contoh 2[7]

```
DECLARE
   v_first_name    VARCHAR2(35);
```

```
   v_last_name   VARCHAR2(35);
BEGIN
     SELECT   first_name,   last_name
     INTO   v_first_name,   v_last_name
     FROM   student
     WHERE   student_id = 123;
     DBMS_OUTPUT.PUT_LINE ('Student name: '||v_first_name||' '||v_last_name);
EXCEPTION
      WHEN   NO_DATA_FOUND   THEN
      DBMS_OUTPUT.PUT_LINE ('There is no student with '||'student id 123');
END;
/
```

Contoh 3[7]

```
DECLARE
   v_name VARCHAR2(50);
   v_total NUMBER;
BEGIN
     SELECT   i.first_name||' '||i.last_name,   COUNT(*)
     INTO   v_name,   v_total
     FROM   instructor i,  section s
     WHERE   i.instructor_id = s.instructor_id  AND   i.instructor_id = 102
     GROUP   BY   i.first_name||' '||i.last_name;
     DBMS_OUTPUT.PUT_LINE('Instructor   '||v_name||'   teaches   '||v_total||'
courses');
EXCEPTION
     WHEN   NO_DATA_FOUND   THEN
     DBMS_OUTPUT.PUT_LINE ('There is no such instructor');
END;
/
```

Bentuk lain dari Blok PL/SQL[7] :

- Anonymous Block
  Merupakan blok PL/SQL yang tidak bernama, tidak dapat disimpan dalam database dan biasanya dibangun secara dinamik dan dieksekusi hanya sekali.
- Named Block
  Merupakan anonymous block dengan label yang diberi nama. Named blok dapat dibagi:
  - SubPrograms

  Merupakan procedure, paket, dan fungsi-fungsi yang disimpan di database. Blok-blok seperti ini dapat digunakan berulang kali dan dieksekusi secara explisit melalui *call* (pemanggilan prosedur, paket atau fungsi tersebut).
  - Triggers
    Merupakan *named blocks* yang juga disimpan di database. Blok ini dapat digunakan berulang kali dan dieksekusi secara implisit dimanapun *event* ***trigger*** terjadi.

III. Pendeklarasian Variabel

Sintaks: **Identifier [CONSTANT] typedata [NOT NULL] [:= value];**

Keterangan:

- **Identifier**, merupakan nama untuk objek-objek PL/SQL seperti variabel, cursor , dsb
- **CONSTANT,** merupakan suatu opsional yang digunakan untuk mendeklarasikan konstanta
- **typedata** merupakan tipe dari identifier tersebut, dan value merupakan nilai awal dari identifier tersebut
- **NOT NULL,** opsional untuk mendeklarasikan suatu identifier agar tidak boleh kosong

- **:= value**, bisa merupakan literal, variabel yang lain atau sebuah ekspresi yang terdiri dari operator dan fungsi. Jika nilai inisial tidak diberikan, maka suatu variabel akan diberikan nilai **Null** untuk nilai inisialisasinya.

IV. Type data pada PL/SQL

Tipe Data	Keterangan
NUMBER(X,Y)	x adalah jumlah digit yang diinginkan (presisi), dan y menempatkan y desimal di belakan koma (skala). Jika y tidak ada maka dibulatkan ke atas. **Contoh**: v_kuantitas NUMBER(2,1);
Tipe Data	**Keterangan**
VARCHAR2(x)	Dimana x adalah panjang karakter yang kita inginkan. Panjang defaultnya 1, dan maksimal 32767. **Contoh**: v_nama_depan VARCHAR2(15);
CHAR(x)	Untuk string dengan panjang yang sudah pasti. Dengan panjang maksimal 255 untuk PL/SQL versi 1 dan 32767 untuk PL/SQL versi 2. **Contoh:** v_nim CHAR (9) := '113040000';
DATE	untuk data tanggal. Jangkanya dari 4712 sebelum Masehi sampai 4712 sesudah Masehi. **Contoh:** Today DATE := SYSDATE;
BINARY_INTEGER	Tipe variable ini digunakan untuk menyimpan nilai mulai dari -2.147.483.647 s/d +2.147.483.647
NATURAL	Tipe variable ini merupakan bagian dari BINARY_INTEGER yang dapat menyimpan mulai 0 s/d 2.147.483.647
POSITIVE	Tipe variable ini merupakan bagian dari BINARY_INTEGER yang dapat menyimpan mulai 0 s/d 2.147.483.647
%TYPE	Tipe variabel ini adalah sebuah atribut variabel untuk mendeklarasikan variabel sesuai dengan type field di table. **Contoh:** V_MAHASISWA MAHASISWA.NIM %TYPE
%ROWTYPE	Adalah atribut variabe yang digunakan untuk mendeklarasikan tipe data yang sesuai degan semua baris pada satu kolom di tabel. **Contoh:** R_MAHASISWA MAHASISWA %ROWTYPE

Contoh 4

```
DECLARE
v_jml_beri   NUMBER   NOT   NULL := 12;
v_nama_depan   VARCHAR2(15) := 'Tono';
v_nim   CHAR(9) := '113010000';
Today   DATE := SYSDATE;
phi   CONSTANT   Number (3,5) := 3.14286
BEGIN
              Dbms_output.put_line(to_char(v_jml_beri));
              Dbms_output.put_line(to_char(v_nama_depan));
              Dbms_output.put_line(v_nim);
              Dbms_output.put_line(to_char(today));
              Dbms_output.put_line(to_char(phi));
END;/
```

Menggunakan %TYPE dan %ROWTYPE

%TYPE merupakan atribut variabel yang berfungsi untuk memberikan tipe data yang sama dengan tipe data suatu kolom dari suatu tabel.

Sintaks: **[schema.]nama_table.nama_column%TYPE;**

Contoh 5

```
DECLARE
var_nim   Mahasiswa.nim%type
var_nama   Mahasiswa.nama%type;
var_alamat   Mahasiswa.alamat%type;
var_tgl   Mahasiswa.ttl%type;
BEGIN
select   nim,   nama,   alamat,   ttl
into   var_nim,   var_nama,   var_alamat,   var_tgl
from   Mahasiswa   where   nim='&nim';
dbms_output.put_line('=========Info Mahasiswa=========');
dbms_output.put_line('NIM : '||var_nama);
dbms_output.put_line('Nama : '||var_nama);
dbms_output.put_line('Alamat : '||var_alamat);
dbms_output.put_line('Tgl lahir : '||var_tgl);
EXCEPTION
when   no_data_found   then
dbms_output.put_line('Data Mahasiswa tidak ditemukan');
END;/
```

%ROWTYPE dapat diartikan melakukan duplikasi pada salah satu tabel melalui suatu variabel, dengan nilai/tipe yang serupa sesuai dengan tipe-tipe pada table.

Cara ini digunakan apabila :

- Kita tidak ingin bersusah payah menspesifikkan semua atribut
- Kita tidak mengetahui secara pasti semua atributnya
- Kita membutuhkan operand, yakni sebuah objek bertipe TABEL.

Sintaks: **[schema.]nama_tabel%ROWTYPE;**

Contoh 6

```
DECLARE
     rec_pemilih   pemilih%rowtype;
BEGIN
     select   no_pemilih,  nama,  alamat
     into   rec_pemilih.no_pemilih,   rec_pemilih.nama,   rec_pemilih.alamat
     from   pemilih
     where   no_pemilih='&no_pemilih';
     dbms_output.put_line('Nomor Pemilih : '||rec_pemilih.no_pemilih);
     dbms_output.put_line('Nama Pemilih : '||rec_pemilih.nama);
     dbms_output.put_line('Alamat Pemilih : '||rec_pemilih.alamat);
END;/
```

V. Statement SELECT dalam PL/SQL

Statement SELECT dalam PL/SQL harus mengembalikan sebuah baris, jika tidak ada baris atau ada banyak baris yang dihasilkan maka kesalahan akan muncul untuk itu diperlukan penanganan kesalahan pada bagian Exception pada struktur PL/SQL. Perilaku operasi **SELECT** pada **PL/SQL** berbeda dengan pada SQL. Operasi SELECT dalam PL/SQL harus diikuti klausa INTO.

Sintaks:

```
SELECT item,item,.....
INTO variable,variable.....
FROM table,table,.....
[WHERE kondisi][GROUP BY item,item][HAVING kondisi];
```

Klausa INTO harus terletak diantara SELECT dan FROM. Klausa ini digunakan untuk menspesifikasikan nama variabel yang digunakan untuk menempatkan nilai kolom yang dihasilkan dari suatu tabel. Satu variabel untuk satu kolom, dan urutannya disesuaikan.

Untuk mencoba contoh dibawah ini buatlah tabel dengan spesifikasi sebagai berikut :

TABEL MAHASISWA

NIM (not null)	NAMA (not null)	ALAMAT (not null)
MHS-001	Budi	Jl. Buah Batu 5
MHS-002	Slamet	Jl. Soekarno 9
MHS-003	Tarjo	Jl. Sukabirus 12

Contoh 7

```
DECLARE
     V_id   mahasiswa.nim%TYPE;
     V_nama    mahasiswa.nama%TYPE;
BEGIN
     select    nim,   nama   into    V_id,   V_nama   from   mahasiswa
     where   id_mahasiswa   LIKE   '%002';
     dbms_output.put_line( V_id||'  '||V_nama);
EXCEPTION
     when   no_data_found   then
     dbms_output.put_line('data tidak ditemukan');
END;
/
```

- Selanjutnya cobalah contoh dibawah ini dan analisa hasilnya.

Contoh 8

```
DECLARE
      Info_mahasiswa mahasiswa%ROWTYPE;
BEGIN
      SELECT nim, nama,  alamat INTO info_mahasiswa
      FROM mahasiswa  where  nim  LIKE '%002';
      dbms_output.put_line(info_mahasiswa.nim);
      dbms_output.put_line(info_mahasiswa.nama||'beralamat
di'||info_mahasiswa.alamat);
END;
/
```

VI. PL/SQL Record

PL/SQL record menjadikan variabel-variabel yang kita deklarasikan menjadi sebuah unit atau tipe baru sesuai dengan kebutuhan kita. Untuk penggunaannya **Memerlukan adanya deklarasi variabel** terhadap tabel terlebih dahulu sebelum data-datanya dapat digunakan.

Adapun manfaat dari penggunaan record[7] yaitu :

1. Abstraksi data
2. Dapat melakukan operasi ke semua variable dalam record secara bersamaan

3. Kode program menjadi lebih efisien

Sintaks:

```
TYPE   type_name   IS   RECORD (
     Field-1   type-1   [NOT NULL]    [:= expr-1]
     Field-2   type-2   [NOT NULL]    [:= expr-2]
     .
     .
     Field-n   type-n   [NOT NULL]    [:= expr-n]
 );
```

Keterangan :

- ***type_name*** merupakan nama dari tipe baru
- ***Field-1*** sampai ***Field-n*** merupakan nama dari field-field yang ada pada record
- ***type-1*** sampai ***type-n*** merupakan type data dari field tersebut.

Contoh 9

```
DECLARE
        TYPE   record_mahasiswa   IS   RECORD (
        id   mahasiswa.nim%TYPE,
        nama   mahasiswa.nama_ %TYPE,
        alamat   mahasiswa.alamat%TYPE);
        info_mahasiswa   record_mahasiswa ;
```

```
BEGIN
      SELECT   nim,   nama,    alamat   INTO   info_mahasiswa
      FROM   mahasiswa    where    nim    LIKE   '%002';
      dbms_output.put_line(info_mahasiswa.id);
      dbms_output.put_line(info_mahasiswa.nama||'beralamat
di'||info_mahasiswa.alamat);
END;
/
```

Selain dengan cara diatas, Record pada PL/SQL juga dapat dilakukan dengan **%ROWTYPE**. Jika variabel yang digunakan pada record tersebut hanya berada pada satu tabel saja, maka cara **%ROWTYPE** akan lebih mudah. Perhatikan perbedaan penggunaan keduanya pada contoh 7 dan 8.

VII. PL/SQL Table (Array)

Tipe data ini dibayangkan sebagai array pada bahasa C. PL/SQL Table bisa terdiri dari satu atau lebih elemen data dimana tipe datanya sama. Setiap elemen data diakses dengan menggunakan indeksnya (indeks bertipe binary_integer).

Sintaks:
```
TYPE   table_name   IS   TABLE   OF   type   INDEX   BY   BINARY_INTEGER;
```

Keterangan :

table_name : nama dari tipe baru yang akan didefinisikan

type : jenis tipe data yang digunakan

Jika ingin menggunakan PL/SQL Table, PL/SQL Table harus sudah didefinisikan terlebih dahulu. Cara pemanggilannya :

Sintaks: *nama_table(index)*

Contoh 10

```
DECLARE
TYPE   tabel_nomor   IS
TABLE   OF   number
INDEX   BY   BINARY_INTEGER;
A   tabel_nomor;
BEGIN
              A(1) := 10;
              A(2) := 20;
              A(3) := 30;
             Dbms_output.put_line('Nilai elemen ke-1'||' = '||to_char(A(1)));
             Dbms_output.put_line('Nilai elemen ke-2'||' = '||to_char(A(2)));
             Dbms_output.put_line('Nilai elemen ke-3'||' = '||to_char(A(3)));
END;
/
```

VIII. Percabangan (IF-THEN-ELSE)

Pernyataan percabangan pada PL/SQL berfungsi seperti statement IF pada bahasa pemrograman lain.

Ada tiga bentuk IF statement pada PL/SQL : [3]

- IF - THEN - END IF
- IF - THEN - ELSE - END IF
- IF - THEN - ELSIF - THEN - END IF

Sintaks:

```
IF   kodisi   THEN   aksi
     [ELSEIF   kondisi   THEN   aksi]
     [ELSE   aksi]
END IF;
```

Contoh 11

```
DECLARE
          equal   Exception;
          v_nilai1   number := '&inputan_1';
          v_nilai2   number := '&inputan_2';
BEGIN
          If   v_nilai1 = v_nilai2   then
                 raise   equal;
          elsif   v_nilai1  <  v_nilai2
                 dbms_output.put_line(v_nilai1||' lebih kecil dari '||v_nilai2);
          else
               dbms_output.put_line(v_nilai1||' lebih besar dari '||v_nilai2);
          end if;
EXCEPTION
                 when   equal   then
                 dbms_output.put_line('nilainya sama');
END; /
```

IX. Pengulangan (Looping)

Sesuai dengan namanya, perintah pengulangan digunakan untuk mengeksekusi suatu bagian dari program berulang kali sampai suatu kondisi tercapai, atau sebanyak yang ditentukan oleh pengguna[7]. Terdapat tiga macam statement pengulangan pada PL/SQL[7] yaitu :

1. LOOP-EXIT STATEMENT

Sintaks:

```
LOOP
    Statement;
    ...
EXIT [WHEN kondisi];
END LOOP;
```

Atau:

```
LOOP
    Statement;
    ...
    IF [kondisi] THEN EXIT;
END LOOP;
```

Contoh 12

```
DECLARE
        V_counter   number := 1;
BEGIN
         LOOP
              dbms_output.put_line(V_counter);
              V_counter := V_counter + 1;
              IF    V_counter   >   10   THEN
                     Exit;
              END   IF;
         END LOOP;
END;/
```

Contoh 13 [3]

```
LOOP
       balance_remaining := account_balance (account_id);
       IF   balance_remaining   <   1000  THEN
              EXIT;
       ELSE
              apply_balance(account_id, balance_remaining);
       END IF;
END LOOP; /
```

2. WHILE-LOOP STATEMENT

Sintaks:

```
WHILE   kondisi   LOOP
     Statement;
     ...
END   LOOP;
```

Contoh 14

```
DECLARE
TYPE   tabel   IS   TABLE   OF   varchar2(10)   INDEX   BY   BINARY_INTEGER;
Data   tabel;
vNoUrut   number := 1;
BEGIN
WHILE   vNoUrut   <=  10   LOOP
```

```
            Data(vNoUrut) := 'data ke-' || to_char(vNoUrut);
            dbms_output.put_line(Data(vNoUrut));
            vNoUrut := vNoUrut + 1;
END LOOP;
END; /
```

3. FOR-LOOP STATEMENT

Sintaks:

```
FOR   variabel-control   IN   [Reverse]   nilai_rendah .. nilai_tinggi
LOOP
          Statement;
          ...
END   LOOP;
```

Keterangan :

- **REVERSE digunakan** apabila kita ingin melakukan iterasi dari nilai tinggi ke rendah.

Sintaks : [3]

```
FOR   loop_counter   IN   REVERSE   1 .. 10
LOOP
       ... executable statements ...
END   LOOP;
```

Contoh 15

```
DECLARE
TYPE   tabel   IS   TABLE   OF   varchar2(10)   INDEX   BY   BINARY_INTEGER;
Data   tabel;
vNoUrut   number;
BEGIN
FOR   vNoUrut   IN   1..10
LOOP
           Data(vNoUrut) := 'data ke-' || to_char(vNoUrut);
           dbms_output.put_line(Data(vNoUrut));
END   LOOP;
END; /
```

X. LABEL dan GO TO

GOTO statement melakukan percabangan yang tidak memerlukan kondisi/ sesuai kebutuhan programmer ke executable statement lain dalam bagian execution dari sebuah blok PL/SQL[7].

Sintaks penulisan label: <<nama_label>>
Sintaks pemrosesan label: GOTO *nama_label*;

Contoh 16 [1]

```
BEGIN
        GOTO   second_output;
        DBMS_OUTPUT.PUT_LINE('This line will never execute.');
        <<second_output>>
        DBMS_OUPUT.PUT_LINE('We are here!);
END;/
```

XI. Penanganan Error (Exception)

Eksepsi merupakan sesuatu yang tidak diinginkan, harus diantisipasi dan ditangani dengan baik. Ada beberapa macam eksepsi yang akan dipelajari :

1. Exception Dasar

Beberapa jenis kesalahan yang umum terjadi antara lain :

Jenis Error	Keterangan
Syntax Error	Kesalahan dalam penulisan sintaks. Misal kurang tanda ';' dalam penulisan statement
Logic Error	Loop yang tanpa berhenti.
Compile Error	Penggunaan perintah yang salah dan baru diketahui pada saat di-compile
Run time Error	Error yang terjadi pada saat program dijalankan.

Adapun daftar nama eksepsi dalam Oracle (pre-defined exception) seperti di tabel berikut[6].

Nama Excpetion	Error Code	Keterangan
CURSOR_ALREADY_OPEN	ORA-6511	Membuka cursor yang sudah dibuka
INVALID_CURSOR	ORA-1001	Menggunakan cursor yang tidak valid
NO_DATA_FOUND	ORA-1403	Tidak ada baris yang dihasilkan oleh statement SELECT INTO
TOO_MANY_ROWS	ORA-1422	Baris yang dikembalikan oleh statement SELECT INTO lebih dari satu baris
ZERO_DEVIDE	ORA-1476	Pembagian dengan nol
ACCESS_INTO_NULL	ORA-6530	Memberikan nilai ke atribut dari objek yang belum diinisialisasi
COLLECTION_IS_NULL	ORA-6531	Dengan collection yang belum di inisialisasi, memberikan nilai ke elemen, atau meminta collection method yang tidak ada
DUP_VAL_ON_INDEX	ORA-1001	Melanggar constraint unique
LOGIN_DENIED	ORA-1722	Konversi dari string ke angka gagal
NOT_LOGGED_ON	ORA-1010	Melakukan operasi database pada saat tidak connect
PROGRAM_ERROR	ORA-6501	Internal Error
ROWTYPE_MISMATCH	ORA6504	Variabel host dan variabel cursor mempunyai tipe yang tidak compatible
STORAGE_ERROR	ORA-6500	PL/SQL kekurangan memory
SUBSCRIPT_BEYOND_COUNT	ORA-6533	Mereferensi suatu collection dengan menggunakan angka index yang lebih besar dari jumlah elemen di dalam collection
SUBSCRIPT_OUTSIDE_LIMIT	ORA-6532	Mereferensi suatu collection dengan menggunakan angka index yang lebih besar dari jumlah elemen di dalam collection
TIMEOUT_ON_RESOURCE	ORA-0051	Waktu yang tersedia habis
TRANSACTION_BACKED_OUT	ORA-0061	Transaksi di Rollback karena deadlock
VALUE_ERROR	ORA-6502	PL/SQL Operasi aritmatik, konversi, truncate, atau batas ukuran yang salah

2. Exception Lanjut

Exception secara umum terjadi dalam dua cara :

1. Sebuah kesalahan ORACLE muncul apabila diasosiasikan dengan sebuah exception. Exception ini muncul secara otomatis sebagai akibat dari kesalahan tersebut.
2. Sebuah Exception muncul secara explicit dengan menggunakan statement RAISE dalam sebuah blok.

- **User Defined Exception[5]**

Merupakan exception yang dibuat oleh user dan diaktifkan dengan perintah RAISE

Sintaks pendeklarasian exception: *exception-identifier* EXCEPTION;
Sintaks pemanggilan exception: RAISE *exception-identifier*;

exception-identifier merupakan variabel yang digunakan untuk mengaktifkan exception.

Contoh 17

```
DECLARE
      jumlah   number;
      Id_pemesan   pemesan.id_pemesan%TYPE := 'PN-011';
      Nama   pemesan.nama%TYPE := 'Tine';
      Alamat   pemesan.alamat%TYPE := 'Jl. Lengkong 7';
      penuh   exception;
BEGIN
      Select   count   into   jumlah from pemesan;
      If   jumlah   >   3
             then   raise   penuh;
      else
            insert   into   pemesan   values(Id_pemesan,  Nama,  Alamat);
      end if;
EXCEPTION
      When   penuh   then
              dbms_output.put_line('maaf jmlh pemesan max 3');
      when   others   then
             dbms_output.put_line('terjadi exception lain');
END;/
```

- **Membuat Error Message**[5]

Error message dapat dibuat dengan RAISE_APLICATION_ERROR. Ini merupakan prosedur yang disediakan oleh ORACLE untuk membuat sekaligus memanggil sebuah exception.

Sintaks : RAISE_APLICATION_ERROR*(error_number, 'pesan_error');*

- Nilai error_number harus berada dalam rentang nilai antara **-20999** dan **-20000.**

- pesan_error adalah pesan kesalahan yang didefinisikan oleh user untuk jenis exception yang dibuat. **Ukuran karakter *pesan_error*** tidak boleh lebih dari **2048 byte(karakter).** Prosedur ini dapat digunakan pada bagian Executable maupun Exception.

Contoh 18

```
BEGIN
          Delete   pemesan   where   id_pemesan='MHS-005';
          If   SQL%NOTFOUND   then
               RAISE_APPLICATION_ERROR(-20202,  'Mahasiswa tidak ditemukan');
        End   if;
END;
/
```

Contoh 19

```
DECLARE
        V_nama    pemesan.nama %TYPE;
BEGIN
        Select    nama    into    V_nama   from   pemesan
        Where   id_pemesan = 'MHS-005';
EXCEPTION
        When   no_data_found   then
            RAISE_APPLICATION_ERROR(-20100,  'Mahasiswa tidak ditemukan');
```

```
END;
/
```

Contoh 20

```
BEGIN
       Insert into mahasiswa(nim,nama) values('MHS-011','Carlos');
END;/
```

- **Exception Init Pragma**[7]

Exception ini memungkinkan developer membuat suatu yang sama atau memodifikasi pesan error dengan predefined exception yang sesuai dengan suatu nomor kesalahan.

Sintaks : PRAGMA EXCEPTION_INIT(exception-identifier, kode_error);

Pada contoh diatas terdapat kesalahan yaitu pada tabel petugas harus ada alamat, tetapi pada contoh di atas tidak ada. Sehingga muncul pesan error :

ORA-01400: cannot insert NULL into ("TIF2"."PEMESAN"."ALAMAT")

Developer dapat membuat exception yang sama dengan predefined exception seperti di atas, yaitu dengan cara:

Contoh 21

```
DECLARE
        Ada_kosong     exception;
        PRAGMA    EXCEPTION_INIT(Ada_kosong, -01400);
BEGIN
        insert    into    pemesan(id_pemesan,nama)    values('PN-013','Aming');
EXCEPTION
        When    Ada_kosong    then
        dbms_output.put_line('kolom NOT NULL tidak boleh kosong');
END;
```

XII. LATIHAN

1. Jelaskan apa yang dimaksud dengan "Declaration Section", "Executable Section" dan "Exception Section"!
2. Buatlah contoh perulangan sederhalan yang menggunakan statement LOOP dan Exit!
3. a

```
Hasil Looping  ke- 1
Hasil Looping  ke- 2
Hasil Looping  ke- 3
Hasil Looping  ke- 4
Hasil Looping  ke- 5

PL/SQL procedure successfully completed.

SQL> _
```

Buatlah PL/SQL yang dapat menampilkan hasil diatas!