Lab. Basis Data II

Oleh : Diana Effendi, ST.,MT

(Digunakan di lingkungan sendiri, sebagai buku ajar

mata kuliah Lab. Basis Data II)

Fakultas Teknik dan Ilmu Komputer

Program Studi Sistem Informasi

Universitas Komputer Indonesia

1. **Pertemuan 1**

Pengantar PL/SQL

1.1 Pengantar Oracle Developer

Oracle merupakan suatu kelompok produk yang mendukung desain, kreasi, dan menjalankan aplikasi melalui platform yang berbeda. Tool desain, development, dan akses data dari oracle desain adalah :

- Oracle designer
- Oracle developer

Oracle developer memungkinkan untuk membangun system dengan performance yang tinggi menguntungkan bagi GUI (Graphical User Interface), database, client-server, dan teknologi web. Tools dari oracle developer mempunyai interface dan karakteristik umum yang memudahkan user dalam penggunaan serta mendesain elemen modular, kemudian dapat digunakan kembali (reusable).

1. Karakteristik utama oracle developer

Karakteristik utama dari oracle developer 6 memberi kontribusi dan fleksibelitas untuk produk oracle.

a. Mendukung GUI yang komprehensif
b. Aplikasi yang dapat didistribusi
c. Tools yang produk dan komperhansif
d. Partisipasi aplikasi
e. Source control yang fleksibel
f. Scalabilty
g. Orientasi object

2. Kelebihan oracle developer Release 6 menyederhanakan beberapa tugas developer. Karakteristik yang termasuk di dalamnya :

- Wizard untuk memudahkan tugas yang dikerjakan berulang-ulang.
- Visual query builder sebagai eksekusi yang berdiri sendiri dan sebagai sebuah utilitas yang disimpan di dalam reports.
- Sebelum didefenisikan dan template ekstensi user.
- Gallery sebelum didefenisikan dan template ekstensi user

- Mendukung klien yang berjalan pada Microsoft Messaging Application Program interface (MAPI) compliant clients untuk memudahkan distribusi report
- Open Application Programming Interface (APIs), memungkinkan developers untuk memanipulasi sebuah file, penyediaan program interface yang sangat cepat.

3. Pengenalan komponen-komponen Oracle Developer

- Project Builder
- Form builder
- Report builder
- Graphic builder
- Query Builder
- Schema Builder
- Procedure Builder
- Translation Builder

Komponen – komponen Utama Builder Oracle developer merupakan produk untuk membuat form, report, dan Graapichs. Oracle Developer memiliki beberapa komponen utamanya. Komponen-komponen ini termasuk pilihan dalam suatu tool yang tidak dipakai pada aplikasi yang lain, tapi fungsional utamanya adalah sama. Komponen –komponen ini membantu menyediakan lingkungan pengembangan Oracle Developer yang fleksibel dan produktif development environment.

1. Navigasi object

Navigasi objek merupakan pencarian yang hirarki dan pengeditan interface yang memungkinkan untuk mengalokasikan dan memanipulasikan objek aplikasi dengan cepat dan mudah. Karakteristiknya antara lain :

- Sebuah hirarki persentasi ulang dengan identasi dan node yang dapat dikembangkan.
- Menemukan field dan ikon
- Icon yang terdapat pada toolbar yang vertical

- Sebuah ikon yang didepannya untuk setiap objek mengindikasikan tipe objek tersebut.

2. Pallete property

Semua objek di dalam module, termasuk module itu sendiri mempunyai properties yang dapat dilihat dan dimodifikasi dalam pallete property. Karakteristiknya antara lain :

- Mengcopy dan menggunakan kembali properties dari objek lain
- Menemukan field dan ikon, sama dengan navigasi objek.

3. Layout editor

Layout editor adalah fasilitas desain grafis untuk membuat dan menyusun item interface dan objek grafikal pada aplikasi. Ketika menggunakan pallete tool dan toolbar yang terdapat pada layout editor, dapat mendesain style, warna, size, dan susunan objek visual pada aplikasi yang ada. Layout dapat memasukkan objek grafik dan image bitmap ketika

dijalankan pada GUI.

4. Lingkungan PL/SQL development
Lingkungan pengembangan merupakan kumpulan fungsionalitas dari procedure builder yang ada pada builder aplikasi, yang menyediakan :
• Pengembangan trigger, procedurs, functions, dan paket data oracle developer sebaik database
• Pengembangan libraries untuk menempatkan unit program PL/SQL
• Mendebug level statement dari PL/SQL pada waktu dijalankan

5. Oracle Toolkit dan Oracle multimedia
Oracle toolkit adalah fungsi library yang menampilkan user interface event, seperti control scroll bar dan menu aktifasi. Bersama oracle multimedia menyediakan kumpulan images, suara, dan fasilitas media lainnya, toolkit mendasari pengembangan tool oracle developer, menjembatani aplikasi oracle developer dan lingkungan asli yang dikerjakan. Sebagai fasilitas yang dibutuhkan oleh

aplikasi seperti membuka jendela atau menampilkan menu, oracle developer menempatkan permintaan pada toolkit yang berkomunikasi dengan platform asli.

Toolkit mencoba untuk menempatkan tugas pada interface awal, seandainya interface tersebut dapat menangani kebutuhan tersebut, maka aplikasi tersebut menggunakan karakteristik natural dari lingkungan. Artinya aplikasi oracle developer melihat dan berkelakuan menyerupai aplikasi windows NT ketika dijalankan pada windows NT, dan seperti motif atau aplikasi macintosh jika dipindahkan pada platform tersebut.
Toolkit itu sendiri menyediakan funsionalitas dimana secara pasti fungsionalitas tersebut tidak dapat melalui interface awal, contohnya pada device mode karakter. Ini menyediakan interface yang dapat beradaptasi pada aplikasi oracle developer.

Tahapan – tahapan penggunaan Oracle developer
Memulai builders
Ada dua cara untuk membuka builders :

- Buka project builder dan pilih buider yang dibutuhkan dari launchpad project builder
- Buka builder individual dari group oracle developer 6.0

Yang dapat dilihat dalam builders
Ketika membuka builder, pertama akan terlihat banner produk dan kemudian wizardnya. Ketika membuka builder lainnya, akan terlihat wizard. Setiap builder wizard menawarkan beberapa pilihan diantaranya :
- Membuat modul baru menggunakan wizard
- Membuat modul baru secara manual
- Membuka modul yang tersedia

Koneksi database
Jika membuat sebuah aplikasi yang mengakses objek database, akan membutuhkan koneksi dengan account database dari builder. Koneksi ke database dilakukan jika ingin :
- Meng-compile kode yang mengandung SQL
- Meng-akses objek database dalam navigasi objek

- Membuat objek oracle developer yang berdasarkan objek database

Variabel oracle developer

Oracle developer menggunakan beberapa jenis variabel. Semua itu dapat dimodifikasi pada lingkungan sendiri, kemungkinan akan terdapat nilai yang salah pada aplikasi yang lain.

1.2 Pengenalan PL/SQL

PL / SQL (Bahasa Prosedur / Structured Query Language) adalah ekstensi prosedural Oracle Corporation untuk SQL dan database relasional Oracle . PL / SQL tersedia di Oracle Database (sejak versi 6 - stored pl / sql procedures / functions / packages / trigger sejak versi 7), database memori TimesTen (sejak versi 11.2.1), dan IBM DB2 (sejak versi 9.7) . Oracle Corporation biasanya memperluas fungsionalitas

PL / SQL dengan setiap rilis Oracle Database berturut-turut.

PL / SQL mencakup elemen bahasa prosedural seperti kondisi dan loop . Hal ini memungkinkan deklarasi konstanta dan variabel , prosedur dan fungsi, tipe dan variabel dari jenis dan pemicu tersebut. Ini bisa menangani pengecualian (runtime error). Array didukung dengan melibatkan penggunaan koleksi PL / SQL. Implementasi dari versi 8 dari Oracle Database dan seterusnya telah menyertakan fitur yang terkait dengan object-orientation . Seseorang dapat membuat unit PL / SQL seperti prosedur, fungsi, paket, jenis, dan pemicu, yang disimpan dalam database untuk digunakan kembali oleh aplikasi yang menggunakan antarmuka pemrograman Database Oracle.

Struktur PL/SQL

Struktur PL/SQL mirip dengan struktur bahasa pascal atau delphi yang menggunakan struktur blok,

sehingga akan mempermudah pengertian dalam pemrograman dengan PL/SQL. Struktur Blok berisi perintah SQL dengan kondisi yang berbeda. Perintah PL/SQL dapat menangani kesalahan saat dijalankan. Setiap pengetikan dengan menggunakan PL/SQL dalam SQL*Plus selalu diakhiri dengan tanda /(slash). Sintaks penggunaan PL/SQL adalah sebagai berikut :

Declare

Begin

Exception

End

Struktur diatas dapat dijelaskan sebagai berikut :

1. Bagian Judul (Header)

Bagian ini hanya digunakan jika PL/SQL diberikan nama,misalnya untuk prosedur atau fungsi. Bagian ini berisi nama blok,daftar parameter, dan pengembalian hasil (return) jika blok adalahfungsi.

2. Bagian Deklarasi (declaration)

Bagian ini untuk membuat deklarasi mengenai semua variable dan konstanta yang direferensikan dalam pernyataan PL/SQL. Bagian deklarasi ini dimulai dengan perintah DECLARE. Jika tidak ada variable atau konstanta yang ingin dideklarasikan bagian ini boleh dihilangkan, bersifat optional.

3. Bagian Eksekusi (Execution).

Bagian ini memuat pernyataan-pernyataan PL/SQL yang akan ditulis. Bagian eksekusi ini harus dimulai dengan perintah BEGIN.

4. Bagian Perkecualian (Exception)

Bagian ini memuat cara menangani kesalahan-kesalahan (error) pada waktu eksekusi program PL/SQL, bersifat optional. Jika program tidak memuat cara menangani kesalahan, bagian ini boleh dihilangkan.Setiap pernyataan PL/SQL harus diakhiri dengan tanda titik koma(;) dan semua program PL/SQL harus diakhiri dengan perintah END.

Bentuk Umum Struktur 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;
```

Jenis-Jenis Blok Struktur PL/SQL

1. Blok Anonim

```
[DECLARE]
   BEGIN
-statements-
[EXCEPTION]
```

END;

2. Procedure

PROCEDUR name
IS
BEGIN
– statements-
[EXCEPTION]
END;

3. Function

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

END;

1.3 Deklarasi PL/SQL

Variabel adalah salah satu lokasi di memori yang digunakan untuk menyimpan program sehingga dapat dimanipulasi nilainya.

Setiap variabel di PL/SQL harus mempunyai tipe data. Kita harus memberikan tipe data yang sesuai pada variabel yang akan kita buat.

Variables digunakan untuk:

- Menyimpan data sementara
- Memanipulasi nilai yang disimpan
- Dapat digunakan kemballi
- Mudah dalam pemeliharaan

Di bawah ini adalah sintak dasar deklarasi variabel di PL/SQL:

nama_variabel [CONSTANT] tipe_data [NOT NULL] [:= | DEFAULT nilai_awal]

Penjelasan sintak:

- nama_variabel adalah nama variabel yang akan digunakan
- [KONSTANTA] adalah kata kunci (keyword) apabila kita mendeklarasikan variabel sebagai konstanta.
- tipe_data adalah tipe data yang akan digunakan pada variabel.
- := adalah assigment operator
- DEFAULT nilai_awal adalah nilai awal (initial value) yang akan diberikan pada variabel

Inisialisasi Variabel

Sebelum variabel digunakan, Anda harus menginisialisasi terlebih dahulu. Inisialisasi adalah proses pemberian nilai pada suatu variabel.

Anda dapat menggunakan 2 cara dalam memberikan nilai variabel pada PL/SQL, yaitu:

- Menggunakan assigment operator (:=)
- Menggunakan keyword DEFAULT

Penanganan Variable dalam PL/SQL

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

Jenis-Jenis Type Data

1. Type data primitive (Sederhana)

Tipe data primitive adalah Tipe data yang mampu menyimpan satu nilai tiap satu variabel. Contoh tipe data primitive adalah tipe numerik (integer dan real), tipe data karakter/char, tipe data boolean.

2. Type data Composite

   Tipe Data Komposit merupakan tipe data yang dapat menampung banyak nilai, antara lain Array, Record atau struct, Image, Date Time, Object.

3. Type Data Lain

   - Type Data terstruktur (Diantaranya terdapat String & Data Set)
   - Type Data Poiter (Typed & Generic)

2. Pertemuan 2

Anonymous Blok PL/SQL dan Tipe Data

2.1 Pengenaalan Blok PL/SQL

Blok PL/SQL Anonim adalah blok PL/SQL tanpa nama dan tidak menggunakan parameter. Blok anonym tidak disimpan dalam database sehingga tidak bisa direferensi oleh blok PL/SQL lain.

Pada saat blok ini dijalankan pertama kali, penempatannya ke shared pool harus didahului dengan proses parsing dan kompilasi.Proses reparsing dan rekompilasi tidak dilakukan jika blok telah berada di shared pool. Apabila blok PL/SQL itu telah dikeluarkan dari shared pool, eksekusi ulang terhadap blok PL/SQL tadi harus melalui tahapan parsing dan kompilasi. Ini disebabkan blok PL/SQL tidak disimpan di databse meskipun blok itu bisa disimpan pada file system operasi.

Dalam pemograman modular, blok-blok PL/SQL anonym itu dapat diubah menjadi fungsi, prosedur, atau paket sehingga blok PL/SQL bisa direferensi melalui namanya dan menghindar duplikasi pembuatan kode program.

2.1.1 Parameter

Subprogram dipanggil dengan melewatkan nilai, variable, atau ekspresi susuai parameter yang ada. Variabel yang bersada pada statement pemanggikl subprogram atau yang disebut parameter actual akan dikirimkan ke parameter formal, yaitu variable yang dideklarasikan pada fungsi atau prosedur yang dipanggil. Parameter formal dan actual itu harus memiliki tipe data yang sama atau kompatibel.

Pemanggilan subprogram untuk melewatkan parameter actual dapat dilakukan dengan cara notasi posisi parameter, notasi nama parameter, dannotasi kombinasi. Penerapan notasi posisi melakukan pemanggilan subprogram berdasarkan posisi atau urutan parameter formal. Notasi nama parameter melakukan pengiriman parameter actual dengan cara menyertakan nama parameter formal diikuti tanda panah '=>' dan nilai atau variabelnya. Pada notasi kombinasi pemanggilan subprogram dapat menyertakan notasi posisi dan notasi nama

parameter dengan ketentuan bahwa notasi posisi dan mandahului notasi nama pada parameter actual.

2.1.2 Mode Parameter

Parameter formal menerima nilai atau variable dari parameter actual. Subprogram dapat menggunakan parameter formal sesuai mode parameter yang ditetapkan yaitu IN, OUT, dan IN OUT. Secara default setiap parameter formal menggunakan mode In yang berarti bahwa parameter itu hanya dapat dibaca atau direfernsi dalam body subprogram. Mode OUT berarti parameter formal hanya dapat ditulis atau diberikan nilai, sedangkan mode IN OUT memungkinkan parameter formal untuk dibaca dan ditulis. Jadi untuk dapat mengmbalikan berbagai nilai maka subprogram harus menggunakan mode OUT atau mode IN OUT, sedangkan mode IN bersifat seperti konstanta yang tidak dapat dimodifikasikan dalam body prosedur yang dipanggil.

2.1.3 Compiler Hint Nocopy

Proses pengiriman parameter actual dilakukan dengan dua cara yaitu by reference dan by value. Pangiriman parameter actual secara by reference dilakukan dengan melewatkan pointer parameter actual ke parameter formal sehingga kedua parameter mereferensi lokasi memori yang sama. Mekanisme itu berlangsung pada parameter formal dengan mode IN.

Secara default, pengiriman parameter actual pada subprogram yang menggunakan mode OUT dan mode IN OUT dilakukan secara by value yaitu nilai parameter actual di-copy-kan ke parameter formal. Setelah eksekusi subprogram diakhiri, nilai parameter formal akan di-copy-kan ke parameter actual sehingga nilai parameter actual berubah. Untuk mengubah pola pengiriman by valuemenjadi by reference gunakan hint compiler NOCOPY sehingga copy nilai datadapat dihindari. Penggunaan copy nilai data antara kedua parameter itu dapat meurunkan kecepatan

eksekusi dan meningkatkan penggunaan memori terutama pada parameter yang menangani data berukuran besar seperti record, collection dan tipe objek.

Nocopy menyebabkan perubahan nilai pada parameter formal segera mempengaruhi nilai parameter actual. Ini berbeda dengan kondisi defaultnya dimana jika subprogram berakhir secara tidak normal maka parameter formal tidak akan dicopy pada parameter actual. Sebagai suatu hint compiler, nocopymelakukan pengiriman parameter by reference, sebaliknya parameter actual dilakukan secara by value.

Hint ini memungkinkan terjadinya parameter aliasing yaitu suatu kondisi di mana beberapa nama variable atau parameter mengacu pada lokasi memori yang sama. Ini bisa terjadi jika variable global digunakan sebagai parameter actual dan variable global itu direferensi dalam subprogram. Situasi itu bisa menimbulkan masalah kesalahan logika. Oleh karena itu dalam

pemrograman perlu diperhatikan scope atau duraasi pengguanaan variable.

2.2 Penggunaan Variables

Inisialisasi Variabel

Sebelum variabel digunakan, Anda harus menginisialisasi terlebih dahulu. Inisialisasi adalah proses pemberian nilai pada suatu variabel.

Anda dapat menggunakan 2 cara dalam memberikan nilai variabel pada PL/SQL, yaitu:

- Menggunakan assigment operator (:=)
- Menggunakan keyword DEFAULT

2.3 Jenis Type Variables

-Variabel PL/SQL s:

- Scalar
- Composite

• Reference

• LOB (large objects)

- Variabel Non-PL/SQL :

- Bind and host
- variables

Tipe data Scalar Dasar

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

Base Scalar Datatypes

– VARCHAR2 (maximum_length)

– NUMBER [(precision, scale)]

– DATE

– CHAR [(maximum_length)]

– LONG

– LONG RAW

– BOOLEAN

– BINARY_INTEGER

– PLS_INTEGER

2.4 Penulisan Operator PL/SQL

DEKLARASI VARIABLE PL/SQL

Identifiers (Pengenal)

Identifiers digunakan untuk :

• Penamaan variable

• Aturan Penamaan :

- Diawali dengan huruf

- Dapat berisi huruf, angka, $, _, atau #

- Maksimal 30 karakter

Penanganan Variable dalam PL/SQL

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

DBMS_OUTPUT.PUT_LINE

- Prosedur Oracle-supplied packaged
- Sebagai alternatif menampilkan data dari blok PL/SQL
- Harus di aktifkan dalam SQL*Plus dengan perintah SET SERVEROUTPUT ON

3. Pertemuan 3

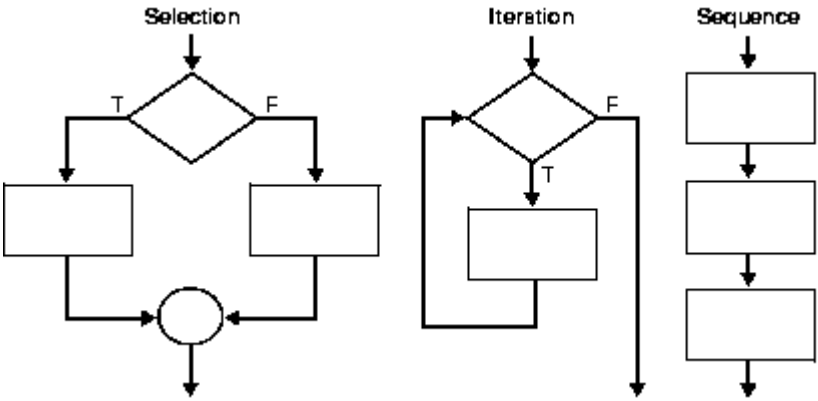
Penulisan Struktur Kontrol Seleksi dan Pengulangan

3.1 Pengenalan dan Penggunaan Struktur Kontrol

Ikhtisar Struktur Kontrol PL / SQL

Menurut teorema struktur , setiap program komputer dapat ditulis menggunakan struktur kontrol dasar yang ditunjukkan pada Gambar 4-1 . Mereka dapat dikombinasikan dengan cara apapun yang diperlukan untuk menangani masalah tertentu.

Gambar 4-1 Struktur Kontrol



Struktur seleksi menguji suatu kondisi, kemudian mengeksekusi satu urutan pernyataan, bukan yang lain, tergantung apakah kondisinya benar atau salah. Suatu *kondisi* adalah variabel atau ekspresi yang mengembalikan nilai Boolean (TRUE or FALSE). Struktur iterasi mengeksekusi serangkaian pernyataan berulang-ulang selama kondisi benar. Struktur urutan hanya mengeksekusi urutan pernyataan sesuai urutan kemunculannya.

3.2 Struktur Kontrol IF .. THEN… ELSE

Pernyataan IF-THEN

Bentuk paling sederhana dari pernyataan IF mengasosiasikan suatu kondisi dengan urutan pernyataan yang dilampirkan oleh kata kunci THEN dan END IF (not ENDIF), sebagai berikut:

```
JIKA kondisi KEMUDIAN
sequence_of_statements
BERAKHIR JIKA;
```

Urutan pernyataan dijalankan hanya jika kondisinya benar. Jika kondisinya salah atau nol, pernyataan IF tidak melakukan apa-apa. Dalam kedua kasus, kontrol lolos ke pernyataan berikutnya. Contoh berikut:

```
JIKA penjualan> kuota KEMUDIAN
compute_bonus (empid);
UPDATE payroll SET pay = bayar + bonus WHERE empno = emp_id;
BERAKHIR JIKA;
```

Anda mungkin ingin menempatkan pernyataan IF singkat pada satu baris, seperti pada

JIKA x> y KEMUDIAN tinggi: = x; BERAKHIR JIKA;

Pernyataan IF-THEN-ELSE

Bentuk kedua dari statement IF menambahkan kata kunci ELSE diikuti dengan urutan urutan pernyataan, sebagai berikut:

JIKA kondisi KEMUDIAN
sequence_of_statements1
LAIN
sequence_of_statements2
BERAKHIR JIKA;

Urutan pernyataan dalam klausa ELSE dijalankan hanya jika kondisinya salah atau null. Jadi, klausa ELSE memastikan bahwa urutan pernyataan dieksekusi. Pada contoh berikut, pernyataan UPDATE pertama dijalankan saat kondisinya benar, namun pernyataan UPDATE kedua akan dijalankan bila kondisinya salah atau null:

JIKA trans_type = 'CR' THEN

Akun UPDATE SET balance = balance + credit WHERE ...

LAIN

Akun UPDATE SET keseimbangan = keseimbangan - debit WHERE ...

BERAKHIR JIKA;

Klausa THEN dan ELSE dapat mencakup pernyataan IF . Artinya, pernyataan IF dapat disarangkan, seperti contoh berikut ini:

JIKA trans_type = 'CR' THEN

Akun UPDATE SET balance = balance + credit WHERE ...

LAIN

JIKA new_balance> = minimum_balance THEN

Akun UPDATE SET keseimbangan = keseimbangan - debit WHERE ...

LAIN

RAISE insufficient_funds;

BERAKHIR JIKA;

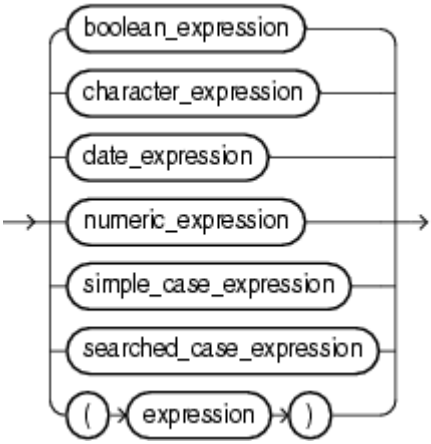
BERAKHIR JIKA;

3.3 Penggunaan Ekspresi

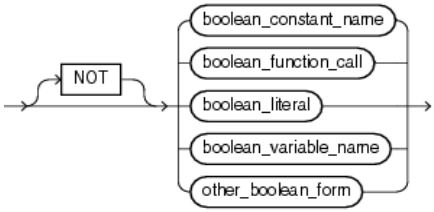
Ekspresi adalah kombinasi operand yang sewenang-wenang, variabel, konstanta, literal, operator, pemanggilan fungsi, dan placeholder) dan operator. Ungkapan paling sederhana adalah satu variabel.Kompilator PL / SQL menentukan jenis data dari ekspresi dari jenis operan dan operator yang terdiri dari ekspresi. Setiap kali ungkapan dievaluasi, satu nilai dari hasil jenis itu.

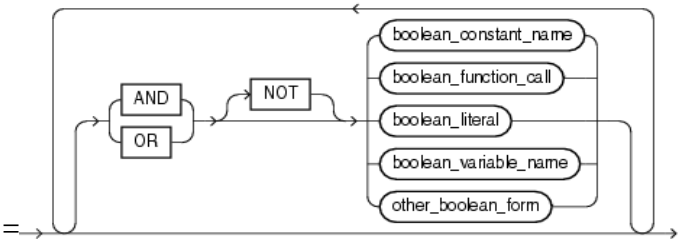
Sintaksis

ekspresi :: =

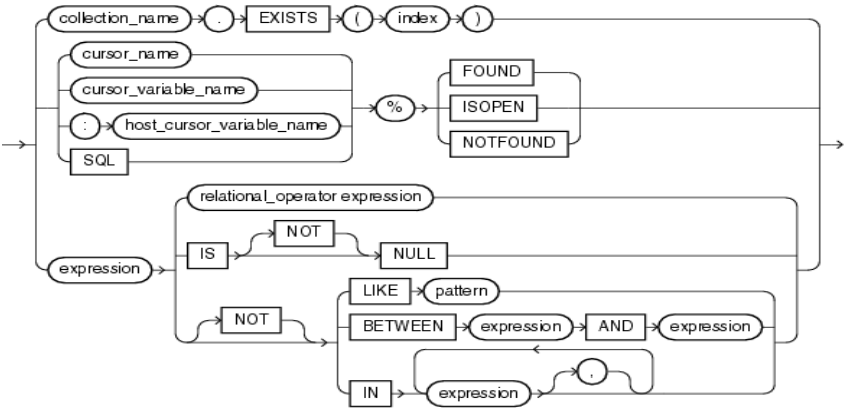


boolean_expression ::

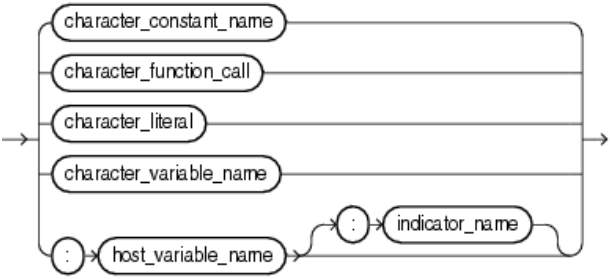


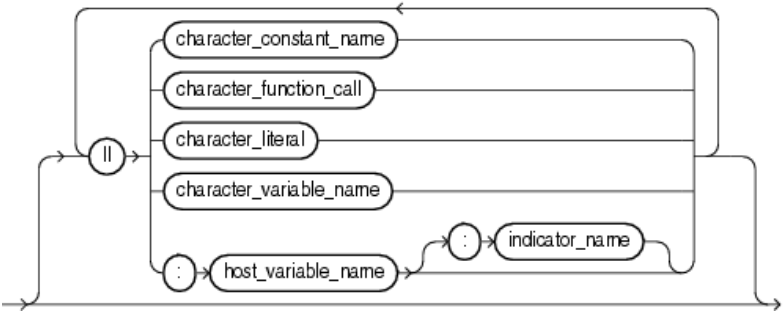


other_boolean_form :: =



character_expression :: =





4. Pertemuan 4

Penulisan Struktur Kontrol Seleksi dan Pengulangan

4.1 Perbedaan Berbagai Jenis LOOP

Untuk perintah LOOP, akan dilakukan pengulangan terus-menerus. Bentuk umum dari pernyataan LOOP sebagai berikut:

```
LOOP
    //Baris perintah
END LOOP;
```

Karena tidak mempunyai kondisi untuk keluar dari iterasi, maka perlu digunakan perintah EXIT. Perintah EXIT dapat digunakan dengan cara seperti berikut:

```
EXIT WHEN kondisi;
```

Contoh:

```
DECLARE
  x number;
BEGIN
  x := 0;
  LOOP
        x := x + 1;
```

```
        EXIT WHEN x > 5; -- exit loop immediately
    END LOOP;
    dbms_output.put_line('Hasil looping : '||x);
END;
```

Pernyataan While - Loop

Perintah WHILE-LOOP akan terus melakukan iterasi (memproses baris perintah secara berulang) selama KONDISI bernilai TRUE. Bentuk umum dari pernyataan LOOP sebagai berikut:

```
WHILE kondisi LOOP
    //Baris perintah
END LOOP;
```

Contoh:

```
DECLARE
  x number;
BEGIN
  x := 0;
  WHILE x <= 5 LOOP
    x := x + 1;
```

```
    dbms_output.put_line('Hasil looping ke- '||x);
  END LOOP;
END;
```

Selain dapat digunakan pada perintah LOOP, perintah EXIT ini juga dapat digunakan pada WHILE-LOOP untuk menambahkan kondisi tertentu. Namun perintah EXIT ini hanya bisa digunakan dalam loop saja.

Pernyataan For - Loop

Struktur pengulangan *For* digunakan untuk menghasilkan pengulangan sejumlah kali tanpa penggunaan kondisi apapun. Stuktur ini menyebabkan aksi diulangi sejumlah beberapa kali (tertentu). Bentuk umum struktur *for* ada dua macam yaitu : menaik (*ascending*) atau menurun (*descending*). Sintaksnya sebagai berikut :

```
FOR counter IN [REVERSE] i_terendah .. i_teratas LOOP
    Baris perintah
```

```
END LOOP;
```

Perintah FOR-LOOP melakukan iterasi selama nilai COUNTER berada dalam range nilai *i_terendah* dan *i_teratas*. Pada FOR-LOOP, *counter* tidak perlu dideklarasikan. Penggunaan kata kunci RESERVE akan menyebabkan nilai counter dimulai dari i_teratas ke i_terendah. Dua titik antara i_terendah dan i_teratas merupakan operator rentang nilai. i_terendah maupun i_terkecil bisa berupa nilai integer ataupun variabel yang bernilai integer yang sudah dideklarasikan sebelumnya. i_upper harus lebih besar dari i_lower dan jika tidak maka iterasi tidak akan dilakukan.

Contoh:

```
BEGIN
FOR vno IN 1..10 LOOP
insert into coba(no) values vno;
dbms_output.put_line('Hasil looping ke- '||x);
```

```
END LOOP;
END;
```

```
BEGIN
FOR vno IN REVERSE 1..10 LOOP
insert into coba(no) values vno;
dbms_output.put_line('Hasil looping ke- '||x);
END LOOP;
END;
```

Ruang Lingkup Pernyataan FOR – LOOP

Untuk menuju ke variabel global, harus ditambahkan label dan notasi *dot*.

Contoh :

```
<<main>>
DECLARE
  ctr INTEGER;
  ...
BEGIN
  ...
  FOR ctr IN 1..25 LOOP
```

```
        ...
        IF main.ctr > 10 THEN -- refers to global variable
          ...
        END IF;
  END LOOP;
END main;
```

Hal ini juga berlaku untuk *nested loop.*

Contoh :

```
<<main>>
DECLARE
  ctr INTEGER;
  ...
BEGIN
  <<outer>>
  FOR step IN 1..25 LOOP
        FOR step IN 1..10 LOOP
          ...
          IF outer.step > 15 THEN
        ...
```

```
    END IF;
        END LOOP;
  END LOOP outer;
END main;
```

Selain dapat digunakan pada perintah LOOP, perintah EXIT ini juga dapat digunakan pada FOR-LOOP untuk menambahkan kondisi tertentu. Namun perintah EXIT ini hanya bisa digunakan dalam loop saja.

Contoh:

```
BEGIN
  FOR j IN 1..10 LOOP
        FETCH c1 INTO mhs_rec;
        EXIT WHEN c1%NOTFOUND;
        ...
  END LOOP;
END;
```

```
BEGIN
```

```
<<outer>>
FOR i IN 1..5 LOOP
        ...
        FOR j IN 1..10 LOOP
          FETCH c1 INTO mhs_rec;
          EXIT outer WHEN c1%NOTFOUND; -- exit
both FOR loops
 ...
        END LOOP;
 END LOOP outer;
 -- control passes here
END;
```

4.2 Penggunaan nested loops (LOOP Bersarang)

Nested Loop atau perulangan bersarang adalah perulangan di dalam perulangan.

Sintak Dasar Nested Loop

```
LOOP
  statement1;
```

```
  LOOP
    statement2;
  END LOOP;
 END LOOP;
```

Sintak Dasar Nested For Loop

```
FOR counter1 IN initial_value1 .. final_value1
LOOP
   statement1;
   FOR counter2 IN initial_value2 .. final_value2
LOOP
      statements2;
   END LOOP;
END LOOP;
```

Sintak Dasar Nested While Loop

```
WHILE condition1 LOOP
statements1;
WHILE condition2 LOOP
 statements2;
END LOOP;
END LOOP;
```

Contoh Program

```
DECLARE
 i number(3);
 j number(3);
BEGIN
 i := 2;
 LOOP
   j:= 2;
```

```
    LOOP
      EXIT WHEN ((mod(i, j) = 0) or (j = i));
      j := j +1;
    END LOOP;
    IF (j = i ) THEN
      DBMS_OUTPUT.PUT_LINE(i || ' adalah
bilangan prima');
    END IF;
    i := i + 1;
    EXIT WHEN i = 50;
  END LOOP;
END;
/
```

5. Pertemuan 5

Pengenalan *Cursors* dan *Handling Exception*

5.1 Penulisan Cursor

Oracle menggunakan area-area kerja (work area) untuk mengeksekusi perintah-perintah PL/SQL dan menyimpan informasi yang sedang diproses. Konstruksi PL/SQL yang disebut cursor mengijinkan kita memberi nama sebuah area kerja dan mengakses informasi yang ada di dalamnya. Terdapat dua macam cursor: implisit dan explisit. PL/SQL secara implisit mendeklarasikan cursor untuk seluruh perintah-perintah manipulasi data SQL, termasuk query-query yang hanya menghasilkan satu baris data. Untuk query-query yang menghasilkan lebih dari satu baris data, kita dapat secara eksplisit mendeklarasikan cursor untuk memproses baris-baris data secara individual.

```
DECLARE
CURSOR c1 IS
SELECT empno, ename, job FROM emp
WHERE deptno = 20;
```

Kumpulan baris-baris data yang dihasilkan oleh query yang menghasilkan banyak baris data disebut dengan result set. Besarnya adalah jumlah baris data sesuai dengan kriteria pencarian. Seperti yang ditunjukkan oleh Gambar 1-2, explicit cursor "menunjuk" kepada current row (baris terkini) di dalam result set.

5.2 Pembagian Cursor

5.2.1 Cursor FOR Loops

Pada banyak situasi yang membutuhkan explicit cursor, secara sederhana kita dapat melakukan*coding* dengan menggunakan cursor FOR loop dibanding menggunakan perintah OPEN, FETCH, dan CLOSE. Cursor FOR loop secara implisit mendeklarasikan *loop index*-nya sebagai record yang merepresentasikan baris data yang didapat database. Kemudian, ia membuka cursor, secara berulangkali mengambil nilai-nilai baris data dari result set ke field-field pada record, kemudian menutup cursor ketika seluruh

baris data telah selesai diproses. Pada contoh berikut ini, cursor FOR loop secara implisit mendeklarasikan emp_rec sebagai sebuah record:

```
DECLARE
CURSOR c1 IS
SELECT ename, sal, hiredate, deptno FROM emp;
...
BEGIN
FOR emp_rec IN c1 LOOP
...
salary_total := salary_total + emp_rec.sal;
END LOOP;
```

Untuk mereferensi field-field tunggal di dalam record, kita menggunakan*dot notation*, dimana dot (.) berlaku sebagai penyeleksi komponen.

5.2.2. Cursor Variable

Seperti halnya cursor, cursor variable menunjuk kepada baris terkini (current row) di dalam result set dari multi-row query. Tetapi, tidak seperti cursor, cursor variable dapat
dibuka untuk type-compatible query. Ia tidak terikat dengan kepada query tertentu. Cursor variable benar-benar merupakan variable PL/SQL, diaman kita dapat memberikan nilai baru dan melewatkannya ke subprogram-subprogram di dalam database Oracle. Hal ini memberikan kita fleksibilitas lebih dan cara yang tepat untuk memusatkan proses menampilkan data.

Biasanya, kita membuka cursor variable dengan melewatkannya ke stored procedure yang mendeklarasikan cursor variable sebagai satu dari parameter-parameter formalnya. Procedure berikut ini membuka cursor variable yang bernama generic_cv untuk query yang dipilih:

```
PROCEDURE open_cv (generic_cv IN OUT
GenericCurTyp,choice NUMBER) IS BEGIN
IF choice = 1 THEN
OPEN generic_cv FOR SELECT * FROM emp;
ELSIF choice = 2 THEN
OPEN generic_cv FOR SELECT * FROM dept;
ELSIF choice = 3 THEN
OPEN generic_cv FOR SELECT * FROM
salgrade;
END IF;
...
END;
```

6. Pertemuan 6

Pengenalan *Cursors* dan *Handling Exception*

6.1 Kolaburasi Cursor dengan Struktur Kontrol

Kursor implisit: Oracle secara otomatis (implisit) mengendalikan atau memproses informasi dari pernyataan SQL yang dieksekusi. Dalam proses ini, pengguna tidak

mengetahui kursor implisit. Oracle secara otomatis melakukan operasi OPEN, FETCH, dan CLOSE .

Kursor eksplisit: Kurs eksplisit digunakan untuk kueri yang menghasilkan lebih dari satu baris data. Kursor ini secara eksplisit dideklarasikan di bagian DECLARE dari blok PL / SQL. Deklarasi ini memungkinkan untuk memproses secara berurutan setiap baris data saat kursor mengembalikannya. Dalam kursor eksplisit DECLARE,OPEN,FETCH, dan CLOSE operasi dilakukan oleh programmer.

Proses bekerja dengan kursor eksplisit:

- Deklarasikan: Kursor diinisialisasi ke area memori sementara.
- Buka: Kursor dibuka yang dideklarasikan, dan area memori sementara dialokasikan.
- Ambil: Kursor yang dideklarasikan dan dibuka sekarang bisa mengambil baris dari data.

- Tutup: Pernyataan CLOSE menonaktifkan kursor, dan melepaskan area memori sementara.

Menampilkan data dengan menggunakan PL/SQL

- Gunakan perintah SELECT untuk mengambil data dari table
- *select_list* berisi sedikitnya satu kolom pada table, atau tanda bintang (*) untuk semua baris, juga bisa berisi ekspresi, fungsi baris atau fungsi group.
- *variable_name* adalah variable scalar yang menerima nilai
- *record_name* nama RECORD PL/SQL yang menerima nilai
- *table_nama* table yang datanya diambil
- *condition* terdiri dari nama kolom, ekspresi, konstanta, operator pembandingan, termasuk variable PL/SQL dan konstanta

petunjuk menampilkan data menggunakan PL/SQL

- akhiri setiap perintah SQL dengan tanda titik koma (;)

- klausa INTO diperlukan pada saat mencantumkan perintah sql pada program perintah select yang ditulis dalam PL/SQL
- klausa where bersifat optional dan dapat digunakan untuk menetukan variable input, konstanta, literal atau ekspresi PL/SQL
- Pastikan untuk mencantumkan jumlah variable yang sama antara yang ditulis setelah SELECT dan pada bagian yang ditulis setelah INTO. Pastika type data yang digunakan oleh variable tersebut compatible dengan type data dari kolom/field yang diambil
- *Perintah SELECT pada PL/SQL*
- Membutuhkan klausa INTO
- Queri harus mengembalikan satu nilai dan hanya satu nilai saja untuk tiap variable

6.2 Handling Execeptions

Bagian ini memberikan ikhtisar pengecualian dalam pemrograman PL / SQL, yang mencakup topik berikut:

- Tentang pengecualian
- Jenis pengecualian

Tentang pengecualian

Pengecualian adalah kesalahan PL / SQL yang diajukan selama eksekusi program, baik secara implisit oleh TimesTen atau secara eksplisit oleh program Anda. Tangani pengecualian dengan menjebaknya dengan pawang atau menyebarkannya ke lingkungan panggilan.

Misalnya, jika pernyataan `SELECT` Anda mengembalikan beberapa baris, TimesTen mengembalikan kesalahan (pengecualian) saat runtime.Seperti ditunjukkan oleh contoh berikut, Anda akan melihat kesalahan TimesTen 8507, lalu pesan kesalahan `ORA` terkait. (Pesan `ORA` , yang awalnya didefinisikan untuk Oracle Database, juga diterapkan oleh TimesTen.)

```
Perintah> DECLARE
        > v_lname VARCHAR2 (15);
        > BEGIN
        > SELECT last_name INTO v_lname
        > FROM karyawan
        > WHERE first_name = 'John';
        > DBMS_OUTPUT.PUT_LINE ('Nama belakang adalah:' || v_lname);
        > END;
        > /

 8507: ORA-01422: pengambilan tepat akan melebihi jumlah baris yang diminta
```

```
8507: ORA-06512: pada baris 4

Perintah gagal
```

Anda dapat menangani pengecualian seperti itu di blok PL / SQL Anda sehingga program Anda selesai dengan sukses. Sebagai contoh:

```
Perintah> DECLARE
     > v_lname VARCHAR2 (15);
     > BEGIN
     > SELECT last_name INTO v_lname
     > FROM karyawan
     > WHERE first_name = 'John';
     > DBMS_OUTPUT.PUT_LINE ('Nama belakang adalah:' || v_lname);
     > PENGECUALIAN
```

```
> KETIKA TOO_MANY_ROWS THEN

> DBMS_OUTPUT.PUT_LINE ('Pernyataan SELECT Anda diambil beberapa

> baris. Pertimbangkan untuk menggunakan kursor. ');

> END;

> /

 Pernyataan SELECT Anda mengambil beberapa baris. Pertimbangkan untuk menggunakan kursor.


 Prosedur PL / SQL berhasil diselesaikan.
```

Jenis Exeception

Ada tiga jenis pengecualian:

- Pengecualian yang telah ditentukan sebelumnya adalah kondisi kesalahan yang didefinisikan oleh PL / SQL.
- Pengecualian yang tidak ditentukan sebelumnya mencakup kesalahan TimesTen standar.
- Pengecualian yang ditentukan pengguna adalah pengecualian khusus untuk aplikasi Anda.

Di TimesTen, ketiga jenis pengecualian ini digunakan dengan cara yang sama seperti di Oracle Database.

Menjebak pengecualian

Bagian ini menjelaskan cara menjebak kesalahan TimesTen yang telah ditentukan sebelumnya atau kesalahan yang ditentukan pengguna.

Trap ping mendahului kesalahan TimesTen

Perangkap kesalahan TimesTen yang telah ditentukan dengan mereferensikan nama yang telah ditentukan dalam rutinitas penanganan pengecualian Anda. PL / SQL menyatakan

pengecualian yang telah ditentukan sebelumnya dalam paket STANDARD .

7. Pertemuan 7

Evaluasi

Tugas / Quis

8. Pertemuan 8

UTS

9. Pertemuan 9

Pembuatan *Procedure* & *Function*

9.1 Keuntungan Procedure dan Function

Adanya fungsi, program besar dapat dipisah menjadi program-program kecil. Ini menerapkan prinsip dalam pemrograman terstruktur dengan pendekatan top-down dan devide-and-conquer.

Program dapat dikerjakan oleh beberapa orang. Dalam hal ini bisa dikerjakan satu orang / team

mengerjakan suatu fungsi , sehingga dalam hal ini koordinasi mudah.

Dengan menggunakan Fungsi, alur logika program akan lebih jelas, dan karena fungsi merupakan implementasi suatu modul, pencarian kesalahan akan menjadi lebih mudah karena kesalahan dapat dilokalisasi dalam suatu modul tertentu saja.

Dalam memodifikasi program, dengan menggunakan fungsi ini, maka perubahan yang terjadi dalam memodifikasi suatu fungsi / modul tertentu tersebut, tidak akan mengganggu bagian program secara keseluruhan.

Dengan adanya fungsi, dapat lebih mudah dibuat dokumentasi.

Fungsi dapat dikatakan secara Reusability, karena suatu fungsi dapat digunakan kembali oleh fungsi lain atau oleh program lain.

9.2 Penulisan Procedure

Dibawah ini adalah sintak dasar untuk membuat Procedur di Oracle PL/SQL

```
CREATE [OR REPLACE] PROCEDURE
[nama_schema] nama_procedure
2       [(paramater [,paramater])]
3       IS | AS
4         [deklarasi]
5       BEGIN
6         [deklarasi]
7       [EXCEPTION]
8         [deklarasi_exception]
9       END [nama_procedure];
```

Penjelasan sintak:

- Tanda yang ada di dalam kurung siku "**[ ]**" merupakan bersifat opsional.
- Paramater yang digunakan untuk membuat Procedure ada 3 macam, yaitu:
  1. **IN** - merupakan parameter yang nilainya dapat digunakan (ditangkap) pada bagian badan procedure atau function. Anda tidak harus menyertakan IN pada paramater, karena secara default Oracle akan membuatkannya.
  2. **OUT** - merupakan parameter yang nilainya dapat digunakan oleh si pemanggil procedure atau function.
  3. **IN OUT** - merupakan parameter yang nilainya digunakan oleh procedure atau function yang kemudian diproses, dan selanjutnya dikembalikan kepada si pemanggil procedure atau function.

Menggunakan/Memanggil Procedure

Anda dapat menggunakan Procedure yang telah Anda buat dengan menggunakan perintah **EXEC nama_procedure**.

Contoh

1 EXEC tambah_bilangan;

tambah_bilangan adalah contoh nama Procedure yang telah Anda buat.

Catatan:

Sebelum Anda menjalankan Procedure, Anda harus mengaktifkan Server Output terlebih dahulu dengan perintah:

1 SQL> SET SERVEROUTPUT O

9.3 Penulisan Procedure Dengan Parameter

Subprogram adalah unit program / modul yang melakukan tugas tertentu. Subprogram ini dikombinasikan untuk membentuk program yang lebih besar. Ini pada dasarnya disebut 'Modular design'.Subprogram dapat dipanggil oleh subprogram atau program lain yang disebut **program pemanggilan** .

Subprogram dapat dibuat -

- Pada tingkat skema
- Di dalam sebuah paket
- Di dalam blok PL / SQL

Pada tingkat skema, subprogram adalah **subprogram mandiri** . Ini dibuat dengan PROSEDUR CREATE atau pernyataan CREATE FUNCTION. Ini disimpan dalam database dan dapat dihapus dengan pernyataan DROP PROSEDUR atau DROP FUNCTION.

Subprogram yang dibuat di dalam paket adalah **subprogram yang dikemas** . Ini disimpan dalam database dan bisa dihapus hanya saat paket dihapus dengan pernyataan DROP PACKAGE. Kita akan membahas paket di bab **'PL / SQL - Packages'** .

PL / SQL subprogram diberi nama blok PL / SQL yang dapat dipanggil dengan seperangkat parameter. PL / SQL menyediakan dua jenis subprogram -

- **Fungsi** - Subprogram ini mengembalikan satu nilai; terutama digunakan untuk menghitung dan mengembalikan sebuah nilai.
- **Prosedur** - Subprogram ini tidak mengembalikan nilai secara langsung; terutama digunakan untuk melakukan suatu tindakan.

Bab ini membahas aspek-aspek penting dari **prosedur PL / SQL** . Kita akan membahas **fungsi PL / SQL** di bab berikutnya.

9.4 IN – OUT Parameter

9.4.1 IN

Parameter IN memungkinkan Anda memberi nilai pada subprogram. Ini adalah parameter read-only. Di dalam subprogram, parameter IN bertindak seperti konstanta. Itu tidak bisa diberi nilai. Anda bisa melewati variabel, variabel terinisialisasi, terinisialisasi, atau ekspresi sebagai parameter IN. Anda juga dapat menginisialisasi ke nilai default; Namun, dalam kasus itu, hal itu dihilangkan dari panggilan subprogram. Ini adalah modus default parameter yang lewat. Parameter dilewatkan dengan referensi.

9.4.2 OUT

Parameter OUT mengembalikan nilai pada program pemanggilan. Di dalam subprogram, parameter OUT bertindak seperti sebuah variabel. Anda dapat mengubah nilainya dan mereferensikan nilainya setelah

menugaskannya. Parameter sebenarnya harus variabel dan dilewatkan nilai.

9.5 Deklarasi Subprograms

sub-program **PL/SQL** juga kita dapat mendeklarasikan suatu *variabel,* guna menampung nilai-nilai yang dibutuhkan dalam melakukan suatu proses tertentu. *variabel* harus di deklarasikan pada bagian **DECLARE**.

bentuk umum pendeklarasian variabel dalam PL/SQL adalah:

```
name_variabel tipe_data [:=nilai_default];
```

9.6 Handled Exceptions pada Procedure

Prosedur RAISE_APPLICATION_ERROR mem ungkinkan Anda mengeluarkan pesan kesalahan RAISE_APPLICATION_ERRORdite ntukan pengguna dari subprogram yang tersimpan. Dengan begitu, Anda dapat

melaporkan kesalahan pada aplikasi Anda dan menghindari pengecualian yang tidak tertangani.

Untuk
memanggil RAISE_APPLICATION_ERROR , gunakan sintaksnya

raise_application_error (error_number, message [, {TRUE | FALSE}]);
dimana error_number adalah bilangan bulat negatif di kisaran -20000 .. -20999 dan message adalah string karakter hingga 2048 byte. Jika parameter ketiga opsional TRUE , kesalahan ditempatkan pada tumpukan kesalahan sebelumnya. Jika parameternya FALSE (default), kesalahan akan menggantikan semua kesalahan sebelumnya.RAISE_APPLICATION_ERROR ad alah bagian dari paket DBMS_STANDARD , dan seperti pada paket STANDARD , Anda tidak perlu memenuhi syarat untuk referensi padanya.

Aplikasi hanya dapat memanggil raise_application_error dari

subprogram tersimpan (atau metode). Saat dipanggil, raise_application_error mengakhiri subprogram dan mengembalikan nomor kesalahan dan pesan yang ditetapkan pengguna ke aplikasi. Nomor kesalahan dan pesan bisa terjebak seperti kesalahan Oracle.

Aplikasi pemanggilan mendapatkan pengecualian PL / SQL, yang dapat memproses menggunakan fungsi pelaporan kesalahan SQLCODE dan SQLERRM pada penangan OTHERS . Selain itu, dapat menggunakan pragma EXCEPTION_INITuntuk memetakan nomor kesalahan tertentu yang dikembalikan oleh raise_application_error ke pengecualian

9.7 Penulisan Function

Di bawah ini adalah sintak dasar untuk membuat function di Oracle PL/SQL.
CREATE [OR REPLACE] FUNCTION

function_name[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
< function_body >
END [function_name];

Keterangan:

- **function_name** adalah nama function yang akan kita buat. Sifatnya mandatory.
- **parameter_name** adalah parameter yang akan ditambahkan. Sifatnya opsional
- **return_datatype** adalah tipe data yang akan dibalikkan (*return value*).
- **function_body** adalah bagian tubuh dari function yang berisi program PL/SQL.

9.8 Menjalankan Procedure dan Function

Kita dapat memanggil function yang telah kita buat dengan menggunakan program PL/SQL,

caranya adalah sebagai berikut:

```
DECLARE
  bil1 NUMBER(3);
  bil2 NUMBER(3);
BEGIN
  bil1 := 10;
 bil2 := 5;
 - Memanggil function tambah(bil1,bil2)
 DBMS_OUTPUT.PUT_LINE('Hasil
 penjumlahan '||bil1||' dan '||bil2||' adalah '||
 tambah(bil1, bil2));
 END;
```

Untuk menjalankan procedure yang telah kita buat, kita dapat menggunakan perintah **EXECUTE nama_procedure;** pada jendela SQLPlus.

```
SQL> set serveroutput on;
```

SQL> execute cetak_tulisan;

Selamat Belajar PL/SQL.. PL/SQL Itu Asyik

Selamat Belajar PL/SQL.. PL/SQL Itu Asyik

Selamat Belajar PL/SQL.. PL/SQL Itu Asyik

PL/SQL procedure successfully completed.

10. Pertemuan 10

Pembuatan *Trigger*

10.1 Types Triggers

Trigger adalah blok PL/SQL atau prosedur yang berhubungan dengan table, view, skema atau database yang dijalankan secara implicit pada saat terjadi sebuah event. Trigger merupakan store procedure yang dijalankan secara automatis saat user melakukan modifikasi data pada tabel. Modifikasi data yang dilakukan pada tabel yaitu berupa perintah

INSERT, UPDATE, dan DELETE. INSERT , UPDATE dan DELETE bisa digabung jadi satu trigger yang dinamakan Multiple Trigger.

Tipe dari trigger adalah :

· Application trigger : diaktifkan pada saat terjadi event yang berhubungan dengan sebuah aplikasi

· Database trigger : diaktifkan pada saat terjadi event yang berhubungan dengan data (seperti operasi DML) atau event yang berhubungan dengan sistem (semisal logon atau shutdown) yang terjadi pada sebuah skema atau database.

10.2 Pembuatan DML Triggers

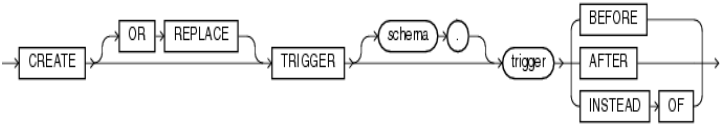
Oracle Database secara otomatis mengeksekusi trigger ketika kondisi tertentu terjadi.

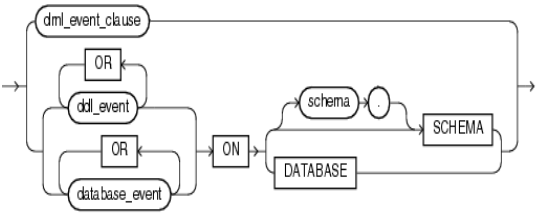
Saat Anda membuat pemicu, database memungkinkannya secara otomatis. Anda

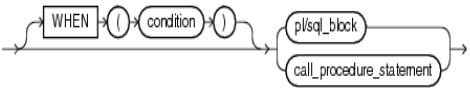
kemudian dapat menonaktifkan dan mengaktifkan pemicu dengan DISABLE dan ENABLE dari pernyataan ALTER TRIGGERatau ALTER TAB LE .

Sintaksis

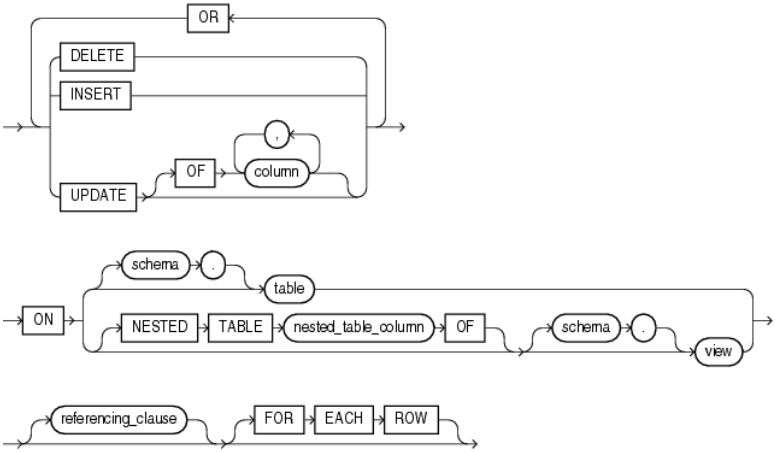
***create_trigger* :: =**







***DML_event_clause* :: =**



DML_event_clause

DML_event_clause memungkinkan Anda menentukan satu dari tiga pernyataan DML yang dapat menyebabkan pemicu menyala. Oracle Database memicu pemicu transaksi pengguna yang ada.

Anda tidak dapat menentukan kata kunci MERGE dalam *DML_event_clause* . Jika Anda menginginkan pemicu kebakaran terkait dengan operasi MERGE , Anda harus membuat pemicu pada

operasi INSERT dan UPDATE yang menyebabkan operasi MERGE terurai.

10.3Penghapusan Triggers

Tentukan DELETE jika Anda ingin database memecat pemicu setiap kali pernyataan DELETE menghapus satu baris dari tabel atau menghapus elemen dari tabel bersarang.

DAFTAR PUSTAKA

http://tania2011210828.blogspot.co.id/2013/09/penjualan-oracle.html

http://apriliawakhyuni.blogspot.co.id/2010/05/pengenalan-plsql.html

https://en.wikipedia.org/wiki/PL/SQL

http://nano-tutorial.blogspot.co.id/2015/03/variabel-pada-plsql.html

https://docs.oracle.com/cd/B10500_01/appdev.920/a96624/04_struc.htm

http://ariiefrahman.blogspot.co.id/2014/06/plsql.html

https://docs.oracle.com/database/121/TTPLS/exceptions.htm