DMH2C3

PEMROGRAMAN BASIS DATA

MODUL PRAKTIKUM

Hanya dipergunakan di lingkungan Fakultas Ilmu Terapan

Fakultas Ilmu Terapan
Telkom University
2016

DAFTAR PENYUSUN

1. Versi 2 : 2016 11 : Elis Hernawati, S.T., M.Kom.
   Ely Rosely, Ir., M.B.S.

Daftar Isi

1 BAB I STRUKTUR BLOK ANONIM PL/SQL DAN TIPE DATA

1.1 IDENTITAS

Kajian

Pengenalan blok anomim PL/SQL

Topik

1. Kerangka blok anonim PL/SQL
2. Tipe Data Skalar dan Komposit

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol: O'Reilly Media, 2009.
2. Benjamin Rosenzweig, Elena Silvestrova. Oracle PL/SQL by Example 3rd edition. Prentice Hall PTR, 2003.
3. -.PL/SQL Framework. http://cisku.com/2013/07/18/plsql-framework/ (accessed January 2014).
4. -.Contoh PL/SQL Sederhana. http://cisku.com/2013/07/18/contoh-plsql-sederhana/ (accessed January 2014).
5. -.Hitung Diskon Pembelian Barang. http://cisku.com/2013/07/18/hitung-diskon-pembelian-barang/ (accessed January 2014).

Kompetensi Utama

1. Mampu mendefinisikan struktur blok anonim PL/SQL
2. Mampu membuat blok anomim PL/SQL sederhana
3. Mampu mendefinisikan variabel pada blok anonim PL/SQL
4. Mampu menerapkan tipe data skalar pada variabel
5. Mampu menggunakan sintaks %TYPE dan %ROWTYPE pada blok anonim PL/SQL
6. Mampu menggunakan 3 jeni s record pada blok anonim PL/SQL
   - Table Based Record
   - Cursor Based Record
   - User-Defined Record

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Jurnal : Hasil Pengamatan 60%
2. Tugas Akhir 40%

1.2 PERTANYAAN PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika selesai, kumpulkan kepada asisten anda. Waktu maksimal 15 menit.

1. Apa yang anda ketahui tentang SQL dan PL/SQL? Jelaskan perbedaannya!

2. Apa yang anda ketahui tentang SQL*Plus? Jelaskan fungsi perintah SET SERVEROUTPUT ON pada SQL*Plus!

3. Jelaskan perintah DBMS_OUTPUT.PUT_LINE pada SQL*Plus!

4. Apa yang anda ketahui tentang tipe data dan variabel? Berikan contoh penggunaan pada Bahasa pemrograman yang anda kuasai!

5. Apakah blok anomim PL/SQL harus selalu memiliki variabel? berikan alasannya!

6. Apakah beda Antara variabel dengan konstanta? Berikan contoh pada Bahasa pemrograman yang anda kuasai!

7. Sebutkan jenis-jenis variabel yang terdapat pada blok anonim PL/SQL!

8. Apa yang anda ketahui tentang record? Berikan contoh penerapannya pada Bahasa pemgrograman yang anda ketahui atau buatlah algoritmanya!

9. Perhatikan blok kode program berikut

```
1.    DECLARE
1.    a CONSTANT INT := 15;
2.    b INT := 10;
3.    c NUMBER;
4.    BEGIN
5.    a     := 10;
6.    b     := 40;
7.    c     := (a+b-a)/b*a;
8.    DBMS_OUTPUT.PUT_LINE('Result = '||c||' dollar');
9.    END;
10.   /
```

a. Sebutkan bagian yang menjadi blok Header, blok Body dan blok Exception!
b. Sebutkan nama variabel beserta tipe datanya (sebutkan baris ke berapanya)!
c. Tuliskan output dari blok anonim PL/SQL tersebut!

d. Apakah konstanta itu? terdapat pada baris ke berapa pada blok anonim PL/SQL di atas? Jelaskan perbedaan Antara konstanta dengan variabel!

9. Diketahui tabel Kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

Perhatikan blok kode program berikut:

```
1. DECLARE
2.    varKdKelas    NUMBER := 7;
3.    varNmKelas    VARCHAR2(9);
4.    var2          NUMBER;
5.    hasil         VARCHAR2(9);
6. BEGIN
7.     SELECT  namakelas  INTO  varNmKelas  FROM  kelas  WHERE
       ROWNUM=1;
8.    var2 := 10;
9.    hasil := varNmKelas + var2;
10.       DBMS_OUTPUT.PUT_LINE (hasil);
11.     DBMS_OUTPUT.PUT_LINE (VARNMKELAS);
12.     DBMS_OUTPUT.PUT_LINE(vaR2);
13. END;
14. /
```

a. Tuliskan atribut apa saja yang ada pada tabel Kelas di atas beserta tipe datanya!
b. Sebutkan variabel apa saja yang ada pada blok PL/SQL di atas!
c. Tuliskan output yang dihasilkan dari blok anonim PL/SQL tersebut!(jika terdapat error tunjukkan baris ke berapa)?

1.3 PRAKTIK

1.3.1 Hitung Diskon Pembelian Barang

Pada bagian ini, akan dipelajari mengenai

1. Cara kerja blok anonim PL/SQL (Header-Body-Exception)
2. Pengenalan variabel substitusi

1.3.1.1 Soal

Pada sebuah toko atau supermarket, digunakan aplikasi komputer untuk mempermudah proses perhitungan harga barang yang di beli oleh konsumen. Seorang operator kasir akan menginputkan kode barang, harga barang, jumlah barang. Diasumsikan saat itu semua item barang sedang mendapatkan diskon 10%. Buatlah blok anonim PL/SQL untuk mendapatkan nilai total bayar konsumen (sudah dikurangi diskon 10%)!

1.3.1.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama atribut
1	Kode Barang
2	Harga Barang
3	Jumlah Barang
4	Harga Diskon
5	Jumlah Bayar

Kode Barang mendefinikan akan barang yang akan di beli konsumen. Harga barang merupakan harga barang satuan akan barang yang di beli. Kode Barang, Harga barang, Jumlah barang diinputkan manual oleh operator. Jumlah barang merupakan jumlah pembelian atau faktor pengali dari harga satuan barang. Harga diskon merupakan faktor pengurang yang didapatkan dari mengalikan **harga satuan barang** dengan nilai diskon (10%). Jumlah bayar akan menampung nilai total barang yang di beli oleh konsumen yang telah dikurangi diskon.

2. Membuat blok anonim PL/SQL untuk menghitung total pembayaran setelah diskon

```
1.   kode_brg := &kode_barang;
2.   harga_brg := &harga_barang;
3.   jumlah_brg := &jumlah_barang;
4.   diskon := (harga_brg*0.1)*jumlah;
5.   jumlah_bayar := (harga*jumlah)-diskon;
6.   DBMS_OUTPUT.PUT_LINE('Jumlah Bayar : ');
7.   DBMS_OUTPUT.PUT_LINE(jumlah_bayar);
```

Baris 1 akan meminta user/operator untuk mengetikkan nilai kode barang

Baris 2 akan meminta user/operator untuk mengetikkan nilai harga satuan barang

Baris 3 akan meminta user/operator mengetikkan jumlah item barang yang di beli

Baris 4 akan dilakukan perhitungan nilai diskon yaitu **harga satuan x 10% x jumlah item tersebut**

Baris 5 akan dilakukan perhitungan jumlah total pembayaran yaitu (**harga satuan x jumlah item) – diskon** (baris 4)

Baris 6 akan menampilkan pernyataan Jumlah Bayar :

Baris 7 akan menampilkan jumlah pembayaran (baris 5)

1.3.1.3 Solusi Lengkap

```
1.  DECLARE
2.       kode_brg          NUMBER;
3.       harga_brg         NUMBER;
4.       jumlah_brg        NUMBER;
5.       diskon            NUMBER;
6.       jumlah_bayar      NUMBER;
7.
8.  BEGIN
9.      kode_brg := &kode_barang;
10.      harga_brg := &harga_barang;
11.      jumlah_brg := &jumlah_barang;
12.      diskon := (harga_brg*0.1)*jumlah;
13.      jumlah_bayar := (harga*jumlah)-diskon;
14.      DBMS_OUTPUT.PUT_LINE('Jumlah Bayar : ');
15.      DBMS_OUTPUT.PUT_LINE(jumlah_bayar);
16.      EXCEPTION
17.        WHEN OTHERS THEN
18.          DBMS_OUTPUT.PUT_LINE(SQLERRM);
19.      END;
20.      /
```

1.3.1.4 Pengamatan

1. Bagian manakah yang termasuk Header?
2. Bagian manakah yang termasuk Body?
3. Bagian manakah yang termasuk Exception?
4. Pada baris program ditemukan tanda & (dan) apakah itu?

5. Komponen blok anonim PL/SQL yang wajib ada apa saja?(berikan contoh sesuai contoh di bagian solusi lengkap di atas)!
6. Kapankan Exception akan dilakukan?
7. Bagaimanakan cara untuk mengubah program jika diskon yang diberikan sekarang adalah 20%?
8. Jika pada baris 6 jumlah_bayar diganti menjadi jbayar, baris berapakah yang harus diperbaiki agar program tetap berjalan dengan baik?

1.3.2 Menampilkan Data Pegawai

Pada bagian ini, akan dipelajari mengenai

1. Cara kerja blok anonim PL/SQL (Header-Body-Exception)
2. Penggunaan blok anonim PL/SQL dengan melibatkan tabel pada database

1.3.2.1 Soal

Diketahui tabel pegawai dengan isi sebagai berikut:

NIP	NAMA	GOLONGAN	LOKASI
ABC123	Andri Sulaiman	4A	Bandung
ABC234	Rahmad Darmawan	3D	Bandung
ABC345	Yuni Sara	3D	Cimahi

Buatlah blok anonim PL/SQL untuk menampilkan data NIP,nama dan golongan pegawai dengan nip=ABC123!

1.3.2.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama atribut
1	v_nip
2	v_nama
3	v_gol

v_nip merupakan variabel untuk menampung data nip, v_nama merupakan variabel untuk menampung data nama dan v_gol merupakan variabel untuk menampung data golongan dari tabel pegawai.

2. Membuat blok anonim PL/SQL untuk menampilkan data NIP, nama dan golongan

```
1. SELECT nip, nama, golongan INTO v_nip, v_nama, v_gol FROM
   pegawai WHERE nip='ABC123';
2. DBMS_OUTPUT.PUT_LINE(v_nip);
3. DBMS_OUTPUT.PUT_LINE(v_nama);
4. DBMS_OUTPUT.PUT_LINE(v_gol);
```

Baris 1 akan melakukan query untuk mengambil atribut nip, nama dan golongan dari tabel pagawai dengan nip='ABC123' untuk disimpan ke dalam variabel v_nip,v_nama,v_gol secara berurutan. Urutan tidak boleh terbalik karena akan mengakibatkan kesalahan data atau bahkan program akan mengalami error. Baris 2,3,4 akan menampilkan nilai nip, nama dan golongan dari pegawai dengan nip='ABC123'.

1.3.2.3 Solusi Lengkap

```
1.  DECLARE
2.       v_nip VARCHAR2(16);
3.       v_nama VARCHAR(64);
4.       v_gol VARCHAR2(2);
5.  BEGIN
6.       SELECT nip, nama, golongan INTO v_nip, v_nama, v_gol
         FROM   pegawai WHERE nip='ABC123';
7.       DBMS_OUTPUT.PUT_LINE(v_nip);
8.       DBMS_OUTPUT.PUT_LINE(v_nama);
9.       DBMS_OUTPUT.PUT_LINE(v_gol);
10.      EXCEPTION
11.        WHEN OTHERS THEN
12.          DBMS_OUTPUT.PUT_LINE(SQLERRM);
13. END;
14. /
```

1.3.2.4 Pengamatan

1. Bagian manakah yang termasuk Header?
2. Bagian manakah yang termasuk Body?
3. Bagian manakah yang termasuk Exception?
4. Apa yang terjadi jika baris ke-7 di tukar dengan baris ke-8?
5. Apa yang terjadi jika baris ke-14 dihilangkan?
6. Apa yang terjadi jika baris ke 13 dihilangkan?
7. Apa yang terjadi jika baris ke-10 s/d 12 dihilangkan?

1.3.3 Konversi Tipe Data

Pada bagian ini, akan dipelajari mengenai :

1. Assignment dari variabel dengan tipe data DATE ke variabel dengan tipe data CHAR (konversi DATE ke CHAR)
2. Assignment dari variabel dengan tipe data CHAR ke variabel dengan tipe data NUMBER (konversi CHAR ke NUMBER)

1.3.3.1 Soal

Hitunglah umur seorang mahasiswa jika diketahui tanggal lahirnya adalah 17 Februari 1990 !

Diketahui:

Umur = tanggal hari ini – tanggal lahir

1.3.3.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama Variabel
1	tglLahir
2	tglSekarang
3	Umur

tglLahir akan menampung nilai dari tanggal lahir mahasiswa (17 Februari 1990). tglSekarang akan menampung nilai tanggal hari ini (misal: 03 Februari 2014). Umur akan menyimpan nilai selisih tahun Antara tanggal sekarang dikurangi tanggal lahir.

2. Blok anonim PL/SQL untuk menghitung selisih umur

```
1. tglLahir := TO_DATE('17-FEB-1990','DD-MON-YYYY');
2. tglSekarang := SYSDATE;
3. umur := FLOOR((tglSekarang-tglLahir)/365);
4. DBMS_OUTPUT.PUT_LINE('Umur : ');
5. DBMS_OUTPUT.PUT_LINE(umur||' tahun');
```

Baris 1 akan menyimpan nilai tanggal lahir mahasiswa (17 Februari 1990) dengan tipe data CHAR yang kemudian di konversi menjadi tipe data DATE

Baris 2 akan menampung tanggal hari ini (SYSDATE)

Baris 3 akan melakukan perhitungan umur mahasiswa dalam tahun. Pengurangan tanggal secara default akan menghasilkan selisih dalam hari oleh karena itu perlu dibagi 365 untuk mendapatkan nilai tahun. Penggunaan fungsi FLOOR digunakan untuk melakukan pembualatan ke bawah.

Baris 4 akan menampilkan tulisan Umur :

Baris 5 akan menampilkan selisih umur dalam tahun

Solusi Lengkap

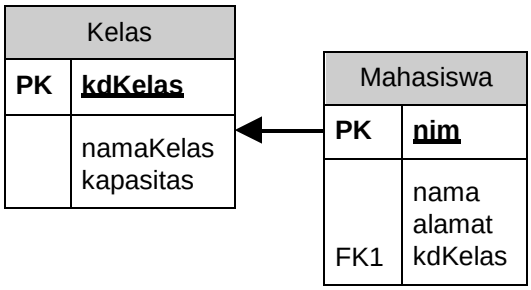
```
1.  DECLARE
2.       tglLahir          DATE;
3.       tglSekarang       DATE;
4.       Umur              NUMBER;
5.  BEGIN
6.      tglLahir := TO_DATE('17-FEB-1990','DD-MON-YYYY');
7.       tglSekarang := SYSDATE;
8.       umur := FLOOR((tglSekarang-tglLahir)/365);
9.       DBMS_OUTPUT.PUT_LINE('Umur : ');
10.      DBMS_OUTPUT.PUT_LINE(umur||' tahun');
11.      EXCEPTION
12.        WHEN OTHERS THEN
13.          DBMS_OUTPUT.PUT_LINE(SQLERRM);
14.  END;
15.  /
```

1.3.3.3 Pengamatan

1. Sebutkan semua variabel yang terdapat di blok PL/SQL di atas!
2. Mengapa pada baris ke-8 perlu digunakan fungsi FLOOR?
3. Apakah bedanya **umur** pada baris ke-9 dengan **umur** pada baris ke-10?
4. Apakah yang terjadi jika baris ke -9 di ubah menjadi
   DBMS_OUTPUT.PUT_LINE('Umur : '||umur||' tahun') ?
5. Bagaimana caranya agar tanggal lahir dapat di ubah-ubah sesuai nilai input dari keyboard?
6. Apa yang terjadi jika variabel umur pada baris ke-4 di ubah tipe datanya menjadi DATE?

1.3.4 Menampilkan Data Mahasiswa

Gunakan diagram relasi antar tabel berikut ini untuk menyelesaikan soal yang diberikan:



Pada bagian ini, akan dipelajari mengenai

1. Cara kerja blok anonim PL/SQL (Header-Body-Exception)
2. Penggunaan blok anonim PL/SQL dengan melibatkan tabel pada database

1.3.4.1 Soal

Diketahui tabel kelas dengan isi sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

Diketahui pula isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

Buatlah blok anonim PL/SQL untuk menampilkan data NIM dan nama mahasiswa yang memiliki angka nim terkecil pada kelas PIS-10-01!

1.3.4.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama atribut
1	vnim
2	vnama

vnim berfungsi untuk menampung nilai atribut nim dari tabel mahasiswa. vnama berfungsi untuk menampung nilai atribut nama dari tabel mahasiswa.

2. SQL query untuk melakukan join antara tabel kelas dengan tabel mahasiswa

```
1. SELECT m.nim, m.nama INTO vnim, vnama
2.       FROM kelas k JOIN mahasiswa m
3.             ON k.kdkelas = m.kdkelas
```

Query di atas akan melakukan **join** antara tabel kelas dengan tabel mahasiswa. Atribut yang dijadikan acuan join antara kedua tabel adalah atribut **kdkelas** dimana kdkelas pada tabel kelas adalah primary key dan kdkelas pada tabel mahasiswa adalah foreign key.

SELECT m.nim, m.nama INTO vnim, vnama artinya mengambil nilai nim dan nama dari tabel untuk kemudian **diisikan** ke dalam variabel vnim dan vnama, harap **perhatikan urutannya** jika tidak sama dapat mengakibatkan kesalahan program.

3. SQL query lanjutan untuk mengambil data mahasiswa kelas PIS-10-01 dengan NIM terkecil:

```
4. AND k.namakelas = 'PIS-10-01'
5. AND ROWNUM = 1
6. ORDER BY nim ASC;
```

Baris ke-4 akan melakukan **filter** terhadap data dimana data yang diambil adalah data dengan **nama kelas PIS-10-01**.

Baris ke-5 akan mengakibatkan **hanya 1 baris record** yang di ambil.

Baris ke-6 akan melakukan pengurutan terhadap data tabel dengan urutan nim terkecil lalu membesar.

1.3.4.3 Solusi Lengkap

```
1.  DECLARE
2.       vnim NUMBER;
3.       vnama VARCHAR2(20);
4.  BEGIN
5.       SELECT m.nim, m.nama INTO vnim, vnama
6.            from kelas k JOIN mahasiswa m
7.            ON k.kdkelas = m.kdkelas
8.            AND k.namakelas='PIS-10-01'
9.            AND ROWNUM=1
10.           ORDER BY nim ASC;
11.      DBMS_OUTPUT.PUT_LINE(vnim||' , '||vnama);
12.      EXCEPTION
13.        WHEN OTHERS THEN
14.          DBMS_OUTPUT.PUT_LINE(SQLERRM);
15. END;
16. /
```

1.3.4.4 Pengamatan

1. Mengapa pada baris ke-2 deklarasi tipe data vnama adalah VARCHAR2(20)?Jelaskan!
2. Apakah beda tipe data VARCHAR dengan VARCHAR2?
3. Silahkan modifikasi blok PL/SQL di atas dengan menggunakan JOIN jenis yang lain!
4. Apa yang terjadi jika baris ke-9 dihilangkan? Jelaskan!
5. Jika baris ke-8 nama kelas diganti PCA-10-01, apa outputnya?
6. Apakah boleh baris ke-9 di ubah menjadi
   `DBMS_OUTPUT.PUT_LINE(vnama||' , '||vnim);`
   Berikan penjelasan!
7. Apa yang terjadi jika baris ke-12 s/d 14 dihilangkan?
8. Apa yang terjadi jika baris ke-16 dihilangkan?

1.3.5 Table Based Record

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan table based record pada blok anonim PL/SQL dengan melibatkan tabel pada database
2. Penggunaan %ROWTYPE

1.3.5.1 Soal

Buatlah blok anonim PL/SQL untuk menampilkan data NIM dan nama mahasiswa yang memiliki NIM 3031001 dengan menggunakan **Table Based Record** dan sintaks **%ROWTYPE**! Gunakan tabel-tabel pada nomor sebelumnya.

1.3.5.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama atribut
1	mhs_rec

mhs_rec merupakan variabel berjenis record yang yang merupakan variabel bertipe komposit. Disebut komposit karena variabel tersebut dapat menampung lebih dari satu nilai atribut tabel. Penggunaan variabel komposit dibarengi dengan penggunaan sintaks %ROWTYPE.

2. Deklarasi blok anonim PL/SQL

```
1. mhs_rec mahasiswa%ROWTYPE;
```

Pada bagian deklarasi perlu dideklarasikan sebuah variabel berjenis record dengan nama mhs_rec. Variabel mhs_rec dideklarasikan bersama dengan sintaks %ROWTYPE. Deklarasi variabel di atas memiliki arti sebagai berikut:

mhs_rec : nama variabel
mahasiswa : nama tabel
%ROWTYPE : digunakan untuk menampung lebih dari 1 nilai atribut

3. SQL query mengambil nilai atribut dari tabel mahasiswa dengan nim 3031001

```
2. SELECT * INTO mhs_rec
3.       FROM mahasiswa
4.            WHERE nim=3031001;
```

Query di atas akan mengambil semua nilai atribut dari tabel mahasiswa dengan nim 3031001. Semua nilai atribut tersebut kemudian akan di tampung pada variabel berjenis record dengan nama mhs_rec. Deklarasi variabel mhs_rec dibarengi dengan penggunaan sintaks %ROWTYPE.

4. Tampilkan :

```
5. DBMS_OUTPUT.PUT_LINE (mhs_rec.nim);
6. DBMS_OUTPUT.PUT_LINE (mhs_rec.nama);
```

Baris ke-5 akan menampilkan nilai nim dari record mhs_rec. Penulisannya adalah nama variabel record diikuti dengan nama atribut tabel.

Baris ke-6 akan menampilkan nilai nama dari record mhs_rec.

1.3.5.3 Solusi Lengkap

```
1. DECLARE
2.    mhs_rec mahasiswa%ROWTYPE;
3.  BEGIN
4.    SELECT * INTO mhs_rec FROM mahasiswa WHERE nim=3031001;
5.    DBMS_OUTPUT.PUT_LINE (mhs_rec.nim);
6.    DBMS_OUTPUT.PUT_LINE (mhs_rec.nama);
7.  END;
8.  /
```

1.3.5.4 Pengamatan

1. Apakah %ROWTYPE itu?
2. Apakah %ROWTYPE dapat menampung beberapa nilai dengan tipe berbeda?
3. Modifikasi baris ke-4, ganti symbol * (bintang) dengan nilai yang diperlukan!
4. Tuliskan baris program jika perlu ditampilkan informasi tentang alamat mahasiswa!
5. Bagaimana cara penulisannya jika informasi nim dan nama harus ditampilkan dalam 1 baris!
6. Jika baris ke-5 dan ke-6 dihilangkan apa yang terjadi dengan output/tampilan program?
7. Modifikasi deklarasi blok anonim PL/SQL tersebut, ganti sintaks %ROWTYPE dengan sintaks %TYPE!
8. Modifikasi baris ke-4 pada solusi lengkap 3.3.1.3 dengan mengacu pada deklarasi variabel pada soal nomor 7!

1.3.6 User-Defined Record

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan **User_Defined record** pada blok anonim PL/SQL dengan melibatkan tabel pada database
2. Penggunaan %TYPE

1.3.6.1 Soal

Buatlah blok anonim PL/SQL untuk menampilkan data NIM dan nama mahasiswa yang memiliki NIM 3031001 dengan menggunakan **Table Based Record** dan sintaks **%ROWTYPE**!

1.3.6.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama atribut
1	mhs_type (TYPE)
2	mhs_rec

mhs_type merupakan sebuah deklarasi TYPE yang akan memuat sebuah record bentukan. Pembuatan TYPE pada dasarnya mirip dengan pembuatan tabel yaitu perlu didefinisikan atribut beserta tipe datanya. mhs_rec merupakan variabel berjenis record yang yang merupakan variabel bertipe komposit yang mengacu pada record mhs_type.

2. Deklarasi blok anonim PL/SQL

```
1. TYPE mhs_type IS RECORD (
2.   v_nim mahasiswa.nim%TYPE,
3.   v_nama mahasiswa.nama%TYPE);
4. mhs_rec mhs_type;
```

Baris ke-1 sampai ke-3 berisikan perintah untuk membuat sebuah record bentukan dengan nama mhs_type. Baris ke-2 akan medeklarasikan atribut v_nim dari record bentukan mhs_type. Baris ke-3 akan mendeklarasikan atribut v_nama dari record bentukan mhs_type.

Baris ke-4 akan mendeklarasikan variabel mhs_rec yang berjenis record dari record bentukan mhs_type, perhatikan di sini **tidak diperlukan** lagi sintaks %TYPE maupun %ROWTYPE.

3. SQL query untuk mengambil nilai atribut nim dan nama dari tabel mahasiswa dengan nim 3031001

```
5. SELECT nim, nama INTO mhs_rec
6.       FROM mahasiswa
7.             WHERE nim=3031001;
```

Query di atas akan mengambil nilai atribut nim dan nama dari tabel mahasiswa dengan nim 3031001. Semua nilai atribut tersebut kemudian akan di tampung pada variabel berjenis record dengan nama mhs_rec.

4. Tampilkan :

```
8. DBMS_OUTPUT.PUT_LINE (mhs_rec.v_nim);
9. DBMS_OUTPUT.PUT_LINE (mhs_rec.v_nama);
```

Baris ke-5 akan menampilkan nilai nim dari record mhs_rec. Penulisannya adalah nama variabel record diikuti dengan nama atribut tabel. Perhatikan atribut yang diambil adalah atribut record bentukan (**v_nim**) bukan atribut tabel(nim)

Baris ke-6 akan menampilkan nilai nama dari record mhs_rec.

1.3.6.3 Solusi Lengkap

```
1. DECLARE
2.    TYPE mhs_type IS RECORD (
3.      v_nim mahasiswa.nim%TYPE,
4.      v_nama mahasiswa.nama%TYPE);
5.    mhs_rec mhs_type;
6.  BEGIN
7.    SELECT nim, nama INTO mhs_rec FROM mahasiswa WHERE
      nim=3031001;
8.    DBMS_OUTPUT.PUT_LINE (mhs_rec.v_nim);
9.    DBMS_OUTPUT.PUT_LINE (mhs_rec.v_nama);
```

```
10. END;
11. /
```

1.3.6.4 Pengamatan

1. Sebutkan minimal 2 perbedaan antara user-defined record dengan record lainnya?
2. Apakah yang terjadi jika baris ke-4 di ubah menjadi **v_nama VARCHAR2(32);** ?Jelaskan!
3. Modifikasi solusi lengkap 3.3.3.3 jika perlu ditampilkan informasi tentang alamat mahasiswa!

1.4 TEST AKHIR

Selesaikan soal-soal berikut:

1. Buatlah sebuah blok anonim PL/SQL untuk melakukan perhitungan berikut :

```
a = 1
b = 4
c = a + b
output (c)
```

2. Buatlah sebuah blok anonim PL/SQL untuk menghitung luas lingkaran dengan algoritma berikut :

```
Input (jari-jari)
luas = 3.14 x jari-jari x jari-jari
output (luas)
```

3. Buatlah sebuah blok anonim PL/SQL untuk menghitung umur seseorang. Umur dihitung secara otomatis dengan menginputkan tanggal lahir. Misalkan tanggal hari ini adalah 03 Februari 2014.

```
Input
 03-02-1990
Output
        24
```

4. Dengan menggunakan tabel mahasiswa, buatlah sebuah blok anonim PL/SQL untuk menampilkan informasi berikut:

```
Nama :
Beta Gamma
Alamat :
Jl. Galaksi 501
NIM :
1110001
```

5. Diketahui BODY dari blok anonim PL/SQL sebagai berikut:

```
Var1 := SYSDATE;
Var2 := TO_CHAR(Var1, 'YYYY/MM');
Var3 := TO_CHAR(Var1, 'YYYY');
Var4 := TO_NUMBER(Var3);
Var5 := TO_DATE(Var2, 'YYYY-MM');
DBMS_OUTPUT.PUT_LINE(Var5);
```

Tuliskan deklarasi variabel dan tipe datanya! Apakah nilai output dari soal nomor 1?

Diketahui data pada tabel kelas adalah sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

Diketahui pula isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

6. Dengan menggunakan tabel mahasiswa dan kelas , buatlah sebuah blok anonim PL/SQL untuk menampilkan informasi **nim dan nama mahasiswa kelas PCA-10-01** yang nimnya **terbesar**!

7. Buatlah sebuah blok anonim PL/SQL untuk menampikan informasi kode, nama dan kapasitas **kelas** dengan dengan **table based record**!

8. Buatlah sebuah blok anonim PL/SQL untuk menampikan informasi kode, nama dan kapasitas **kelas** dengan dengan **user-defined record**!

9. Tampilkan jumlah total mahasiswa setiap prodi yang ada (PCA, PIS, PCE)!

10. Tampilkan informasi yang berisi nim, nama, namakelas mahasiswa prodi PIS!

1.5 RESUME

1.5.1 Jurnal Pengamatan

Selesaikan contoh kasus 1-3, kemudian jawablah pertanyaan-pertanyaan yang terdapat pada bagian pengamatan yang ada dibawah kode soal lengkapnya. Terdapat 15 soal pertanyaan disana, masing-masing pertanyaan memiliki komponen penilaian yang sama.

1.5.2 Lain-lain

1. Kumpulkan jawaban anda kepada para asisten sebelum praktikum diakhiri.
2. Media pengerjaan dan pengumpulan bebas, ikuti aturan yang telah ditetapkan oleh asisten praktikum.

2 BAB II STRUKTUR PERCABANGAN

2.1 IDENTITAS

Kajian

Struktur Percabangan

Topik

1. Pembuatan Struktur Percabangan IF THEN
2. Pembuatan Struktur Percabangan IF THEN ELSIF (Barsarang)
3. Pembuatan Struktur Percabangan CASE WHEN

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol: O'Reilly Media, 2009.
2. Benjamin Rosenzweig, Elena Silvestrova. Oracle PL/SQL by Example 3rd edition. Prentice Hall PTR, 2003.

Kompetensi Utama

1. Mampu membuat blok anonim PL/SQL yang memiliki struktur percabangan IF THEN
2. Mampu membuat blok anonim PL/SQL yang memiliki struktur percabangan IF THEN ELSIF (Bersarang)
3. Mampu membuat blok anonim PL/SQL yang memiliki struktur percabangan CASE WHEN

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Jurnal : Hasil Pengamatan 60%
2. Tugas Akhir 40%

2.2 PERTANYAAN PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika selesai, kumpulkan kepada asisten anda. Waktu maksimal 15 menit.

1. Apa yang anda ketahui tentang struktur percabangan? Berikan contoh dengan menggunakan Bahasa pemrogrman yang anda ketahui atau buatlah algoritmanya!

2. Mengapa sebuah program membutuhkan struktur percabangan? Berikan contoh penerapannya!

3. Perhatikan blok anonim PL/SQL berikut:

```
1. DECLARE
2.      Var1 NUMBER;
3.      Hasil VARCHAR2(16);
4. BEGIN
5.      Var1 := &n1;
6.      IF Var1 >= 70 THEN
7.           Hasil := 'Lulus';
8.      ELSE
9.           Hasil := 'Gagal';
10.     END IF;
11.     DBMS_OUTPUT.PUT_LINE (Hasil);
12. END;
13. /
```

   a. Tuliskan yang anda pahami tentang blok PL/SQL tersebut!
   b. Sebutkan variabel apa saja yang ada pada blok PL/SQL di atas beserta tipe datanya!
   c. Tuliskan output yang dihasilkan dari blok anonim PL/SQL tersebut!(jika terdapat eror tunjukkan baris ke berapa)?

4. Diketahui tabel Kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

   Buat blok PL/SQL untuk menampilkan **Keterangan** Kelas dari tabel kelas !

   PIS -> keterangan = MI
   PCE -> keterangan = TK
   PCA -> keterangan = KA

2.3 PRAKTIK

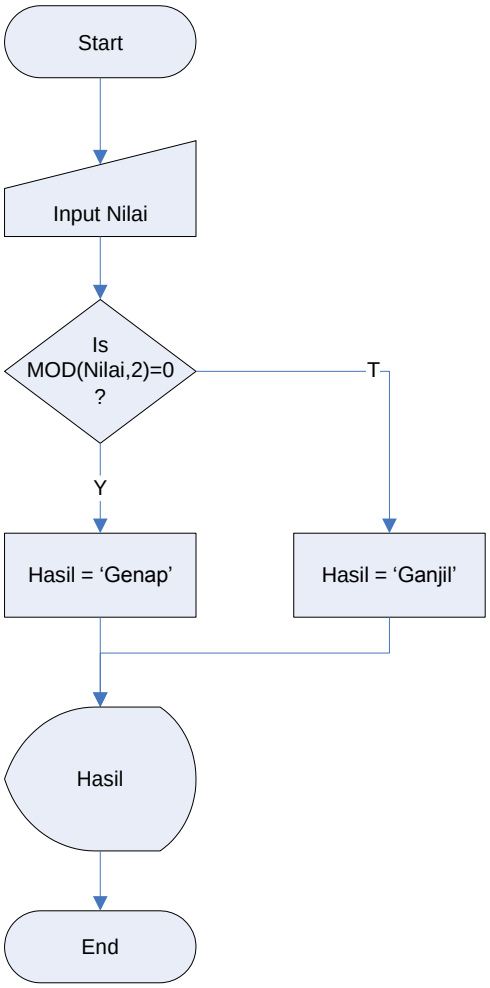
2.3.1 Ganjil atau Genap

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur percabangan IF THEN

2.3.1.1 Soal

Diketahui sebuah algoritma dalam bentuk flow map sebagai berikut:



Buatlah blok PL/SQL anonim dari algoritma tersebut untuk menentukan apakah suatu bilangan termasuk genap atau ganjil!

2.3.1.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel

1	Nilai
2	Hasil

Nilai merupakan variabel yang akan menerima input data dari user. Isi dari variabel nilai tersebut akan dibandingkan dengan suatu kondisi, apakah nilai jika dimoduluskan 2 hasilnya 0 atau tidak. Jika Ya maka variabel Hasil akan berisi keterangan **Genap**, tapi jika tidak maka variabel hasil akan bernilai **Ganjil**.

2. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil VARCHAR2(8);
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data VARCHAR2(8) berfungsi untuk menampung 2 nilai kemungkinan, apakah **Lulus** atau **Gagal.**

3. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
4. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai.

4. Struktur Percabangan :

```
5. IF MOD(Nilai,2) = 0 THEN
6.       Hasil := 'Genap';
7. ELSE
8.       Hasil := 'Ganjil';
9. END IF;
```

Baris ke-5 akan melakuan pengecekkan terhadap variabel Nilai, jika sisa hasil bagi dengan 2 (MOD 2) adalah 0 (nol) maka variabel Hasil pada baris ke-6 akan berisi **Genap**.

Baris ke-7 berisi sintaks ELSE akan mendefinisika suatu kondisi default dari struktur percabangan. Kondisi default tini akan dieksekusi jika tidak ada kondisi lain yang terpenuhi. Jika kondisi default ini aktif, baris selanjutnya (baris ke-8) akan dieksekusi dan variabel Hasil akan berisi **Ganjil**.

Baris ke-9 berisi END IF; merupakan penutup dari suatu struktur percabangan. END IF; harus selalu ada jika kita menggunakan struktur percabangan IF THEN.

5. Tampilan Output :

```
10. DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-10 akan menampilkan nilai variabel Hasil. Penulisan variabel pada blok PL/SQL **tidak case sensitive.**

2.3.1.3 Solusi Lengkap

```
1. DECLARE
2.     Nilai NUMBER;
3.     Hasil VARCHAR2(8);
4.   BEGIN
5.     DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
6.     Nilai := &input_nilai;
7.     IF MOD(Nilai,2)=0 THEN
8.       Hasil := 'Genap';
9.     ELSE
10.      Hasil := 'Ganjil';
11.    END IF;
12.    DBMS_OUTPUT.PUT_LINE (hasil);
13. END;
14. /
```

2.3.1.4 Pengamatan

1. Tuliskan dan jelaskan struktur percabangan IF THEN pada blok PL/SQL!
2. Apakah boleh tipe data pada variabel Hasil di ubah menjadi NUMBER? Tuliskan alasannya!
3. Apakah boleh tipe data pada variabel Hasil di ubah menjadi VARCHAR2(4)? Tuliskan alasannya!
4. Apakah boleh tipe data pada variabel Hasil di ubah menjadi VARCHAR2(128)? Tuliskan alasannya!
5. Apakah boleh tipe data pada variabel Nilai di ubah menjadi VARCHAR2(4)? Tuliskan alasannya!
6. Apakah blok anonim PL/SQL tersebut beroperasi sesuai dengan algoritmanya (Nilai=9, Hasil=Ganjil)? Jika tidak sebutkan dan perbaiki baris keberapanya!
7. Apa yang terjadi jika baris ke-11 dihilangkan? Jelaskan alasannya!

2.3.2 Ganjil atau Genap dengan CASE WHEN

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur percabangan CASE WHEN

2.3.2.1 Soal

Buatlah blok PL/SQL anonim dari algoritma pada nomor 2.3.1.1 untuk menentukan apakah suatu bilangan termasuk genap atau ganjil dengan struktur percabangan CASE WHEN!

2.3.2.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel

1	Nilai
2	Hasil

Nilai merupakan variabel yang akan menerima input data dari user. Isi dari variabel nilai tersebut akan dibandingkan dengan suatu kondisi, apakah nilai jika dimoduluskan 2 hasilnya 0 atau tidak. Jika Ya maka variabel Hasil akan berisi keterangan **Genap**, tapi jika tidak maka variabel hasil akan bernilai **Ganjil**.

2. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil VARCHAR2(8);
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data VARCHAR2(8) berfungsi untuk menampung 2 nilai kemungkinan, apakah **Lulus** atau **Gagal.**

3. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
4. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai.

4. Struktur Percabangan :

```
5. CASE
6.       WHEN MOD(Nilai,2)=0 THEN Hasil := 'Genap';
7. ELSE
8.       Hasil := 'Ganjil';
9. END CASE;
```

Baris ke-5 berisi sintaks pembuka struktur percabangan CASE. Penggunaan struktur percabangan CASE akan di tutup dengan END CASE; pada baris ke-9.

Baris ke-6 akan melakuan pengecekkan terhadap variabel Nilai, jika sisa hasil bagi dengan 2 (MOD 2) adalah 0 (nol) maka variabel Hasil pada baris ke-6 akan berisi **Genap**.

Baris ke-7 berisi sintaks ELSE akan mendefinisika suatu kondisi default dari struktur percabangan. Kondisi default tini akan dieksekusi jika tidak ada kondisi lain yang terpenuhi. Jika kondisi default ini aktif, baris selanjutnya (baris ke-8) akan dieksekusi dan variabel Hasil akan berisi **Ganjil**.

5. Tampilan Output :

```
10. DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-10 akan menampilkan nilai variabel Hasil. Penulisan variabel pada blok PL/SQL **tidak case sensitive.**

2.3.2.3 Solusi Lengkap

```
1. DECLARE
2.     Nilai NUMBER;
3.     Hasil VARCHAR2(8);
4.   BEGIN
5.     DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
6.     Nilai := &input_nilai;
7.     CASE
8.       WHEN MOD(Nilai,2)=0 THEN Hasil := 'Genap';
9.     ELSE
10.      Hasil := 'Ganjil';
11.    END CASE;
12.    DBMS_OUTPUT.PUT_LINE (hasil);
13. END;
14. /
```

2.3.2.4 Pengamatan

1. Tuliskan dan jelaskan struktur percabangan CASE WHEN pada blok PL/SQL!
2. Apakah blok anonim PL/SQL tersebut beroperasi sesuai dengan algoritmanya (Nilai=9, Hasil=Ganjil)? Jika tidak sebutkan dan perbaiki baris keberapanya!
3. Apa yang terjadi jika baris ke-11 dihilangkan? Jelaskan alasannya!
4. Jika dibandingkan dengan struktur percabangan IF THEN, lebih mudah mana penggunaannya? Jelaskan!

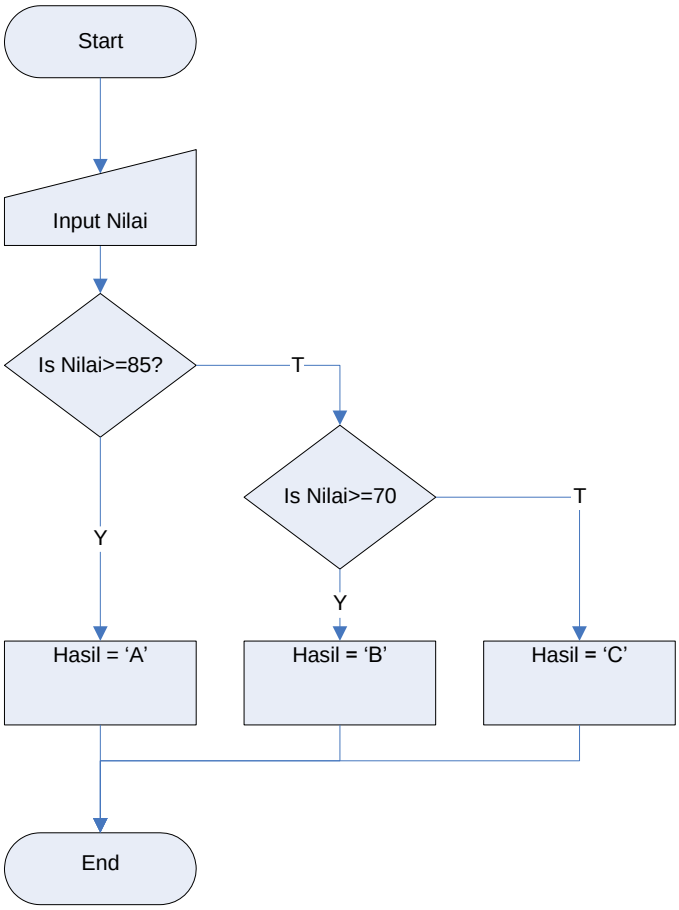
2.3.3 Input Nilai Mahasiswa

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur percabangan IF THEN ELSIF (bersarang)
2. Penggunaan struktur percabangan CASE WHEN (bersarang)

2.3.3.1 Soal

Diketahui sebuah algoritma dalam bentuk flow map sebagai berikut:



Buatlah blok anonim PL/SQL untuk menampilkan huruf mutu dari suatu nilai yang diinputkan!

2.3.3.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	Nilai
2	Hasil

Nilai merupakan variabel yang akan menerima input data dari user. Isi dari variabel nilai tersebut akan dibandingkan dengan suatu nilai, apakah nilai jika dibandingkan dengan 85 lebh besar atau lebih kecil? Jika lebih besar atau sama dengan maka variabel Hasil akan berisi keterangan **A**, tapi jika tidak maka variabel Nilai akan dibandingkan lagi dengan 70. Jika Nilai variabel lebih besar atau sama dengan 70 maka hasil akan bernilai **B**, tapi jika tidak maka Hasil akan bernilai **C**.

2. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil VARCHAR2(1);
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data VARCHAR2(1) berfungsi untuk menampung 3 nilai kemungkinan, apakah **A, B** atau **C.**

3. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
4. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai.

4. Struktur Percabangan :

```
5. IF Nilai >= 85 THEN
6.       Hasil := 'A';
7. ELSIF Nilai >= 70 THEN
8.            Hasil := 'B';
9. ELSE
10.           Hasil := 'C';
11. END IF;
```

Baris ke-5 akan melakuan pengecekkan terhadap variabel Nilai, jika nilai yang diinputkan >=85 maka variabel Hasil pada baris ke-6 akan berisi **A**.

Baris ke-7 berisi sintaks ELSIF (perhatikan **bukan ELSEIF**), baris ini akan melakukan pengecekkan **level ke-2** apakah nilai >= 70 dan nilai <85. Variabel hasil pada baris ke-8 akan bernilai **B**.

Baris ke-9 berisi sintaks ELSE akan mendefinisika suatu kondisi default dari struktur percabangan. Kondisi default tini akan dieksekusi jika tidak ada kondisi lain yang terpenuhi. Jika kondisi default ini aktif, baris selanjutnya (baris ke-10) akan dieksekusi dan variabel Hasil akan berisi **C**.

Baris ke-11 berisi END IF; merupakan penutup dari suatu struktur percabangan. END IF; harus selalu ada jika kita menggunakan struktur percabangan IF THEN.

5. Tampilan Output :

```
12. DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-12 akan menampilkan nilai variabel Hasil. Penulisan variabel pada blok PL/SQL **tidak case sensitive.**

Dengan menggunakan CASE WHEN

1. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil VARCHAR2(1);
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data VARCHAR2(1) berfungsi untuk menampung 3 nilai kemungkinan, apakah **A, B** atau **C.**

2. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
4. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai.

3. Struktur Percabangan :

```
5. CASE
6.   WHEN Nilai >= 85 THEN Hasil := 'A';
7.   WHEN Nilai >= 70 THEN Hasil := 'B';
8.   ELSE Hasil := 'C';
9. END CASE;
```

Baris ke-5 berisi sintaks pembuka struktur percabangan CASE. Penggunaan struktur percabangan CASE akan di tutup dengan END CASE; pada baris ke-9.

Baris ke-6 akan melakuan pengecekkan terhadap variabel Nilai, jika nilai yang diinputkan >=85 maka variabel hasil pada baris ke-6 akan berisi **A**.

Baris ke-7 akan melakukan pengecekkan **level ke-2** apakah nilai >= 70 dan nilai <85, jika terpenuhi maka variabel hasil pada baris ke-7 akan bernilai **B**.

Baris ke-9 berisi sintaks ELSE akan mendefinisika suatu kondisi default dari struktur percabangan. Kondisi default tini akan dieksekusi jika tidak ada kondisi lain yang terpenuhi. Jika kondisi default ini aktif, variabel Hasil pada baris ke-9 akan berisi **C**.

4. Tampilan Output :

```
10. DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-10 akan menampilkan nilai variabel Hasil. Penulisan variabel pada blok PL/SQL **tidak case sensitive.**

2.3.3.3 Solusi Lengkap

```
1. DECLARE
2.    Nilai NUMBER;
3.    Hasil VARCHAR2(1);
4.  BEGIN
5.    DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
6.    Nilai := &input_nilai;
7.    IF Nilai >= 85 THEN
8.      Hasil := 'A';
9.    ELSIF Nilai >= 70 THEN
10.   Hasil := 'B';
11.   ELSE
12.     Hasil := 'C';
13.   END IF;
```

```
14.    DBMS_OUTPUT.PUT_LINE (hasil);
15.  END;
16.  /
```

Dengan menggunakan CASE WHEN

```
1.  DECLARE
2.     Nilai NUMBER;
3.     Hasil VARCHAR2(1);
4.   BEGIN
5.     DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
6.     Nilai := &input_nilai;
7.     CASE
8.       WHEN Nilai >= 85 THEN Hasil := 'A';
9.       WHEN Nilai >= 70 THEN Hasil := 'B';
10.      ELSE Hasil := 'C';
11.    END CASE;
12.    DBMS_OUTPUT.PUT_LINE (hasil);
13.  END;
14.  /
```

2.3.3.4 Pengamatan

1. Tuliskan struktur percabangan bersarang!
2. Jika nilai input adalah 75, outputnya apa?
3. Jika nilai input adalah 100, outputnya apa?
4. Jika nilai input adalah 120, outputnya apa?
5. Modifikasi blok anonim PL/SQL tersebut sehingga nilai yang diinputkan harus berada dalam range 0-100!
6. Modifikasi blok anonim PL/SQL tersebut sehingga dapat menampilkan hasil D dan E!

2.3.4 Tampilkan Program Studi

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur percabangan IF THEN pada tabel di database
2. Penggunaan struktur percabangan CASE WHEN pada tabel di database

2.3.4.1 Soal

Diketahui data pada tabel kelas adalah sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40

5	PCA-10-01	40

Buatlah blok anonim PL/SQL untuk menampilkan informasi program studi berdasarkan nama kelas, PIS=MI, PCE=TK, PCA=KA (contoh: PIS-10-01 maka output adalah MI)!

2.3.4.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	varkelas
2	varprodi

Variabel varkelas akan menampung nilai dari nama kelas dari tabel kelas. Variabel varprodi akan menampung hasil konversi dari nama kelas menjadi nama program studi.

2. Deklarasi blok anonim PL/SQL

```
1. varkelas VARCHAR2(16);
2. varprodi VARCHAR2(16);
```

Baris ke-1 mendeklarasikan varkelas dengan tipe data VARCHAR2(16).
Baris ke-2 mendeklarasikan varprodi dengan tipe data VARCHAR2(16).

3. SQL query untuk mengambil nilai atribut nim dan nama dari tabel mahasiswa dengan nim 3031001

```
3. SELECT namakelas INTO varkelas
4.       FROM kelas
5.            WHERE kdkelas=1;
```

Query di atas akan mengambil nilai atribut namakelas dari tabel kelas dengan kode kelas=1. Nilai atribut tersebut kemudian akan di tampung pada variabel varkelas.

4. Struktur percabangan

```
6. IF substr(varkelas,1,3) = 'PIS' THEN
7.       keterangan := 'MI';
8. END IF;
```

Baris ke-6 akan melakukan pengecekkan terhadap nilai variabel varkelas. Tiga karakter pertama akan di ambil dan dibandingkan dengan kata **PIS**. Jika sama maka baris ke-7 akan mengisi variabel keterangan dengan **MI**. Baris ke-8 wajib ada jika sebuah struktur percabangan di buat.

5. Tampilkan :

```
9.  DBMS_OUTPUT.PUT_LINE (keterangan);
```

Baris ke-9 akan menampilkan nilai variabel keterangan.

Dengan menggunakan CASE WHEN:

1. Deklarasi blok anonim PL/SQL

```
1. varkelas VARCHAR2(16);
2. varprodi VARCHAR2(16);
```

Baris ke-1 mendeklarasikan varkelas dengan tipe data VARCHAR2(16).
Baris ke-2 mendeklarasikan varprodi dengan tipe data VARCHAR2(16).

2. SQL query untuk mengambil nilai atribut nim dan nama dari tabel mahasiswa dengan nim 3031001

```
3. SELECT namakelas INTO varkelas
4.       FROM kelas
5.            WHERE kdkelas=1;
```

Query di atas akan mengambil nilai atribut namakelas dari tabel kelas dengan kode kelas=1. Nilai atribut tersebut kemudian akan di tampung pada variabel varkelas.

3. Struktur percabangan

```
6. CASE
7.   WHEN substr(varkelas,1,3) = 'PIS' THEN
8.       keterangan := 'MI';
9. END CASE;
```

Baris ke-7 akan melakukan pengecekkan terhadap nilai variabel varkelas. Tiga karakter pertama akan di ambil dan dibandingkan dengan kata **PIS**. Jika sama maka baris ke-8 akan mengisi variabel keterangan dengan **MI**.

4. Tampilkan :

```
10. DBMS_OUTPUT.PUT_LINE (keterangan);
```

Baris ke-10 akan menampilkan nilai variabel keterangan.

2.3.4.3 Solusi Lengkap

```
1. DECLARE
2.    varkelas VARCHAR2(16);
3.    varprodi VARCHAR2(16);
4.  BEGIN
5.    SELECT namakelas INTO varkelas
6.      FROM kelas
7.           WHERE kdkelas=1;
10.   IF substr(varkelas,1,3) = 'PIS' THEN
11.     keterangan := 'MI';
12.   END IF;
8.    DBMS_OUTPUT.PUT_LINE (keterangan);
9.  END;
10. /
```

2.3.4.4 Pengamatan

1. Blok anonim PL/SQL pada solusi lengkap 4.3.3.3 memiliki kekurangan, sebutkan kekurangannya dan tuliskan kekurangannya sehingga blok anonim PL/SQL tersebut menjadi benar!
2. Jika tipe data varkelas pada baris ke-2 di ubah menjadi NUMBER, apa yang terjadi?
3. Jika baris ke-7 dihilangkan, apa yang terjadi? Jelaskan!
4. Apa yang terjadi jika baris 5,6,7 di tukar dengan baris ke 10,11,12? Jelaskan!

2.4 TEST AKHIR

Diketahui data pada tabel kelas adalah sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

Diketahui pula isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

Selesaikan soal-soal berikut:

1. Buat blok PL/SQL untuk menampilkan **Keterangan** Kelas dari tabel kelas !

   PIS -> keterangan = MI
   PCE -> keterangan = TK
   PCA -> keterangan = KA

2. Buat blok PL/SQL untuk menampilkan nama kelas yang memiliki **jumlah total mahasiswa 2** dengan menggunakan struktur percabangan IF THEN dan CASE WHEN.

3. Buat blok PL/SQL yang menampilkan informasi Nim dan Nama Mahasiswa, jika Mahasiwa beralamat di **jl. Buahbatu** maka nama yang ditampilkan **hanya nama depannya** saja!

2.5 RESUME

2.5.1 Jurnal Pengamatan

Selesaikan contoh kasus 1-3, kemudian jawablah pertanyaan-pertanyaan yang terdapat pada bagian pengamatan yang ada dibawah kode soal lengkapnya. Terdapat 17 soal pertanyaan disana, masing-masing pertanyaan memiliki komponen penilaian yang sama.

2.5.2 Lain-lain

1. Kumpulkan jawaban anda kepada para asisten sebelum praktikum diakhiri.
2. Media pengerjaan dan pengumpulan bebas, ikuti aturan yang telah ditetapkan oleh asisten praktikum.

3 BAB III STRUKTUR PERULANGAN

3.1 IDENTITAS

Kajian

Struktur Perulangan

Topik

1. Pembuatan Struktur Percabangan FOR LOOP, LOOP END, dan WHILE LOOP

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol: O'Reilly Media, 2009.
2. Benjamin Rosenzweig, Elena Silvestrova. Oracle PL/SQL by Example 3rd edition. Prentice Hall PTR, 2003.

Kompetensi Utama

1. Mampu membuat blok anonim PL/SQL yang memiliki struktur Perulangan FOR LOOP
2. Mampu membuat blok anonim PL/SQL yang memiliki struktur Perulangan LOOP END
3. Mampu membuat blok anonim PL/SQL yang memiliki struktur Perulangan WHILE LOOP

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Jurnal : Hasil Pengamatan 60%
2. Tugas Akhir 40%

3.2 PERTANYAAN PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika selesai, kumpulkan kepada asisten anda. Waktu maksimal 15 menit.

1. Apa yang anda ketahui tentang struktur perulangan FOR? Berikan contoh koding programnya pada bahasa pemrograman yang anda ketahui atau tuliskan algoritnyanya!

2. Mengapa sebuah program membutuhkan struktur perulangan? Berikan contoh penerapannya!

3. Perhatikan blok anonim PL/SQL berikut:

```
1. DECLARE
2.    Jumlah NUMBER;
3.  BEGIN
4.    Jumlah := 10;
5.    FOR i IN 1..Jumlah LOOP
6.      DBMS_OUTPUT.PUT_LINE (i);
7.    END LOOP;
8.  END;
9.  /
```

   a. Tuliskan yang anda pahami tentang blok PL/SQL tersebut!
   b. Sebutkan variabel apa saja yang ada pada blok PL/SQL di atas beserta tipe datanya!
   c. Tuliskan output yang dihasilkan dari blok anonim PL/SQL tersebut!(jika terdapat eror tunjukkan baris ke berapa)?

4. Perhatikan blok anonim PL/SQL berikut:

```
1. DECLARE
2.    Jumlah NUMBER;
3.  BEGIN
4.    Jumlah := 0;
5.    LOOP
6.      Jumlah := Jumlah + 1;
7.      DBMS_OUTPUT.PUT_LINE (Jumlah);
8.      EXIT WHEN Jumlah=10;
9.    END LOOP;
10. END;
11. /
```

   a. Tuliskan yang anda pahami tentang blok PL/SQL tersebut!
   b. Tuliskan output yang dihasilkan dari blok anonim PL/SQL tersebut!(jika terdapat eror tunjukkan baris ke berapa)?

4. Diketahui tabel Kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	37
3	PIS-10-03	40
4	PIS-10-04	39
5	PIS-10-05	40

Buat blok anonim PL/SQL untuk menampilkan kode kelas yang memiliki kapasitas=37!

3.3 PRAKTIK

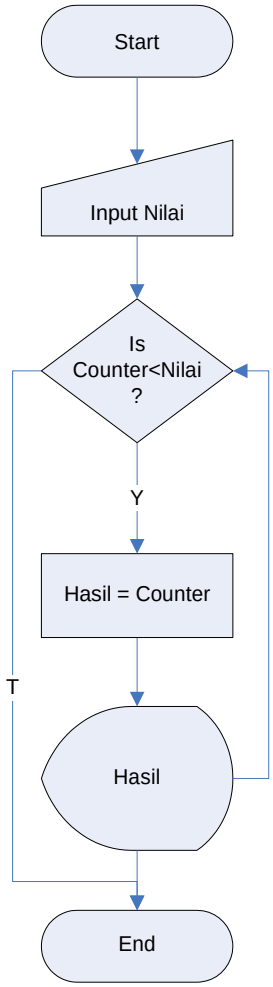
3.3.1 Menampilkan Bilangan dengan N Perulangan

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur perulangan FOR LOOP

3.3.1.1 Soal

Diketahui sebuah algoritma dalam bentuk flow map sebagai berikut:



Buatlah blok PL/SQL anonim dari algoritma tersebut untuk menampilkan bilangan bulat sebanyak Nilai yang diinputkan dengan menggunakan struktur perulangan FOR LOOP!

3.3.1.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	Nilai
2	Hasil

Nilai merupakan variabel yang akan menerima input data dari user. Isi dari variabel nilai tersebut akan dibandingkan dengan kondisi nilai suatu counter dari perulangan FOR LOOP, selama nilai counter lebih kecil dari isi variabel Nilai (Y / Ya) maka hasil akan diisi dengan isi variabel Nilai dan ditampilkan ke layar. Perulangan akan berhenti jika kondisi berada pada status T (Tidak) yaitu saat nilai Counter sama dengan nilai variabel Nilai.

2. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER:=1;
2. Hasil NUMBER;
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data NUMBER berfungsi untuk nilai Counter selama kondisi perulangan Y**.**

3. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
4. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai.

4. Struktur perulangan FOR LOOP :

```
5. FOR counter IN 1..Nilai LOOP
6.   Hasil := counter;
7.   DBMS_OUTPUT.PUT_LINE(Hasil);
8. END LOOP;
```

Baris ke-5 berisi sintaks pembuka struktur percabangan FOR LOOP. Pada baris ke-5 ditentukan kondisi perulangan untuk kondisi Y (Ya) yaitu membandingkan nilai counter dengan variabel Nilai. Penggunaan struktur percabangan FOR LOOP akan di tutup dengan END LOOP; pada baris ke-8.

Baris ke-6 akan mengisi variabel Hasil dengan nilai counter.

Baris ke-7 akan menampilkan nilai variabel Hasil ke layar.

5. Tampilan Output :

```
9.  DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-7 akan menampilkan nilai variabel Hasil. Penulisan variabel pada blok PL/SQL **tidak case sensitive.**

Dengan menggunakan LOOP END

1. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil NUMBER:=0;
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data NUMBER berfungsi untuk nilai Counter selama kondisi perulangan Y**.**

2. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Bilangan :');
4. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai. Nantinya perulangan aan dilakukan sebanyak bilangan yang diinputkan.

3. Struktur perulangan LOOP END:

```
5. LOOP
6.   Hasil := Hasil + 1;
7.   DBMS_OUTPUT.PUT_LINE(Hasil);
8.   EXIT WHEN Hasil = Nilai;
9. END LOOP;
```

Baris ke-5 berisi sintaks pembuka struktur perulangan LOOP END.

Baris ke-6 akan menambah satu (+1) variabel Hasil. Penambahan akan berhenti ketika nilai variabel Hasil sama dengan nilai variabel Nilai, pengecekkan dilakukan pad abaris ke-8.

4. Tampilan Output :

```
10. DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-7 akan menampilkan nilai variabel Hasil. Nilai variabel Hasil yang ditampilkan sebanyak bilangan yang diinputkan.

3.3.1.3 Solusi Lengkap

- menggunakan FOR LOOP

```
1. DECLARE
2.    Nilai NUMBER:=1;
3.    Hasil NUMBER;
4.  BEGIN
5.    DBMS_OUTPUT.PUT_LINE('Masukkan Nilai :');
6.    Nilai := &input_nilai;
7.    FOR counter IN 1..Nilai LOOP
8.      Hasil := counter;
9.      DBMS_OUTPUT.PUT_LINE(Hasil);
10.   END LOOP;
11. END;
12. /
```

- menggunakan LOOP END

```
1. DECLARE
2.    Nilai NUMBER;
3.    Hasil NUMBER:=0;
4.  BEGIN
5.    DBMS_OUTPUT.PUT_LINE('Masukkan Bilangan :');
6.    Nilai := &input_nilai;
7.    LOOP
8.      Hasil := Hasil + 1;
9.      DBMS_OUTPUT.PUT_LINE(Hasil);
10.     EXIT WHEN Hasil = Nilai;
11.   END LOOP;
12. END;
13. /
```

3.3.1.4 Pengamatan

- menggunakan FOR LOOP

1. Tuliskan dan jelaskan struktur percabangan FOR LOOP pada blok PL/SQL!
2. Tuliskan Output/tampilannya!
3. Apa yang terjadi jika baris ke-7 di ubah menjadi **FOR counter IN REVERSE 1..Nilai LOOP**?
4. Apa yang terjadi jika variabel Nilai diinisialisasikan dengan nilai 0 (Nilai NUMBER:=0)?
5. Apa yang terjadi jika pada baris ke-9 disisipkan **Hasil := Hasil+1**?
6. Apa yang terjadi jika pada baris ke-9 disisipkan **Counter := Counter+1**?

-menggunakan LOOP END

1. Tuliskan dan jelaskan struktur perulangan LOOP END pada blok PL/SQL!
2. Tuliskan Output/tampilannya solusi lengkap 7.3.1.3!
3. Apa yang terjadi jika variabel Hasil diinisialisasikan dengan nilai 1 (Nilai Hasil := 1)?
4. Apa yang terjadi jika pada baris ke-8 di ubah menjadi **Hasil := Hasil+2**?
5. Apa yang terjadi jika pada baris ke-10 dimodifikasi menjadi **EXIT WHEN Hasil < Nilai**?
6. Apa yang terjadi jika pada baris ke-10 dimodifikasi menjadi **EXIT WHEN Hasil > Nilai**?
7. Apa yang terjadi jika pada baris ke-10 dimodifikasi menjadi **EXIT WHEN Hasil <= Nilai**?
8. Apa yang terjadi jika pada baris ke-10 dimodifikasi menjadi **EXIT WHEN Hasil >= Nilai**?
9. Apa yang terjadi jika pada baris ke-10 dihilangkan?

3.3.2 Hitung Nilai Faktorial

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur perulangan dengan FOR LOOP untuk menghitung nilai faktorial

3.3.2.1 Soal

Buatlah blok anonim PL/SQL dengan struktur perulangan FOR LOOP untuk menghitung nilai faktorial (**misal : 5! = 5x4x3x2x1 = 120**) !

3.3.2.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	Nilai
2	Hasil

Nilai merupakan variabel yang akan menerima input data dari user. Isi dari variabel nilai tersebut akan di olah lebih lanjut dengan menggunakan perulangan sehingga didapatkan hasil nilai faktorialnya.

2. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil NUMBER:=1;
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data NUMBER yang akan menampung nilai hasil faktorial dari variabel Nilai. Variabel hasil perlu diinisialisasikan dengan nilai 1.

3. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Bilangan :');
4. Nilai := &bilangan;
```

Baris di atas meminta user melakukan input bilangan yang akan difaktorialkan. Hasil input tersebut akan di tampung pada variabel Nilai.

4. Struktur Perulangan :

```
5. FOR i IN 1..nilai LOOP
6.       hasil := hasil * i;
7. END LOOP;
```

Baris ke-5 berisi sintaks pembuka struktur perulangan FOR LOOP. Penggunaan struktur FOR LOOP akan di tutup dengan END LOOP; pada baris ke-7. Pada baris ke-5 akan dilakukan perulangan sebanyak **nilai** kali (misal nilai=5 maka perulangan akan dilakukan sebanyak 5 kali).

Baris ke-6 akan melakuan perhitungan nilai faktorial dari bilangan (misal nilai=5, dilakukan perhitungan sebanyak 5! = 5x4x3x2x1).

5. Tampilan Output :

```
8.  DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-8 akan menampilkan nilai variabel Hasil. Penulisan variabel pada blok PL/SQL **tidak case sensitive.**

Dengan menggunakan LOOP END:

1. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil NUMBER:=1;
3. a NUMBER:=0;
```

Pada bagian deklarasi perlu dideklarasikan 3 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data NUMBER yang akan menampung nilai hasil faktorial dari variabel Nilai. Variabel hasil perlu diinisialisasikan dengan nilai 1. Variabel a merupakan counter yang diinisialisasikan dengan nol (0).

2. Input data user dari keyboard

```
4. DBMS_OUTPUT.PUT_LINE('Masukkan Bilangan :');
5. Nilai := &bilangan;
```

Baris di atas meminta user melakukan input bilangan yang akan difaktorialkan. Hasil input tersebut akan di tampung pada variabel Nilai.

3. Struktur Perulangan :

```
1. hasil := hasil * Nilai;
2.  IF Nilai > 2 THEN
3.   LOOP
4.    Nilai := Nilai - 1;
5.    hasil := hasil * Nilai;
6.    a := a + 1;
7.    EXIT WHEN a > Nilai;
8.   END LOOP;
9.  END IF;
```

Baris ke-8 berisi sintaks pembuka struktur perulangan LOOP END. Penggunaan struktur LOOP END akan di tutup dengan END LOOP; pada baris ke-13. Pada baris ke-9 akan dilakukan pengurangan satu (-1) pada variabel Nilai, ini akan terus dilakukan hingga nilai variabel Nilai lebih kecil dari variabel a, selama perulangan variabel a ditambah satu (+1).

4. Tampilan Output :

```
10. DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-15 akan menampilkan nilai variabel Hasil. Penulisan variabel pada blok PL/SQL **tidak case sensitive.**

3.3.2.3 Solusi Lengkap

```
1. DECLARE
2.    Nilai NUMBER;
3.    Hasil NUMBER := 1;
4.  BEGIN
5.    DBMS_OUTPUT.PUT_LINE('Masukkan Bilangan :');
6.    Nilai := &bilangan;
7.    FOR i IN 1..nilai LOOP
8.      hasil := hasil * i;
9.    END LOOP;
10.   DBMS_OUTPUT.PUT_LINE (hasil);
11. END;
12. /
```

Dengan menggunakan LOOP END:

```
1. DECLARE
```

```
2.  Nilai NUMBER;
3.  hasil NUMBER := 1;
4.  a NUMBER := 0;
5. BEGIN
6.    Nilai := &bilangan;
7.    hasil := hasil * Nilai;
8.    IF Nilai > 2 THEN
9.     LOOP
10.     Nilai := Nilai - 1;
11.     hasil := hasil * Nilai;
12.     a := a + 1;
13.     EXIT WHEN a > Nilai;
14.    END LOOP;
15.   END IF;
16.   DBMS_OUTPUT.PUT_LINE (hasil);
17. END;
18. /
```

3.3.2.4 Pengamatan

- menggunakan FOR LOOP

1. Mengapa variabel hasil pada baris ke-3 perlu dideklarasikan dengan nilai 1?
2. Berapakah output dari blok anonim PL/SQL tersebut jika input bilangan adalah 10?
3. Apa maksud **hasil := hasil * i** pada baris ke-8?
4. Tuliskan outputnya jika baris ke-10 dipindahkan ke baris ke-9!

- menggunakan FOR LOOP

1. Mengapa variabel hasil pada baris ke-3 perlu dideklarasikan dengan nilai 1?
2. Mengapa variabel a perlu diinisialisasikan dengan nilai 0?
3. Berapakah output dari blok anonim PL/SQL tersebut jika input bilangan adalah 10?
4. Apa maksud **hasil := hasil * Nilai** pada baris ke-7?
5. Apa maksud baris ke-8?
6. Apa maksud baris ke=13?

3.3.3 Tampilkan Program Studi

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur perulangan FOR LOOP untuk menampilkan data dari tabel

3.3.3.1 Soal

Diketahui data pada tabel kelas adalah sebagai berikut:

Kelas

kdKelas	NamaKelas	Kapasitas
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

Buatlah blok anonim PL/SQL untuk menampilkan informasi program studi (kode kelas, nama kelas, kapasitas) dengan struktur perulangan FOR LOOP!

3.3.3.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	Ckelas (CURSOR)

Ckelas merupakan cursor yang akan menampung query dari tabel kelas.

2. Deklarasi blok anonim PL/SQL

```
1. CURSOR Ckelas IS SELECT kdkelas, namakelas, kapasitas FROM
   kelas;
```

Baris ke-1 mendeklarasikan cursor Ckelas yang berisi atribut kdkelas, namakelas dan kapasitas dari tabel kelas.

3. Struktur percabangan

```
2. FOR i IN Ckelas LOOP
3.    DBMS_OUTPUT.PUT_LINE(i.kdkelas||'-'||i.namakelas||'-
      '||i.kapasitas);
4. END LOOP;
```

Baris ke-2 akan membaca cursor Ckelas yang di tampung pada variabel i. Perulangan akan dilakukan sebanyak jumlah record dari cursor Ckelas.

Baris ke-3 akan menampilkan informasi seluruh kode kelas, nama kelas dan kapasitas yang ada pada tabel kelas.

3.3.3.3 Solusi Lengkap

```
1. DECLARE
2. CURSOR Ckelas IS SELECT kdkelas, namakelas, kapasitas FROM
   kelas;
3. BEGIN
4. FOR i IN Ckelas LOOP
5.       DBMS_OUTPUT.PUT_LINE(i.kdkelas||'-'||i.namakelas||'-
         '||i.kapasitas);
6. END LOOP;
7. END;
8. /
```

3.3.3.4 Pengamatan

1. Tuliskan output dari blok PL/SQL pada solusi lengkap 5.3.3.3!
2. Apa maksud i.kdkelas pada baris ke-5?
3. Modifikasi blok PL/SQL tersebut jika dideklarasikan 3 buah variabel (kode, nama, kapasitas)! (output tidak boleh berbeda)

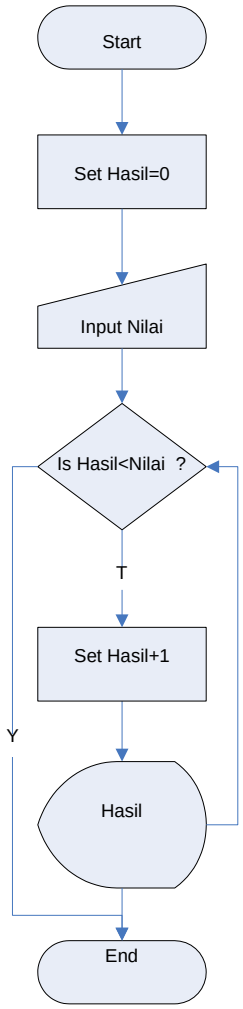
3.3.4 Menampilkan Bilangan dengan N Perulangan dengan While Loop

Pada bagian ini, akan dipelajari mengenai

1. Penggunaan struktur perulangan WHILE LOOP

3.3.4.1 Soal

Diketahui sebuah algoritma dalam bentuk flow map sebagai berikut:



Buatlah blok PL/SQL anonim dari algoritma tersebut untuk menampilkan bilangan bulat sebanyak Nilai yang diinputkan dengan menggunakan struktur perulangan WHILE LOOP!

3.3.4.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	Nilai
2	Hasil

Nilai merupakan variabel yang akan menerima input data dari user. Isi dari variabel nilai tersebut akan dibandingkan dengan variabel Hasil dari perulangan WHILE LOOP, selama nilai variabel Hasil lebih kecil dari isi variabel Nilai (kondisi T / Tidak) maka hasil akan ditambahkan satu (+1) dan ditampilkan ke layar. Perulangan akan berhenti jika kondisi berada pada status Y (Ya) yaitu saat nilai Hasil lebih besar atau sama dengan nilai variabel Nilai.

2. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil NUMBER:=0;
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data NUMBER berfungsi untuk nilai Counter selama kondisi perulangan Y.

3. Input data user dari keyboard

```
3. DBMS_OUTPUT.PUT_LINE('Masukkan Bilangan :');
4. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai. Nantinya perulangan aan dilakukan sebanyak bilangan yang diinputkan.

4. Struktur perulangan LOOP END:

```
5.  WHILE Hasil < Nilai LOOP
6.   Hasil := Hasil + 1;
7.   DBMS_OUTPUT.PUT_LINE(Hasil);
8.  END LOOP;
```

Baris ke-5 berisi sintaks pembuka struktur perulangan WHILE LOOP. Pada baris ini didefinisikan suatu kondisi yang akan bernilai TRUE / Ya.

Baris ke-6 akan menambah satu (+1) variabel Hasil. Penambahan akan berhenti ketika nilai variabel Hasil lebih besar atau sama dengan nilai variabel Nilai, pengecekkan dilakukan pad abaris ke-5.

5. Tampilan Output :

```
9.  DBMS_OUTPUT.PUT_LINE (hasil);
```

Baris ke-7 akan menampilkan nilai variabel Hasil. Nilai variabel Hasil yang ditampilkan sebanyak bilangan yang diinputkan.

3.3.4.3 Solusi Lengkap

```
1. DECLARE
2.    Nilai NUMBER;
3.    Hasil NUMBER:=0;
4.  BEGIN
5.    DBMS_OUTPUT.PUT_LINE('Masukkan Bilangan :');
6.    Nilai := &input_nilai;
7.    WHILE Hasil < Nilai LOOP
8.     Hasil := Hasil + 1;
9.     DBMS_OUTPUT.PUT_LINE(Hasil);
10.   END LOOP;
11. END;
12. /
```

3.3.4.4 Pengamatan

1. Tuliskan dan jelaskan struktur perulangan WHILE LOOP pada blok PL/SQL!
2. Tuliskan Output/tampilannya solusi lengkap 7.3.3.3!
3. Apa yang terjadi jika variabel Hasil diinisialisasikan dengan nilai 1 (Nilai Hasil := 1)?
4. Apa yang terjadi jika baris ke-7 di ubah menjadi **WHILE Hasil > Nilai LOOP**?
5. Apa yang terjadi jika pada baris ke-8 dihilangkan?Jelaskan!

3.4 TEST AKHIR

Diketahui data pada tabel kelas adalah sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

Diketahui pula isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

Selesaikan soal-soal berikut:

1. Buat blok PL/SQL untuk menampilkan informasi NIM, nama mahasiswa, nama kelas dengan menggunakan struktur perulangan FOR LOOP, LOOP END, dan WHILE LOOP!

2. Buat blok non modular PL/SQL untuk menghitung nilai pangkat n dari sebuah bilangan, bilangan dan n diinput dari keyboard! (gunakan struktur perulangan FOR LOOP, LOOP END, dan WHILE LOOP)
   Misal : n=3 -> bil=2 : $2^3 = 8$

3. Buat blok non modular PL/SQL untuk menampilkan bilangan ganjil antara 1-10, bilangan yang tampil adalah 3, 5, 7! (gunakan struktur perulangan FOR LOOP, LOOP END, dan WHILE LOOP)

3.5 RESUME

3.5.1 Jurnal Pengamatan

Selesaikan contoh kasus 1-3, kemudian jawablah pertanyaan-pertanyaan yang terdapat pada bagian pengamatan yang ada dibawah kode soal lengkapnya. Terdapat 13 soal pertanyaan disana, masing-masing pertanyaan memiliki komponen penilaian yang sama.

3.5.2 Lain-lain

1. Kumpulkan jawaban anda kepada para asisten sebelum praktikum diakhiri.
2. Media pengerjaan dan pengumpulan bebas, ikuti aturan yang telah ditetapkan oleh asisten praktikum.

4 BAB IV EXCEPTION

4.1 IDENTITAS

Kajian

Exception

Topik

1. Penggunaan Exception pada blok anonim PL/SQL

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol: O'Reilly Media, 2009.
2. Benjamin Rosenzweig, Elena Silvestrova. Oracle PL/SQL by Example 3rd edition. Prentice Hall PTR, 2003.

Kompetensi Utama

1. Mampu menerapkan Pre-defined Exception pada blok anonim PL/SQL
2. Mampu menerapkan Non Pre-Defined Exception pada blok anonim PL/SQL

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Jurnal : Hasil Pengamatan 60%
2. Tugas Akhir 40%

4.2 PERTANYAAN PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika selesai, kumpulkan kepada asisten anda. Waktu maksimal 15 menit.

1. Apa yang anda ketahui tentang Exception / Error Handling?

2. Tuliskan error handling pada Bahasa pemgograman yang anda ketahui (sertakan contoh)!

3. Perhatikan blok anonim PL/SQL berikut:

```
1. DECLARE
2.     Var1 NUMBER;
3.     Var2 VARCHAR2(30);
4. BEGIN
5.     SELECT 10/0, kodekelas INTO Var1, Var2 from Kelas WHERE
       rownum=1;
6.     DBMS_OUTPUT.PUT_LINE (Var1||' '||Var2);
7. EXCEPTION
8.     WHEN ZERO_DIVIDE THEN
9.          DBMS_OUTPUT.PUT_LINE ('Bilangan tak hingga');
10. END;
11. /
```

   a. Tuliskan yang anda pahami tentang blok PL/SQL tersebut!
   b. Blok PL/SQL tersebut akan memiliki error, pada baris ke berapa errornya dan apa penyebabnya?

4. Diketahui tabel Kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	37
3	PIS-10-03	40
4	PIS-10-04	39
5	PIS-10-05	40

   Buat blok anonim PL/SQL untuk menampilkan kode kelas dan nama kelas yang kapasitasnya 38!

4.3 PRAKTIK

4.3.1 Penggunaan Exception untuk Menangani Error Pembagian dengan Bilangan Nol

Pada bagian ini, akan dipelajari mengenai penggunaan Exception untuk menangani error akibat operasi pembagian dengan bilangan nol.

4.3.1.1 Soal

Buatlah blok PL/SQL anonim untuk menangani error yang diakibatkan sebuah bilangan dibagi dengan bilangan nol, bilangan diinputkan dari keyboard.

4.3.1.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	Nilai
2	Hasil

Nilai merupakan variabel yang akan menerima input data dari user. Variabel hasil akan menyimpan nilai hasil perhitungan.

2. Deklarasi blok anonim PL/SQL

```
1. Nilai NUMBER;
2. Hasil NUMBER:=12;
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel. Variabel Nilai memiliki tipe data NUMBER berfungsi untuk menampung nilai input dari user. Variabel Hasil memiliki tipe data NUMBER yang memiliki nilai awal / inisialisasi 12**.**

3. Input data user dari keyboard

```
3. Nilai := &input_nilai;
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Nilai.

4. Blok Exception PL/SQL

```
4. Hasil := Nilai / 0;
5. EXCEPTION
6.   WHEN ZERO_DIVIDE THEN
7.     DBMS_OUTPUT.PUT_LINE('Bilangan tidak boleh dibagi nol!');
```

Baris ke-4 akan melakukan perhitungan nilai variabel Nilai dibagi dengan bilangan nol, variabel Hasil akan menampung hasil perhitungan tersebut.

Baris ke-5 merupakan sintaks awal dari sebuah Exception. Baris ke-6 berisikan pernyataan kondisi yang akan menangkap error yang akan ditangani oleh Exception (ZERO_DIVIDE). Baris ke-7 akan menampilkan pesan kesalahan akibat dari kesalahan yang terjadi.

4.3.1.3 Solusi Lengkap

```
1. DECLARE
2.    Nilai NUMBER;
3.    Hasil NUMBER:=12;
4.  BEGIN
5.    Nilai := &input_nilai;
6.    Hasil := Nilai / 0;
7.  EXCEPTION
8.    WHEN ZERO_DIVIDE THEN
9.      DBMS_OUTPUT.PUT_LINE('Bilangan tidak boleh dibagi
    nol!');
10. END;
11. /
```

4.3.1.4 Pengamatan

1. Tuliskan dan jelaskan blok anonim PL/SQL yang memiliki Exception!
2. Tuliskan Output/tampilannya solusi lengkap 8.3.1.3!
3. Jika variabel Hasil nilainya ditampilkan pada baris ke-7 apa yang ditampilkan?
4. Tuliskan pengertian dari **WHEN ZERO_DIVIDE** pada bagis ke-8!
5. Apakah blok Exception harus selalu ditempatkan pada akhir sebuah blok PL/SQL? Jelaskan alasannya!

4.3.2 Exception untuk Menangani Kesalahan Pengisian Tanggal

Pada bagian ini akan dipelajari mengenai penggunaan Exception untuk menangani error yang disebabkan oleh kesalahan pengisian tipe data.

4.3.2.1 Soal

Buatlah blok anonim PL/SQL untuk menampilkan keterangan hari ini. Misal tanggal hari ini adalah 5 April 2014 maka keterangan hari yang ditampilkan adalah **Saturday**. Jika Input yang diberikan salah format maka akan ditampilkan kode error dan nama errornya.

4.3.2.2 Langkah Penyelesaian

1. Penentuan variabel program.

No	Nama variabel
1	Var1
2	Var2

Var1 merupakan variabel yang akan menerima input tanggal dari user. Var1 merupakan variabel yang akan menampung tanggal dalam format hari untuk kemudian ditampilkan.

2. Deklarasi blok anonim PL/SQL

```
1. Var1 DATE;
2. Var2 VARCHAR2(32);
```

Pada bagian deklarasi perlu dideklarasikan 2 buah variabel skalar. Variabel Var1 memiliki tipe data DATE berfungsi untuk menampung nilai input tanggal dari user. Variabel Var2 memiliki tipe data VARCHAR2(32) berfungsi untuk menampung nilai yang akan ditampilkan ke layar.

3. Input data user dari keyboard

```
3. Var1 := TO_DATE('&tanggal','DD-MON-YYYY');
4. Var2 := TO_CHAR(Var1,'DAY');
5. DBMS_OUTPUT.PUT_LINE ('Today is '||Var2);
```

Baris ke-3 memiliki variabel substitute &tanggal. Baris tersebut akan meminta user melakukan input tanggal, format tanggal yaitu DD-MON-YYYY (contoh: 15-APR-2014). Jika format tanggal betul maka baris ke-5 akan menampilkan **Today is Saturday**.

4. Exception :

```
6. EXCEPTION
7.       WHEN OTHERS THEN
8.            DBMS_OUTPUT.PUT_LINE (SQLCODE||'-'||SQLERRM);;
```

Baris ke-6 merupakan sintaks awal Exception. Baris ke-7 akan menangkap error yang mungkin terjadi. Jika terjadi error maka akan ditampilkan kode errornya pada baris ke-8 (SQLCODE) beserta keterangan errornya (SQLERRM).

4.3.2.3 Solusi Lengkap

```
1. DECLARE
2.       Var1 DATE;
3.       Var2 VARCHAR2(32);
4. BEGIN
5.       Var1 := TO_DATE('&tanggal','DD-MON-YYYY');
6.       Var2 := TO_CHAR(Var1,'DAY');
7.       DBMS_OUTPUT.PUT_LINE ('Today is '||Var2);
8. EXCEPTION
9.       WHEN OTHERS THEN
10.           DBMS_OUTPUT.PUT_LINE (SQLCODE||'-'||SQLERRM);
11. END;
12. /
```

4.3.2.4 Pengamatan

1. Perhatikan baris ke-5, apa maksud **TO_DATE('&tanggal','DD-MON-YYYY');** ?
2. Perhatikan baris ke-6, apa maksud **TO_CHAR(Var1,'DAY');** ?
3. Kapankah exception aktif? Berikan contoh input yang menyebabkan exception aktif!
4. Apakah SQLCODE dan SQLERRM itu?

5. Apakah maksud **OTHERS** pada baris ke-9?

4.3.3 Exception untuk Menangani Error pada Duplikasi Primary Key

Pada bagian ini akan dipelajari mengenai penanganan error jika sebuah atribut primary key akan diinputkan nilai yang sama.

4.3.3.1 Soal

Diketahui tabel kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	37
3	PIS-10-03	40
4	PIS-10-04	39
5	PIS-10-05	40

Tabel kelas tersebut memiliki constraint Primary Key pada atribut kdkelas. Jika dieksekusi perintah INSERT INTO kelas (kdkelas, namakelas, kapasitas) VALUES (5, 'PIS-10-06',40) akan mengakibatkan terjadinya error. Gunakan exception untuk menangkap error tersebut dan menampilkan **kode error** serta pesan **Kode Kelas tidak boleh Ganda!.**

4.3.3.2 Langkah Penyelesaian

1. Perintah insert pada body blok anonim PL/SQL

```
1. INSERT INTO kelas (kdkelas, namakelas, kapasitas) VALUES (5,
   'PIS-10-06', 40);
2. COMMIT;
```

Baris ke-1 akan melakukan insert data ke tabel kelas dengan kdkelas=6**.** Baris ke-2 akan melakukan commit data.

2. Blok Exception

```
3. EXCEPTION
4.   WHEN DUP_VAL_ON_INDEX THEN
5.     DBMS_OUTPUT.PUT_LINE(SQLCODE||' : Kode Kelas tidak boleh
       Ganda!');
```

Baris ke-4 akan mengecek error duplikasi pada primary key (DUP_VAL_ON_INDEX). Baris ke-5 akan menampilkan kode error dan pesan error.

4.3.3.3 Solusi Lengkap

```
1. BEGIN
2.     INSERT INTO kelas (kdkelas, namakelas, kapasitas) VALUES
       (5, 'PIS-10-06', 40);
3.     COMMIT;
4.  EXCEPTION
5.       WHEN DUP_VAL_ON_INDEX THEN
6.    DBMS_OUTPUT.PUT_LINE(SQLCODE||' : Kode Kelas tidak boleh
      Ganda!');
7.  END;
```

```
8.  /
```

4.3.3.4 Pengamatan

1. Apakah maksud DUP_VAL_ON_INDEX pada baris ke-5
2. Apa yang terjadi jika perintah baris ke-2 di ubah menjadi **INSERT INTO kelas (kdkelas, namakelas, kapasitas) VALUES ('PIS-10-06', 5, 40)**?
3. Jika soal no-2 di atas mengalami error, tuliskan penggunaan exception untuk dapat menampilkan pesan **Perintah Insert ada yang salah!**

4.3.4 Non Predefined Exception

Pada bagian ini akan dipelajari mengenai penanganan error dengan non predefined exception. Non predefined exception adalah exception yang mengubah kode error menjadi keyword khusus.

4.3.4.1 Soal

Diketahui tabel kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	37
3	PIS-10-03	40
4	PIS-10-04	39
5	PIS-10-05	40

Buatlah blok anonim PL/SQL untuk menampilkan informasi **kdkelas**. Gunakan SELECT INTO setelah BEGIN untuk menampilkan informasi tersebut, jika baris data yang didapatkan lebih dari satu, gunakan exception untuk menampilkan pesan **Hanya dapat menampilkan satu kode kelas saja**.

4.3.4.2 Langkah Penyelesaian

1. Deklarasi Variabel

```
1. Var1 NUMBER;
2. Kebanyakan EXCEPTION;
3. PRAGMA EXCEPTION_INIT (kebanyakan, -01422);
```

Baris ke-1 akan mendeklarasikan variabel **var1** dengan tipe data NUMBER. Baris ke-2 akan mendeklarasikan variabel Kebanyakan dengan tipa data EXCEPTION. Baris ke-3 akan mendeklarasikan inisialisasi Exception dimana akan memetakan kode error -01422 ke variabel kebanyakan.

2. Query SQL

```
4. SELECT kdkelas INTO var1 FROM kelas WHERE kapasitas=40;
5. DBMS_OUTPUT.PUT_LINE(var1);
```

Baris ke-4 akan melakukan query ke tabel kelas dengan mengambil atribut kdkelas yang memiliki kapasitas 40.

3. Blok Exception

```
6. EXCEPTION
7.   WHEN kebanyakan THEN
8.     DBMS_OUTPUT.PUT_LINE('Hanya dapat menampilkan satu kode
       kelas saja!');
```

Baris ke-7 akan menggunakan exception dengan nama **kebanyakan**. Jika terjadi error **kebanyakan** makan pesan error akan tampil.

4.3.4.3 Solusi Lengkap

```
1. DECLARE
2.  Var1 NUMBER;
3.  Kebanyakan EXCEPTION;
4.  PRAGMA EXCEPTION_INIT (kebanyakan, -01422);
5. BEGIN
6.  SELECT kdkelas INTO var1 FROM kelas WHERE kapasitas=40;
7.  DBMS_OUTPUT.PUT_LINE(var1);
8. EXCEPTION
9.   WHEN kebanyakan THEN
10.     DBMS_OUTPUT.PUT_LINE('Hanya dapat menampilkan satu kode
        kelas saja!');
11. END;
12. /
```

4.3.4.4 Pengamatan

1. Apakah maksud kebanyakan pada baris ke-9?
2. Apa yang terjadi jika perintah baris ke-6 di ubah menjadi **SELECT kdkelas INTO var1 FROM kelas WHERE kapasitas=37**?
3. Jelaskan pengertian baris ke-3!
4. Jelaskan pengertian baris ke-4!
5. Jika baris ke-3 di ubah menjadi **hasil EXCEPTION;** , modifikasi blok PL/SQL agar tetap berfungsi dengan baik!

4.4 TEST AKHIR

Diketahui data pada tabel kelas adalah sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40

5	PCA-10-01	40

Diketahui pula isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

Selesaikan soal-soal berikut:

1. Buat blok non modular PL/SQL untuk menggunakan Exception NO_DATA_FOUND ketika melakukan query select dari tabel kelas dan tidak didapatkan data yang sesuai!

2. Buat blok non modular PL/SQL menggunakan Exception TOO_MANY_ROWS ketika melakukan query menampilkan seluruh data dari tabel mahasiswa!

3. Modifikasi soal nomor 1, buatlah blok anonim PL/SQL untuk membuat exception KOSONG!

4.5 RESUME

4.5.1 Jurnal Pengamatan

Selesaikan contoh kasus 1-3, kemudian jawablah pertanyaan-pertanyaan yang terdapat pada bagian pengamatan yang ada dibawah kode soal lengkapnya. Terdapat 20 soal pertanyaan disana, masing-masing pertanyaan memiliki komponen penilaian yang sama.

4.5.2 Lain-lain

1. Kumpulkan jawaban anda kepada para asisten sebelum praktikum diakhiri.
2. Media pengerjaan dan pengumpulan bebas, ikuti aturan yang telah ditetapkan oleh asisten praktikum.

5 BAB V EXPLICIT CURSOR

5.1 IDENTITAS

Kajian

Explicit Cursor

Topik

1. Penggunaan Explicit Cursor pada blok PL/SQL

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol: O'Reilly Media, 2009.
2. Benjamin Rosenzweig, Elena Silvestrova. Oracle PL/SQL by Example 3rd edition. Prentice Hall PTR, 2003.
3. Michael Rosenblum, Paul Dorsey. Oracle PL/SQL for Dummies. Wiley Publishing, Inc., 2006.

Kompetensi Utama

3. Mampu menggunakan Explicit Cursor pada blok anonim PL/SQL

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Jurnal : Hasil Pengamatan 60%
2. Tugas Akhir 40%

5.2 PERTANYAAN PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika selesai, kumpulkan kepada asisten anda. Waktu maksimal 15 menit.

1. Apa yang anda ketahui tentang Cursor pada blok PL/SQL? Berikan contoh!

2. Apa yang anda ketahui dengan sintaks %TYPE dan %ROWTYPE? Berikan contoh penggunaannya!

3. Perhatikan blok anonim PL/SQL berikut:

```
1. DECLARE
2.     CURSOR ckelas IS SELECT kdkelas, namakelas FROM kelas;
3.     Kode kelas.kdkelas%TYPE;
4.     NamaK kelas.namakelas%TYPE;
5. BEGIN
6.     FOR i IN ckelas LOOP
7.       Kode := i.kdkelas;
8.       NamaK := i.namakelas;
9.       DBMS_OUTPUT.PUT_LINE (Kode);
10.      DBMS_OUTPUT.PUT_LINE (NamaK);
11.    END LOOP;
12.    EXCEPTION
13.          WHEN OTHERS THEN
14.               DBMS_OUTPUT.PUT_LINE ('SQLERRM');
15. END;
16. /
```

   a. Tuliskan yang anda pahami tentang blok PL/SQL tersebut!
   b. Jelaskan penggunaan struktur perulangan FOR LOOP pada blok PL/SQL tersebut!

4. Diketahui tabel Kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	37
3	PIS-10-03	40
4	PIS-10-04	39
5	PIS-10-05	40

   Buat blok anonim PL/SQL untuk menampilkan kode kelas dan nama kelas yang kapasitasnya 40!

5.3 PRAKTIK

5.3.1 Menampilkan Informasi Kelas dengan LOOP END

Pada bagian ini, akan dipelajari mengenai penggunaan Cursor Explicit untuk menampilkan informasi kelas dengan menggunakan perulangan LOOP END.

5.3.1.1 Soal

Buatlah blok anonim PL/SQL untuk menampilkan informasi kode dan nama kelas dengan menggunakan Cursor Explicit dan perulangan LOOP END!

5.3.1.2 Langkah Penyelesaian

1. Deklarasi blok anonim PL/SQL

```
1. CURSOR ckelas IS SELECT kdkelas, namakelas FROM kelas;
2. Reckelas ckelas%ROWTYPE;
```

Pada bagian deklarasi perlu dideklarasikan sebuah Cursor dengan nama ckelas yang menampung nilai atribut kdkelas dan namakelas. Variabel komposit Reckelas dipersiapkan untuk menampung nilai dari Cursor ckelas.

2. Blok Body PL/SQL

```
3. OPEN ckelas;
4.   LOOP
5.     FETCH ckelas INTO Reckelas;
6.     EXIT WHEN ckelas%NOTFOUND;
7.     DBMS_OUTPUT.PUT_LINE(Reckelas.kdkelas);
8.     DBMS_OUTPUT.PUT_LINE(Reckelas.namakelas);
9.   END LOOP;
10. CLOSE ckelas;
```

Baris ke-3 dan ke-10 merupakan perintah untuk membuka dan menutup Cursor ckelas.

Baris ke4 dan ke-9 merupakan perintah untuk membuka dan menutup struktur perulangan LOOP END.

Baris ke-5 mengisikan semua isi Cursor ckelas ke dalam variabel Reckelas (per baris/record), pengisian dilakukan terus menerus sampai ckelas habis (baris ke-6).

Baris ke-7 menampilkan isi variabel Reckelas pada atribut **kdkelas** ke layar.

Baris ke-8 menampilkan isi variabel Reckelas pada atribut **namakelas** ke layar.

5.3.1.3 Solusi Lengkap

```
1. DECLARE
2.       CURSOR ckelas IS SELECT kdkelas, namakelas FROM kelas;
3.       Reckelas ckelas%ROWTYPE;
4. BEGIN
5.       OPEN ckelas;
6.            LOOP
7.                 FETCH ckelas INTO Reckelas;
8.                 EXIT WHEN ckelas%NOTFOUND;
9.                 DBMS_OUTPUT.PUT_LINE(Reckelas.kdkelas);
10.                DBMS_OUTPUT.PUT_LINE(Reckelas.namakelas);
11.           END LOOP;
12.      CLOSE ckelas;
13. END;
14. /
```

5.3.1.4 Pengamatan

1. Jelaskan fungsi Cursor Explicit pada blok anonim PL/SQL!
2. Tuliskan Output/tampilannya solusi lengkap 9.3.1.3!
3. Tuliskan struktur utama blok anonim PL/SQL yang menggunakan Cursor Explicit dan struktur perulangan LOOP END!
4. Apa yang terjadi jika baris ke-5 dihilangkan? Jelaskan!
5. Apa yang terjadi jika baris ke-7 dihilangkan? Jelaskan!
6. Apa yang terjadi jika baris ke-8 dihilangkan? Jelaskan!
7. Apa yang terjadi jika baris ke-11 dihilangkan? Jelaskan!
8. Apa yang terjadi jika baris ke-12 dihilangkan? Jelaskan!
9. Modifikasi blok PL/SQL tersebut agar dapat menampilkan **namakelas** dan **kapasitas**!

5.3.2 Menampilkan Informasi Kelas dengan FOR LOOP

Pada bagian ini, akan dipelajari mengenai penggunaan Cursor Explicit untuk menampilkan informasi kelas dengan menggunakan perulangan FOR LOOP.

5.3.2.1 Soal

Buatlah blok anonim PL/SQL untuk menampilkan informasi kode dan nama kelas dengan menggunakan Cursor Explicit dan perulangan FOR LOOP!

5.3.2.2 Langkah Penyelesaian

1. Deklarasi blok anonim PL/SQL

```
1. CURSOR ckelas IS SELECT kdkelas, namakelas FROM kelas;
```

Pada bagian deklarasi perlu dideklarasikan sebuah Cursor dengan nama ckelas yang menampung nilai atribut kdkelas dan namakelas.

2. Blok Body PL/SQL

```
2. FOR Reckelas IN ckelas LOOP
3.       DBMS_OUTPUT.PUT_LINE(Reckelas.kdkelas);
4.       DBMS_OUTPUT.PUT_LINE(Reckelas.namakelas);
5. END LOOP;
```

Baris ke-2 dan ke-5 merupakan perintah untuk membuka dan menutup struktur perulangan FOR LOOP. Pada baris ke-2 terdapat deklarasi variabel Reckelas di dalam struktur FOR LOOP, variabel Reckelas akan menampung semua isi dari Cursor ckelas.

Baris ke-3 menampilkan isi variabel Reckelas pada atribut **kdkelas** ke layar.

Baris ke-4 menampilkan isi variabel Reckelas pada atribut **namakelas** ke layar.

5.3.2.3 Solusi Lengkap

```
1. DECLARE
2.       CURSOR ckelas IS SELECT kdkelas, namakelas FROM kelas;
3. BEGIN
4.   FOR Reckelas IN ckelas LOOP
5.       DBMS_OUTPUT.PUT_LINE(Reckelas.kdkelas);
6.       DBMS_OUTPUT.PUT_LINE(Reckelas.namakelas);
7.   END LOOP;
8. END;
9.  /
```

5.3.2.4 Pengamatan

1. Tuliskan Output/tampilannya solusi lengkap 9.3.2.3!
2. Tuliskan struktur utama blok anonim PL/SQL yang menggunakan Cursor Explicit dan struktur perulangan FOR LOOP!
3. Sebutkan yang mana variabel yang mana cursor pada baris ke-4!
4. Apa yang terjadi jika baris ke-7 dihilangkan? Jelaskan!
5. Apa yang terjadi jika baris ke-6 dihilangkan? Apakah terjadi error program?
6. Modifikasi blok PL/SQL tersebut agar dapat menampilkan **namakelas** dan **kapasitas**!

5.4 TEST AKHIR

Diketahui data pada tabel kelas adalah sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	40
3	PIS-10-03	40
4	PCE-10-01	40
5	PCA-10-01	40

Diketahui pula isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	4

Selesaikan soal-soal berikut:

1. Buat blok non modular PL/SQL menggunakan Explicit Cursor dan LOOP END untuk menampilkan informasi Nama Kelas, NIM, Nama Mahasiswa!

2. Buat blok non modular PL/SQL menggunakan Explicit Cursor dan LOOP END untuk menampilkan informasi Nama Mahasiswa, Nama Prodi (Nama Prodi ditentukan dari nama kelas: PIS=MI, PCE=TK, PCA=KA)!

3. Modifikasi soal nomor 1, gunakan struktur perulangan FOR LOOP!

4. Modifikasi soal nomor 2, gunakan struktur perulangan FOR LOOP!

5. Buat blok non modular PL/SQL menggunakan Explicit Cursor untuk menampilkan NIM, Nama Mahasiswa, Nama kelas dimana yang ditampikan adalah yang memiliki NIM genap!

6. Buatlah blok PL/SQL anonymous untuk menerima sebuah input prodi dan menampilkan jumlah mahasiswanya. Berikut contoh pemanggilannya :

   Input : MI ; Output : Jumlah mahasiswa = 2

   Input : TK; Output : Jumlah mahasiswa = 1

   Input : KA; Output : Jumlah mahasiswa = 2

5.5 RESUME

5.5.1 Jurnal Pengamatan

Selesaikan contoh kasus 1-2, kemudian jawablah pertanyaan-pertanyaan yang terdapat pada bagian pengamatan yang ada dibawah kode soal lengkapnya. Terdapat 15 soal pertanyaan disana, masing-masing pertanyaan memiliki komponen penilaian yang sama.

5.5.2 Lain-lain

1. Kumpulkan jawaban anda kepada para asisten sebelum praktikum diakhiri.
2. Media pengerjaan dan pengumpulan bebas, ikuti aturan yang telah ditetapkan oleh asisten praktikum.

6 BAB VI IMPLICIT CURSOR

6.1 IDENTITAS

Kajian

Implicit Cursor

Topik

1. Penggunaan Implicit Cursor pada blok PL/SQL

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol: O'Reilly Media, 2009.
2. Benjamin Rosenzweig, Elena Silvestrova. Oracle PL/SQL by Example 3rd edition. Prentice Hall PTR, 2003.
3. Michael Rosenblum, Paul Dorsey. Oracle PL/SQL for Dummies. Wiley Publishing, Inc., 2006.

Kompetensi Utama

1. Mampu menggunakan Implicit Cursor pada blok anonim PL/SQL

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Jurnal : Hasil Pengamatan 60%
2. Tugas Akhir 40%

6.2 PERTANYAAN PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika selesai, kumpulkan kepada asisten anda. Waktu maksimal 15 menit.

1. Apa yang anda ketahui tentang Explicit Cursor pada blok PL/SQL? Berikan contoh!

2. Apa yang anda ketahui tentang Implicit Cursor pada blok PL/SQL? Jelaskan perbedaannya dengan Explicit Cursor!

3. Perhatikan blok anonim PL/SQL berikut:

```
1. DECLARE
2.     Var2 kelas.kdkelas%TYPE;
3. BEGIN
4.     SELECT kdkelas INTO Var2 from Kelas WHERE kdkelas=1;
5.     DBMS_OUTPUT.PUT_LINE (Var1||' '||Var2);
6. EXCEPTION
7.     WHEN NO_DATA_FOUND THEN
8.          DBMS_OUTPUT.PUT_LINE ('Kelas Tidak Ditemukan');
9.  END;
10. /
```

   a. Tuliskan yang anda pahami tentang blok PL/SQL tersebut!
   b. Pada baris ke berapa terdapat Implicit Cursor?

4. Diketahui tabel Kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PIS-10-02	37
3	PIS-10-03	40
4	PIS-10-04	39
5	PIS-10-05	40

   Buat blok anonim PL/SQL untuk menampilkan kode kelas dan nama kelas yang kapasitasnya 37!

6.3 PRAKTIK

6.3.1 Melakukan Update pada Tabel Mahasiswa dengan Kondisi Tertentu

Pada bagian ini akan dipelajari mengenai penggunaan Implicit Cursor untuk melakukan pengubahan nilai dari tabel kelas

Diketahui isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

6.3.1.1 Soal

Buatlah blok anonim PL/SQL untuk melakukan pengubahan data mahasiswa, misal akan mengubah nama Abdul Hakim menjadi Abdul Azis (gunakan Implicit Cursor)!

6.3.1.2 Langkah Penyelesaian

1. Deklarasi blok anonim PL/SQL

```
1. Vnama mahasiswa.nama%type;
```

Pada bagian deklarasi perlu dideklarasikan 1 buah variabel. Variabel Vnama memiliki tipe data yang sesuai dengan tipe data atribut nama pada tabel mahasiswa.

2. Input data user dari keyboard

```
2. Vnama := '&nama_mhs';
```

Baris di atas meminta user melakukan input nilai. Hasil input tersebut akan di tampung pada variabel Vnama, perhatikan bahwa diperlukan penggunaan tanda petik.

3. Blok PL/SQL

```
3. update mahasiswa set nama='Abdul Azis' where nama=Vnama;
4. commit;
5.  if sql%found then
6.   dbms_output.put_line('Mahasiswa dengan nama '||vnama||'
     telah di update');
7.  else
8.   dbms_output.put_line('Tidak ada mahasiswa dengan nama :
     '||vnama);
9.   end if;
```

Baris ke-3 adalah cursor implisit yang akan melakukan pengubahan nilai atribut nama.

Baris ke-5 akan melakukan pengecekkan apakah proses update berhasil, jika berhasil akan tampil keterangan data mahasiswa telah di ubah dan jika tidak akan tampil keterangan gagal.

6.3.1.3 Solusi Lengkap

```
1. DECLARE
2.    Vnama mahasiswa.nama%type;
3.  BEGIN
4.    Vnama := '&nama_mhs';
5.    update mahasiswa set nama='Abdul Azis' where nama=Vnama;
6.    commit;
7.    if sql%found then
8.      dbms_output.put_line('Mahasiswa dengan nama '||vnama||'
        telah di update');
9.    else
10.     dbms_output.put_line('Tidak ada mahasiswa  dengan nama :
        '||vnama);
11.   end if;
12. EXCEPTION
13.   WHEN OTHERS THEN
14.     DBMS_OUTPUT.PUT_LINE(SQLERRM);
15. END;
16. /
```

6.3.1.4 Pengamatan

1. Tuliskan Output/tampilannya solusi lengkap 10.3.1.3 jika variabel substitusi nama_mhs diisikan **Bambang**!
2. Apa yang terjadi jika baris ke-6 dihilangkan?
3. Jelaskan maksud baris ke-7!
4. Pada kondisi apa Exception tersebut aktif?
5. Apa yang terjadi jika kondisi where pada baris ke-5 dihilangkan?

6.3.2 Menampilkan Jumlah Data dengan Implicit Cursor

Pada bagian ini akan dipelajari penggunaan Implicit Cursor untuk menghitung jumlah data pada suatu tabel. Diketahui isi tabel mahasiswa adalah sebagai berikut:

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

6.3.2.1 Soal

Buatlah blok anonim PL/SQL untuk jumlah data pada tabel mahasiswa dengan menggunakan Implicit Cursor!

6.3.2.2 Langkah Penyelesaian

1. Deklarasi blok anonim PL/SQL

```
1. Jumlah NUMBER;
```

Pada bagian deklarasi perlu dideklarasikan 1 buah variabel skalar. Variabel Jumlah memiliki tipe data NUMBER yang akan difungsikan untuk menampung jumlah data dari tabel mahasiswa.

2. Blok PL/SQL

```
2. SELECT COUNT(*) INTO jumlah FROM mahasiswa;
3. DBMS_OUTPUT.PUT_LINE('Jumlah Mahasiswa adalah '||jumlah);
```

Baris ke-2 akan menghitung jumlah seluruh record dari tabel mahasiswa dengan fungsi agregasi COUNT. Hasil perhitungan kemudian disimpan pada variabel jumlah.

Baris ke-3 akan menampilkan isi dari variabel jumlah yaitu sebanyak record yang ada pada tabel mahasiswa.

6.3.2.3 Solusi Lengkap

```
1. DECLARE
2.       Jumlah NUMBER;
3. BEGIN
4.       SELECT COUNT(*) INTO jumlah FROM mahasiswa;
5.       DBMS_OUTPUT.PUT_LINE     ('Jumlah     Mahasiswa     adalah
   '||Jumlah);
6. EXCEPTION
7.       WHEN OTHERS THEN
8.       DBMS_OUTPUT.PUT_LINE (SQLCODE||'-'||SQLERRM);
9.  END;
10. /
```

6.3.2.4 Pengamatan

1. Tuliskan output dari blok anonim PL/SQL pada solusi lengkap 10.3.2.3!
2. Jelaskan pengertian baris ke-4!
3. Tuliskan fungsi agregasi lain yang dapat digunakan beserta output hasilnya!
4. Kapankah exception aktif? Berikan contoh input yang menyebabkan exception aktif!

6.3.3 Menampilkan Informasi Mahasiswa berdasarkan Input Kelas

Pada bagian ini akan dipelajari penggunaan Implicit Cursor untuk menampilkan informasi mahasiswa dengan mencarinya berdasarkan input kelas. Jika didapatkan kelas yang mahasiswanya lebih dari satu maka exception akan aktif.

6.3.3.1 Soal

Diketahui tabel kelas sebagai berikut:

Kelas		
kdKelas	**NamaKelas**	**Kapasitas**
1	PIS-10-01	40
2	PCE-10-01	39
3	PCA-10-01	40

Diketahui tabel mahasiswa sebagai berikut, mahasiswa berelasi dengan kelas pada atribut kdkelas.

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	3
3031002	Beni Lemkari	Jl. Buahbatu 123	3
3011001	Angga Sunandar	Jl. Buahbatu 444	1
3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

Buatkan blok anonim PL/SQL untuk menampilkan nim dan nama mahasiswa jika diinputkan prodi dari keyboard. Jika terdapat prodi yang memiliki mahasiswa lebih dari satu maka akan ditampilkan jumlah mahasiwanya. Jika input kode prodi salah maka akan tampil pesan **Kode Prodi Salah!**.

6.3.3.2 Langkah Penyelesaian

1. Deklarasi variabel

```
1. jumlah NUMBER:=0;
2. vnim mahasiswa.nim%TYPE;
3. vnama mahasiswa.nama%TYPE;
4. vkode VARCHAR2(2);
5. vprodi NUMBER;
```

Variabel jumlah berfungsi untuk menampung nilai jumlah mahasiswa per prodi. Variabel vnim berfungsi untuk menampung nilai atribut nim dari tabel mahasiswa. Variabel vnama berfungsi untuk menampung nilai atribut nama dari tabel mahasiswa. Variabel vkode

berfungsi untuk menampung kode prodi dalam bentuk angka (1/2/3). Variabel vprodi berfungsi untuk menampung kode prodi dalam bentuk karakter (MI/TK/KA).

2. Blok struktur percabangan

```
1. vprodi := '&prodi';
2.  IF vprodi='MI' THEN
3.    vkode := 1;;
4.  ELSIF vprodi='TK' THEN
5.    vkode := 2;
6.  ELSE
7.    vkode := 3;
8.  END IF;
```

Baris ke-6 akan meminta input prodi berupa karakter (MI/TK/KA). Baris ke-7 s/d 13 merupakan struktur percabangan yang akan melakukan pengecekkan isi variabel Vprodi, jika MI maka variabel vkode akan bernilai 1, jika TK vkode akan bernilai 2 dan jika selain itu maka vkode akan bernilai 3.

3. Query mengambil nilai dari tabel mahasiswa

```
9.  SELECT COUNT(*) INTO jumlah FROM mahsiswa WHERE
    kdkelas=vkode;
10. SELECT nim,nama INTO vnim,vnama FROM mahasiswa WHERE
    kdkelas=vkode;
11. DBMS_OUTPUT.PUT_LINE(vnim||' - '||vnama);
```

Baris ke-14 akan menghitung jumlah mahasiswa dengan kode tertentu sesuai nilai variabel vkode.

Baris ke-15 akan mengisi variabel vnim dengan nilai atribut nim dan variabel vnama dengan atribut nama dari tabel mahasiswa dengan prodi=vkode.

Baris ke-16 akan menampilkan isi dari variabel vnim dan vnama.

4. Exception

```
12. EXCEPTION
13.     WHEN TOO_MANY_ROWS THEN
14.          DBMS_OUTPUT.PUT_LINE('Jumlah      mahasiswa      =
    '||jumlah);
15.     WHEN NO_DATA_FOUND THEN
16.          DBMS_OUTPUT.PUT_LINE('Kode Prodi Salah!');
```

Baris ke-17 s/d ke 21 akan melakukan error trapping.

6.3.3.3 Solusi Lengkap

```
1. DECLARE
2.      jumlah NUMBER:=0;
3.      vnim mahasiswa.nim%TYPE;
4.      vnama mahasiswa.nama%TYPE;
5.      vkode VARCHAR2(2);
6.      vprodi NUMBER;
```

```
7. BEGIN
8.       vprodi := '&prodi';
9.       IF vprodi='MI' THEN
10.           vkode := 1;;
11.      ELSIF vprodi='TK' THEN
12.           vkode := 2;
13.      ELSE
14.           vkode := 3;
15.      END IF;
16.      SELECT COUNT(*) INTO jumlah FROM mahsiswa WHERE
         kdkelas=vkode;
17.      SELECT nim,nama INTO vnim,vnama FROM mahasiswa WHERE
         kdkelas=vkode;
18.      DBMS_OUTPUT.PUT_LINE(vnim||' - '||vnama);
19. EXCEPTION
20.      WHEN TOO_MANY_ROWS THEN
21.           DBMS_OUTPUT.PUT_LINE('Jumlah      mahasiswa      =
   '||jumlah);
22.      WHEN NO_DATA_FOUND THEN
23.           DBMS_OUTPUT.PUT_LINE('Kode Prodi Salah!');
24. END;
25. /
```

6.3.3.4 Pengamatan

1. Tuliskan outputnya jika input prodi=**MI**!
2. Tuliskan outputnya jika input prodi=**TK**!
3. Tuliskan outputnya jika input prodi=**KA**!
4. Tuliskan outputnya jika input prodi=**IF**!
5. Tuliskan outputnya jika input prodi=**IF** dan baris ke-16 **ditukar** dengan baris ke-17!
6. Apakah Exception pada baris ke-20 mungkin untuk dieksekusi? Berikan alasannya!
7. Apakah Exception pada baris ke-22 mungkin di eksekusi? Berikan alasannya!

6.4 TEST AKHIR

Diketahui tabel mahasiswa dan nilai assessment sebagai berikut

Mahasiswa			
NIM	**Nama**	**Alamat**	**kdKelas**
3031001	Abdul Hakim	Jl. Banda 101	5
3031002	Beni Lemkari	Jl. Buahbatu 123	5
3011001	Angga Sunandar	Jl. Buahbatu 444	1

3011002	Brian Saragih	Jl. Buahbatu 444	1
3021001	Anisa Suhendar	Jl. Ambon 100, Jl. Buahbatu 12	2

Assessment				
kdMatkul	**NIM**	**kdDosen**	**tglAssessment**	**Nilai**
IS-143	3031001	BBY	10-Jan-11	70
IS-143	3031002	BBY	10-Jan-11	80
IS-143	3011001	BBY	12-Jan-11	90
IS-182	3011001	BBY	15-Jan-11	60
IS-182	3031002	BBY	15-Feb-11	85
IS-143	3011001	WHY	12-Feb-11	40
IS-143	3011002	WHY	12-Feb-11	30
CA-132	3021001	DAD	12-Jan-11	65
CA-132	3011001	DAD	14-Jul-11	55
CA-132	3031002	DAD	14-Jul-11	95

Selesaikan soal-soal berikut:

1. Buat blok non modular PL/SQL untuk menggunakan Implicit Cursor untuk menampilkan rata-rata nilai mata kuliah CA-132!

2. Buat blok anonim PL/SQL menggunakan Implicit Cursor untuk menampilkan nilai akhir yang didapatkan mahasiswa (A/B/C/D/E). NIM mahasiswa diinput dari keyboard. Nilai akhir ditentukan debagai berikut:
   A = 80-100
   B = 70-79
   C = 60-69
   D = 40-59
   E = 0-39

3. Tampilkan jumlah dosen yang melakukan assessment di bulan February dengan menggunakan Implicit Cursor!

6.5 RESUME

6.5.1 Jurnal Pengamatan

Selesaikan contoh kasus 1-3, kemudian jawablah pertanyaan-pertanyaan yang terdapat pada bagian pengamatan yang ada dibawah kode soal lengkapnya. Terdapat 16 soal pertanyaan disana, masing-masing pertanyaan memiliki komponen penilaian yang sama.

6.5.2 Lain-lain

1. Kumpulkan jawaban anda kepada para asisten sebelum praktikum diakhiri.
2. Media pengerjaan dan pengumpulan bebas, ikuti aturan yang telah ditetapkan oleh asisten praktikum.

7 BAB VII STORED PROCEDURE TANPA PARAMETER

7.1 IDENTITAS

Kajian

Stored Procedure

Topik

1. Konsep dan struktur strored procedure dengan parameter
2. Manajemen stored procedure (Create, Create Or Replace, Drop)

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu membuat dan mengeksekusi Stored Procedure dengan parameter.
2. Mampu menjalankan dan memanfaatkan stored procedure.

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Tugas Pendahuluan: 20%
2. Jurnal Hasil Pengamatan: 40%
3. Tugas Akhir: 40%

7.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Jelaskan karakteristik dan perbedaan antara anonymous block PL/SQL dan stored procedure!
2. Perhatikan sintaks pembuatan stored procedure dibawah ini

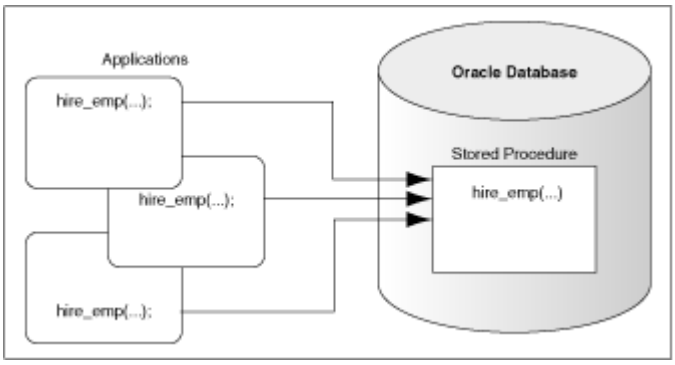
```
CREATE OR REPLACE PROCEDURE name
[(parameter[, parameter, ...])]
AS
[local declarations]
BEGIN
executable statements
[EXCEPTION
exception handlers]
END [name];
```

   Jelaskan fungsi dari bagian local declarations dan exceptions!
3. Jelaskan bagaimana cara untuk melihat stored procedure yang sudah dibuat!
4. Jelaskan perbedaan antara statement CREATE dan CREATE OR REPLACE!

7.3 PRAKTIK

7.3.1 Pengenalan Oracle Stored Procedure dan Schema HR

Stored Procedure merupakan modul program yang melakukan satu atau lebih aksi. Stored Procedure tidak membutuhkan nilai balikan. Stored procedure disimpan secara permanen dalam database. Ilustrasi dapat dilihat pada gambar berikut:



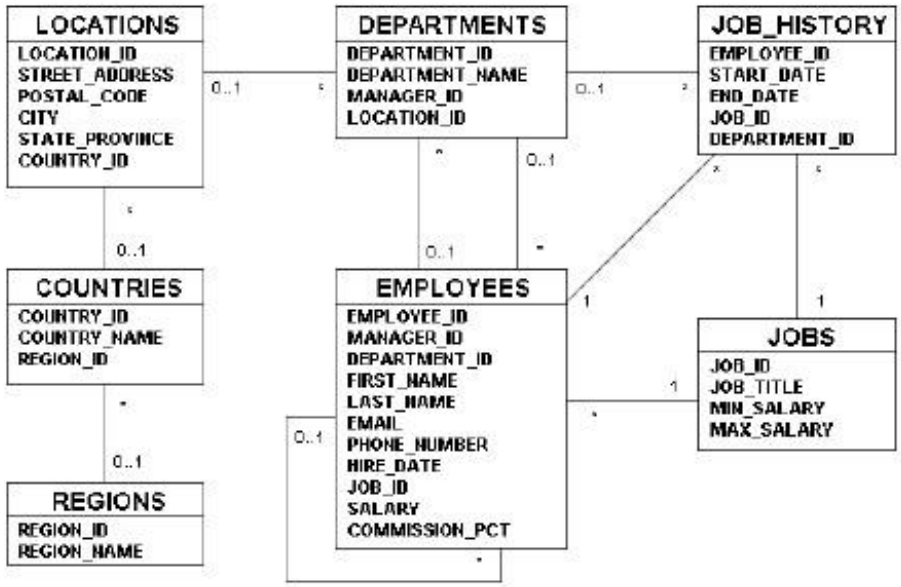
Sintaks untuk membuat procedure sebagai berikut:

```
CREATE OR REPLACE PROCEDURE name
[(parameter[, parameter, ...])]
AS
[local declarations]
BEGIN
executable statements
[EXCEPTION
exception handlers]
END [name];
```

Perintah untuk memanggil procedure adalah sebagai berikut:

1. Menggunakan anonymous block
   ```
   BEGIN
   Procedure_name;
   END;
   ```
2. Menggunakan perintah execute
   Exec[ute] procedure_name;

Human Resources (HR) Schema



Schema HR merupakan bagian dari schema contoh yang sudah ada pada saat pertama kali kita melakukan instalasi oracle. Pada praktikum ini dan seterusnya kita akan menggunakan schema HR untuk mengerjakan latihan soal. Adapun tabel-tabel yang ada pada schema HR adalah sebagai berikut:

REGIONS: merupakan tabel yang berisi data region/ wilayah seperti Americas, Asia, dan lain-lain.

COUNTRIES: merupakan tabel yang berisi data negara yang berhubungan dengan data wilayah.

LOCATIONS: merupakan alamat lengkap dari kantor, gudang, atau tempat produksi dari perusahaan di suatu negara.

DEPARTMENTS: merupakan tabel yang berisi data detail department tempat pegawai bekerja. Tabel ini juga berelasi dengan tabel employees dalam bentuk penempatan pegawai dan manajer dari department tersebut.

EMPLOYEES: merupakan tabel yang menyimpan data pegawai yang ada di perusahaan.

JOB : merupakan tabel yang menyimpan detail data jenis-jenis job/ posisi dari pegawai.

JOB_HISTORY: merupakan tabel yang menyimpan data riwayat posisi pegawai.

7.3.2 Stored Procedure Tanpa Parameter 1

Pada bagian ini dipelajari mengenai pembuatan stored procedure tanpa parameter sederhana

7.3.2.1 Soal

Buatlah stored procedure dengan nama HALO_DUNIA, stored procedure halo dunia akan menampilkan kalimat “halo dunia!!!” ketika dijalankan!

7.3.2.2 Langkah Penyelesaian

1. Buatlah stored procedure HALO_DUNIA menggunakan statement CREATE OR REPLACE

```
CREATE OR REPLACE PROCEDURE halo_dunia AS
```

2. Gunakan perintah DBMS_OUTPUT.PUT_LINE untuk menampilkan kalimat “halo dunia!!!”!

```
BEGIN
DBMS_OUTPUT.PUT_LINE('halo dunia!!!');
END;
/
```

3. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
4. Jalankan stored procedure HALO_DUNIA menggunakan anonymous block!

7.3.2.3 Pengamatan

1. Pastikan stored procedure tersebut sudah berhasil dibuat dengan melihat pada data dictionary sbb:

```
SELECT object_name, object_type, status
FROM user_objects
WHERE object_name = [nama stored procedure];

dan

SELECT text
FROM user_source
WHERE name = [nama stored procedure]
/
```

2. Jalankan stored procedure HALO_DUNIA menggunakan anonymous block!
3. Buatlah stored procedure lagi dengan nama HALO_DUNIA yang akan memunculkan teks “Halo duniaku yang indah!!!”, buat stored procedure tersebut menggunakan perintah CREATE, jelaskan apa yang terjadi dan berikan solusinya!
4. Jalankan DROP PROCEDURE [nama stored procedure] dan ulangi langkah nomor 1, jelaskan apa yang terjadi!

7.3.3 Stored Procedure Tanpa Parameter 2

Pada bagian ini dipelajari mengenai pembuatan stored procedure tanpa parameter yang memiliki variabel.

7.3.3.1 Soal

Buatlah stored procedure dengan nama ARITMATIKA. Stored procedure tersebut memiliki dua buah variabel yaitu “a” dan “b” yang bertipe number. Ketika dieksekusi stored procedure tersebut akan menampilkan hasil penjumlahan, pengurangan, perkalian, dan pembagian dari variabel “a” dan “b”!

7.3.3.2 Langkah Penyelesaian

1. Buatlah stored procedure ARITMATIKA menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE PROCEDURE aritmatika AS
```

2. Deklarasikan variabel “a” dan “b” pada bagian local declaration dengan tipe data number!

```
a NUMBER;
b NUMBER;
```

3. Inisialisasi nilai variabel “a” dan “b” dengan nilai 4 dan 5!

```
a:=4;
b:=5;
```

4. Gunakan perintah DBMS_OUTPUT.PUT_LINE untuk menampilkan operasi dan hasil penjumlahan, pengurangan, perkalian, dan pembagian!

```
DBMS_OUTPUT.PUT_LINE(a||'+'||b||'='||TO_CHAR(a+b));
DBMS_OUTPUT.PUT_LINE(a||'-'||b||'='||TO_CHAR(a-b));
DBMS_OUTPUT.PUT_LINE(a||'x'||b||'='||TO_CHAR(a*b));
DBMS_OUTPUT.PUT_LINE(a||'/'||b||'='||TO_CHAR(a/b));
```

5. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
6. Jalankan stored procedure ARITMATIKA menggunakan anonymous block!

7.3.3.3 Pengamatan

1. Apa yang terjadi jika function TO_CHAR dihilangkan?
2. Tambahkan variabel “c” yang berfungsi untuk menampung hasil operasi aritmatika!

7.3.4 Stored Procedure Tanpa Parameter 3

Pada bagian ini dipelajari mengenai pembuatan stored procedure yang menggunakan cursor implisit

7.3.4.1 Soal

Buatlah stored procedure yang bernama TOTAL_SALARY yang akan menampilkan jumlah total dari gaji di tabel employees!

7.3.4.2 Langkah Penyelesaian

1. Buatlah stored procedure TOTAL_SALARY menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE PROCEDURE total_salary AS
```

2. Deklarasikan variabel untuk menampung hasilnya!

```
jumlah NUMBER;
```

3. Buatlah cursor implisit untuk mengambil data dari tabel employees!

```
SELECT SUM(salary) INTO jumlah FROM employees;
```

4. Gunakan perintah DBMS_OUTPUT.PUT_LINE untuk menampilkan hasilnya!

```
DBMS_OUTPUT.PUT_LINE('Jumlah total gaji adalah '||jumlah);
```

5. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
6. Jalankan stored procedure TOTAL_SALARY menggunakan perintah execute!

7.3.4.3 Pengamatan

1. Supaya hasilnya lebih jelas coba tambahkan karakter “$” sebelum nilai total gaji dan pisahkan ribuan mengunakan koma (“,”) dan disertai dengan dua angka decimal di belakang dengan pemisah titik (“.”). sebagai contoh nilai 1000000 akan menjadi $1,000,000.00!
2. Tampilkan juga nilai rata-rata gaji pegawai pada stored procedure TOTAL_SALARY!

7.3.5 Stored Procedure Tanpa Parameter 4

Pada bagian ini dipelajari mengenai pembuatan stored procedure yang menggunakan cursor eksplisit

7.3.5.1 Soal

Buatlah stored procedure yang bernama VIEW_DEPARTMENT yang akan menampilkan semua data yang ada di tabel departments!

7.3.5.2 Langkah Penyelesaian

1. Buatlah stored procedure VIEW_DEPARTMENT menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE PROCEDURE view_department AS
```

2. Deklarasikan cursor eksplisit untuk mengambil data dari tabel departments!

```
CURSOR cur IS SELECT * FROM departments;
```

3. Buat perulangan untuk menampilkan data yang ada pada cursor!

```
FOR hasil IN cur LOOP
…
…
END LOOP;
```

4. Gunakan perintah DBMS_OUTPUT.PUT_LINE untuk menampilkan department_id, department_name, manager_id, dan location_id!

```
DBMS_OUTPUT.PUT_LINE('----------------------------------------
');
DBMS_OUTPUT.PUT_LINE('department_id: '||hasil.department_id);
DBMS_OUTPUT.PUT_LINE('department_name:
'||hasil.department_name); DBMS_OUTPUT.PUT_LINE('manager_id:
'||hasil.manager_id); DBMS_OUTPUT.PUT_LINE('location_id:
'||hasil.location_id); DBMS_OUTPUT.PUT_LINE('------------------
----------------------
');
```

5. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
6. Jalankan stored procedure VIEW_DEPARTMENT menggunakan perintah execute!

7.3.5.3 Pengamatan

1. Apa yang terjadi jika “`hasil.department_id`” diganti menjadi “`department_id`”? jelaskan!
2. Ubahlah cursor eksplisit menjadi cursor implisit, jelaskan apa yang terjadi!

7.4 TEST AKHIR

1. Tuliskan perintah untuk menghapus semua stored procedure yang telah anda buat sebelumnya!

2. Buatlah stored procedure tanpa parameter. Stored procedure ini memiliki dua buah variabel yaitu “x” dan “y”. inisialisasi isi kedua variabel tersebut dengan nilai tertentu kemudian tukar nilai “x” dan “y”. sebagai contoh nilai awal x=8 dan y=5. Diakhir proses nilai akan tertukar sehingga nilai x=5 dan y=8!
3. Buatlah stored procedure tanpa parameter untuk menghitung luas lingkaran dan keliling dengan jari-jari sebagai inputan (inputan berupa variabel)!
4. Buatlah stored procedure tanpa parameter untuk menampilkan output sbb:

   *
   **
   ***
   ****
5. Buatlah stored procedure tanpa parameter untuk menampilkan nama department dan nama manager-nya!
6. Buatlah stored procedure tanpa parameter untuk menampilkan department_name dan jumlah total pegawai yang ada pada setiap department!
7. Buatlah stored procedure tanpa parameter untuk menampilkan negara-negara yang ada di region “Asia” dan “Middle East and Africa”!
8. Buatlah stored procedure tanpa parameter untuk menampilkan nama pegawai beserta total salary-nya! (total salary dihitung dari salary ditambah dengan komisi)

8 BAB VIII STORED PROCEDURE DENGAN PARAMETER

8.1 IDENTITAS

Kajian

Stored Procedure

Topik

3. Konsep dan struktur strored procedure dengan parameter

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu menjelaskan Konsep dan struktur Stored Procedure dengan parameter
2. Mampu membuat dan mengeksekusi Stored Procedure dengan parameter.

Lama Kegiatan Praktikum

3. Pertemuan Terbimbing : 1 x 120 menit
4. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

3. Tugas Pendahuluan: 20%
4. Jurnal Hasil Pengamatan: 40%
5. Tugas Akhir: 40%

8.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Jelaskan perbedaan mode IN, OUT dan IN OUT pada saat pendefinisian parameter?
2. Apa yang kamu ketahui mengenai BIND Variable?
3. Jelaskan perbedaan antara parameter formal dan aktual?
4. Perhatikan contoh berikut,

```
CREATE OR REPLACE PROCEDURE TAMBAH(A NUMBER,B IN OUT NUMBER) IS
BEGIN
B := A+B;
END TAMBAH;
/
Bagaimana menampilkan hasil keluaran dengan memanfaatkan prosedur
TAMBAH diatas
(*hint : gunakan anonymous block)
```

5. Buat prosedur untuk mengecek apakah dua buah bilangan yang menjadi masukan pada prosedur tersebut bernilai sama (nilai balikan true) atau tidak (nilai balikan false)?
6. Dari prosedur berikut

```
CREATE OR REPLACE PROCEDURE BINTANG(a NUMBER) IS
i number :=1;
j number :=1;
BEGIN
FOR i in 1..a LOOP
    FOR j in 1..a LOOP
          IF i=ceil(a/2) or j=ceil(a/2) or i+j=a+1 or i=j THEN
          DBMS_OUTPUT.PUT('*');
          ELSE
          DBMS_OUTPUT.PUT('_');
          END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(' ');
END LOOP;
END BINTANG;
```

Jika prosedur tersebut diakses dengan EXEC BINTANG(9) bagaimana keluarannya?

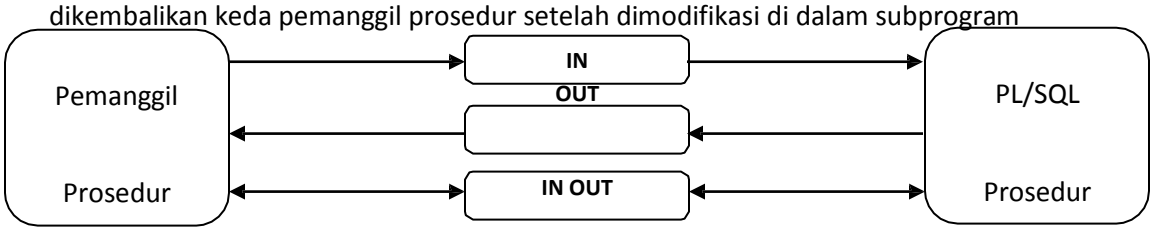
8.3 PRAKTIK

8.3.1 Pengenalan Parameter

Parameter disebut juga bentuk spesial dari lokal variabel, dideklarasikan setelah nama subprogram (prosedur) pada PL/SQL Header. Fungsi utamanya sebagai jembatan komunikasi antara pemanggil prosedur dengan subprogram dari prosedur itu sendiri. Penggunaan parameter hampir sama seperti variabel local akan tetapi bergantung pada **mode** dalam pendeklarasiannya, antara lain :

a. IN (*default jika mode tidak dituliskan*) menyediakan nilai yang bisa diproses didalam subprogram
b. OUT mengembalikan nilai pada pemanggil prosedur

c. IN OUT menyediakan nilai masukan dari pemanggil yang memungkinkan juga untuk dikembalikan keda pemanggil prosedur setelah dimodifikasi di dalam subprogram




```
CREATE OR REPLACE PROCEDURE name
    [(parameter1 [mode] datatype1,
      Parameter2 [mode] datatype2, ...)]
IS|AS
    [local declarations]
BEGIN
    executable statements
[EXCEPTION exception handlers]
END [name];
```

Formal Parameter merupakan parameter yang dideklarasikan pada isi (body) dari subprogram sebuah prosedur. Sedangkan **Actual Parameter merupakan p**arameter yang meneruskan informasi menuju prosedur jadi actual parameter terletak pada pemanggil prosedur.

Host variable atau disebut juga *bind* atau *global variable* merupakan variabel yang dideklarasikan diluar subprogram blok PL/SQL. Variabel tersebut dapat dipanggil/digunakan dalam anonymous block tetapi tidak didalam *stored subprogram*. Bind variable dideklarasikan dengan

VARIABLE nama_variabel tipe_data_variabel;

Pemanggilan Bind Variable menggunakan tanda ':' didepan nama_variabel, berikut salah satu cara pemanggilannya dengan perintah EXECUTE

EXECUTE :nama_variabel := nilai_variabel;

Menampilkan semua prosedur yang ada di dalam schema dengan cara

```
SELECT object_name
FROM user_objects
WHERE object_type = 'PROCEDURE';
```

Sedangkan menghapus sebuah prosedur yang ada di dalam schema dengan cara

DROP PROCEDURE nama_prosedur;

8.3.2 Stored Procedure Dengan Parameter 1

Pada bagian ini dipelajari mengenai pembuatan stored procedure dengan parameter untuk mode IN

8.3.2.1 Soal

Buatlah prosedur bernama PENURUNAN_GAJI yang digunakan untuk memodifikasi data pegawai dengan menurunkan gaji pegawai dengan nilai masukan ID pegawai dan berapa persen penurunan gajinya.

8.3.2.2 Langkah Penyelesaian

1. Buatlah stored procedure `PENURUNAN_GAJI` menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE PROCEDURE PENURUNAN_GAJI
```

2. Deklarasikan parameter "id_pegawai" dengan mode IN sebagai variabel penampung masukan untuk prosedur dengan tipe data sesuai dengan kolom employee_id pada tabel employees dan "persen" dengan mode IN sebagai variabel penampung presentase penurunan gaji

```
(id_pegawai IN EMPLOYEES.EMPLOYEE_ID%TYPE,
 persen IN NUMBER) IS
```

3. Pada bagian body dari Prosedur, buatlah query untuk memodifikasi gaji pegawai berdasarkan variabel masukan dari pemanggil prosedur

```
  BEGIN
      UPDATE EMPLOYEES
      SET salary := salary * (1-persen/100)
      WHERE employee_id = id_pegawai;
  END PENURUNAN_GAJI;
/
```

4. Jalankan stored procedure `PENURUNAN_GAJI` menggunakan perintah EXECUTE!

```
  EXECUTE PENURUNAN_GAJI(204,5);
```

8.3.2.3 Pengamatan

1. Apa yang terjadi jika perintah mode "IN" pada pendeklarasian parameter dihapus?
2. Jelaskan apa yang terjadi jika prosedur penurunan gaji dipanggil dengan menjalankan perintah `PENURUNAN_GAJI(334,5)`?

8.3.3 Stored Procedure Dengan Parameter 2

Pada bagian ini dipelajari mengenai pembuatan stored procedure dengan parameter untuk mode IN dan OUT

8.3.3.1 Soal

Buatlah prosedur bernama TERMASUK_BENUA yang digunakan untuk menampilkan nama benua (REGION) dari sebuah Negara (COUNTRY) yang menjadi masukan dari prosedur.

8.3.3.2 Langkah Penyelesaian

1. Buatlah stored procedure `TERMASUK_BENUA` menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE PROCEDURE TERMASUK_BENUA
```

2. Deklarasikan parameter "negara" dengan mode IN sebagai variabel penampung masukan untuk prosedur dan "benua" dengan mode OUT sebagai variabel penampung keluaran dari

prosedur dengan tipe data sesuai dengan kolom Benua (REGION) dan Negara (COUNTRY) pada masing masing tabel!

```
(negara IN COUNTRIES.COUNTRY_NAME%TYPE,
 benua OUT REGIONS.REGION_NAME%TYPE) IS
```

3. Pada bagian body dari Prosedur, buatlah query dari dua tabel yaitu `COUNTRIES dan REGIONS` untuk mencari nama benua berdasarkan variabel negara

```
  BEGIN
      select REGION_NAME INTO benua from COUNTRIES a, REGIONS b
      where a.REGION_ID=b.REGION_ID and COUNTRY_NAME=negara;
  END TERMASUK_BENUA;
/
```

4. Jalankan stored procedure `TERMASUK_BENUA` menggunakan anonymous block!

```
  DECLARE
      b REGIONS.REGION_NAME%TYPE;
  BEGIN
      TERMASUK_BENUA('India',b);
      DBMS_OUTPUT.PUT_LINE(b);
  END;
  /
```

8.3.3.3 Pengamatan

1. Apa yang terjadi jika pernyataan "`a.REGION_ID=b.REGION_ID`" pada query prosedur `TERMASUK_BENUA` dihilangkan ?
2. Gunakan variabel "a" pada anonymous block yang berfungsi untuk menampung nama Negara kemudian tentukan variabel mana saja yang termasuk dalam parameter actual maupun formal?

8.3.4 Stored Procedure Dengan Parameter 3

Pada bagian ini dipelajari mengenai pembuatan stored procedure dengan parameter untuk mode IN OUT

8.3.4.1 Soal

Buatlah prosedur untuk menghitung nilai faktorial dari sebuah bilangan masukan

8.3.4.2 Langkah Penyelesaian

1. Buatlah stored procedure FAKTORIAL menggunakan statement CREATE OR REPLACE

```
CREATE OR REPLACE PROCEDURE FAKTORIAL
```

2. Deklarasikan parameter masukan!

```
(bilangan IN OUT NUMBER) IS
```

3. Deklarasikan variabel lokal

```
    i number;
    hasil NUMBER:=1;
```

4. Buat algoritma faktorial !

```
BEGIN
FOR i in 1..bilangan LOOP
 hasil := hasil*i;
END LOOP;
Bilangan:=hasil;
END FAKTORIAL;
/
```

5. Gunakan anonymous block dan BIND VARIABLE untuk mengakses prosedur FAKTORIAL

```
VARIABLE BILANGAN_INPUT NUMBER;
BEGIN
:BILANGAN_INPUT:=10;
FAKTORIAL(:BILANGAN_INPUT);
DBMS_OUTPUT.PUT_LINE(:BILANGAN_INPUT);
END;
/
```

8.3.4.3 Pengamatan

1. Bisakah prosedur FAKTORIAL diatas, diakses menggunakan perintah EXECUTE? Jelaskan!
2. Jika BILANGAN_INPUT bernilai minus, tentukan keluaran dari nilai faktorialnya? Jelaskan mengapa demikian?

8.4 TEST AKHIR

1. Tuliskan perintah untuk menampilkan semua prosedur yang telah anda buat sebelumnya!
2. Buatlah prosedur yang bisa mengeluarkan nilai selisih antara dua buah bilangan masukan sebagai contoh SELISIH(8,5,a) dan nilai a adalah 3!
3. Buatlah prosedur untuk menampilkan nama pegawai (FIRST_NAME , LAST_NAME) dan gaji (SALARY) dimana gaji dari pegawai tersebut lebih besar dari :BILANGAN_INPUT (BIND VARIABLE) dari tabel pegawai (EMPLOYEES)
   (*hint : gunakan kursor eksplisit)
4. Buatlah stored procedure dengan mode parameter IN untuk menampilkan output sbb jika nilai masukan=5!

```
  *
  *
*****
  *
  *
```

5. Buatlah prosedur untuk mengkonversi 3 digit pertama dan menambahkan tanda kurung dari sebuah no telpon misal 02134567 yang menjadi parameter masukan menjadi (021-34567)
   (*hint : memecah karakter gunakan fungsi SUBSTR)
6. Buatlah prosedur untuk menentukan dari kedua bilangan A dan B pada parameter masukan, manakah bilangan yang lebih besar A atau B!

7. Buatlah prosedur GENAP untuk menampilkan bilangan positif genap yang kurang dari nilai :BILANGAN_INPUT!
   Misal :BILANGAN_INPUT:=10;
   Maka keluaran dari prosedur Genap setelah dieksekusi oleh anonymous block adalah 2 4 6 8
8. Buatlah prosedur dengan parameter masukan EMPLOYEE_ID untuk menampilkan nama pegawai (FIRST_NAME) beserta total salary-nya! (total salary dihitung dari salary ditambah dengan persentase komisi/commission_pct dari salary) Jika commission_pct kosong maka total salary adalah tetap salary itu sendiri.
   Input : 176
   Output : total salary = 8600+(0.2*8600)=10320

   Input : 181
   Output : total salary = 3100

9 BAB IX STORED FUNCTION TANPA PARAMETER

9.1 IDENTITAS

Kajian

Stored Function

Topik

1. Konsep dan struktur strored function tanpa parameter
2. Manajemen parameter pada stored procedure
3. Manajemen stored function (Create, Create or Replace, Drop)

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu membuat dan mengeksekusi Stored Function tanpa parameter.
2. Mampu menjalankan dan memanfaatkan Stored Function.

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Tugas Pendahuluan: 20%
2. Jurnal Hasil Pengamatan: 40%
3. Tugas Akhir: 40%

9.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Jelaskan perbedaan antara stored procedure dan stored function?
2. Apa perintah untuk menampilkan semua fungsi yang ada pada users object?
3. Buat FUNCTION untuk menghitung jumlah bilangan genap antara 1 sampai 11?
4. Perhatikan contoh berikut,

```
CREATE OR REPLACE FUNCTION TAMBAH RETURN number2 IS
BEGIN
B := A+B;
RETURN B;
END TAMBAH;
/
```

   Tentukan kesalahan-kesalahan pada function tanpa parameter TAMBAH yang ada dan perbaiki sehingga dapat memberikan keluaran hasil dari penjumlahan antara dua buah bilangan '123' dan '456'?

9.3 PRAKTIK

9.3.1 Pengenalan Oracle Stored Function

Stored Function tanpa parameter merupakan block PL/SQL yang tanpa menerima masukan parameter tetapi bisa menjalankan aksi tertentu dan mengembalikan nilai. Stored function disimpan secara permanen dalam database. Struktur dari procedure dan function hampir sama, akan tetapi setiap function harus mengembalikan nilai (paling tidak mempunyai sebuah pernyataan RETURN) pada block yang memanggil function tersebut. Sintaks untuk membuat function sebagai berikut:

```
CREATE OR REPLACE FUNCTION name
RETURN datatype
IS|AS
  [local declarations]
BEGIN
  executable statements
RETURN expressions;
[EXCEPTION
  exception handlers]
END [name];
```

Cara mengakses stored function ada 3, yaitu :

a) Anonymous Block PL/SQL
b) Perintah EXECUTE
c) Penggunaan di dalam SQL statement

Penggunaan function untuk SQL statement mempunyai batasan atau limitasi yang tidak bisa dilanggar diantaranya :

a) Pernyataan *Select* tidak dapat mengandung DML (Data Manipulation Language)
b) Pernyataan *Update* atau *Delete* pada sebuah tabel 'A' tidak dapat menjalankan query atau update pada tabel yang sama 'A'

c) Pernyataan SQL tidak dapat mengakhiri transaksi (tidak dapat mengesekusi COMMIT ataupun ROLLBACK)

Menampilkan semua Function yang ada di dalam schema dengan cara

```
SELECT object_name
FROM user_objects
WHERE object_type = 'FUNCTION';
```

Menghapus Stored Function dengan perintah

```
DROP FUNCTION function_name
```

9.3.2 Stored Function Tanpa Parameter 1

Pada bagian ini dipelajari mengenai pembuatan stored function tanpa parameter sederhana

9.3.2.1 Soal

Buatlah fungsi untuk menampilkan daftar bilangan genap antara 1 sampai 10.

9.3.2.2 Langkah Penyelesaian

1. Buatlah stored function `BILANGAN_GENAP_DIBAWAH_10` menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE FUNCTION BILANGAN_GENAP_DIBAWAH_10
```

2. Definisikan tipe data balikan dari stored function

```
RETURN VARCHAR2 IS
```

3. Pada bagian body dari function, buatlah perulangan dari 1 sampai 10 dan mengecek tiap angkanya apakah termasuk dalam bilangan genap atau tidak

```
      I NUMBER;
      temp VARCHAR2(50);
  BEGIN
      FOR i in 1..10 LOOP
          IF (MOD(i,2)=0) THEN
          temp := temp ||' '||i;
          END IF;
      END LOOP;
      RETURN temp;
  END BILANGAN_GENAP_DIBAWAH_10;
/
```

4. Jalankan stored function `BILANGAN_GENAP_DIBAWAH_10` menggunakan perintah EXECUTE!

```
  EXECUTE DBMS_OUTPUT.PUT_LINE(BILANGAN_GENAP_DIBAWAH_10);
```

9.3.2.3 Pengamatan

1. Jelaskan apa kegunaan dari `MOD(i,2)` dan apa kaitannya dengan bilangan genap?
2. Jelaskan apa yang terjadi jika pada saat pendeklarasian variabel lokal `VARCHAR2(50)` diubah menjadi `VARCHAR2`? mengapa hal itu bisa terjadi ? Jelaskan apa maksud dari dari angka 50?

9.3.3 Stored Function Tanpa Parameter 2

Pada bagian ini dipelajari mengenai pembuatan stored function tanpa parameter dengan cursor eksplisit

9.3.3.1 Soal

Buatlah stored function bernama VIEW_EMPLOYEES untuk menampilkan daftar nama pegawai.

9.3.3.2 Langkah Penyelesaian

1. Buatlah stored function VIEW_EMPLOYEES menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE FUNCTION VIEW_EMPLOYEES return varchar2 AS
```

2. Deklarasikan cursor eksplisit untuk mengambil data dari tabel employees!

```
CURSOR cur IS SELECT * FROM employees;
hasil varchar2(5000);
temp varchar2(5000);
i number;
```

3. Buat perulangan untuk menampilkan data yang ada pada cursor!

```
Begin
    i:=1;
    FOR hasil IN cur LOOP
        temp := temp||chr(13)||
        chr(10)||i||'. '||hasil.first_name||' '||hasil.last_name;
        i:=i+1;
    END LOOP;
    Return temp;
End VIEW_EMPLOYEES;
```

4. Gunakan perintah anonymous block PL/SQL untuk menampilkan output dari function VIEW_EMPLOYEES!

```
BEGIN
DBMS_OUTPUT.PUT_LINE('-----------------------------------------
');
DBMS_OUTPUT.PUT_LINE('Daftar Nama Pegawai: ');
DBMS_OUTPUT.PUT_LINE('-----------------------------------------
');
DBMS_OUTPUT.PUT_LINE(VIEW_EMPLOYEES);
DBMS_OUTPUT.PUT_LINE('-----------------------------------------
');
END;
```

9.3.3.3 Pengamatan

1. Jelaskan apa kegunaan dari variabel hasil?
2. Jelaskan fungsi dari chr(10) dan chr(13)? Apa yang terjadi jika statement tersebut dihilangkan?
3. Modifikasi fungsi diatas, tampilkan nama depan, email dan gaji(salary) dari tiap pegawai?

9.3.4 Stored Function Tanpa Parameter 3

Pada bagian ini dipelajari mengenai bagaimana mengakses stored function tanpa parameter di dalam query

9.3.4.1 Soal

Buatlah stored function `MIN_KENAIKAN_GAJI` untuk menghitung kenaikan gaji yang wajar untuk tiap pegawai yaitu 10% dari rata-rata gaji seluruh pegawai. Kemudian gunakan fungsi tersebut untuk menunjukkan perbedaan antara gaji saat ini dengan gaji yang diharapkan untuk tiap-tiap pegawai.

9.3.4.2 Langkah Penyelesaian

1. Buatlah stored function `MIN_KENAIKAN_GAJI` menggunakan statement CREATE OR REPLACE!

```
CREATE OR REPLACE FUNCTION MIN_KENAIKAN_GAJI RETURN number AS
```

2. Deklarasikan variabel lokal untuk menampung data rata-rata gaji seluruh pegawai dari tabel EMPLOYEES!

```
avgsal number;
```

3. Buat fungsi untuk menghitung 10% dari rata-rata gaji seluruh pegawai!

```
BEGIN
    SELECT FLOOR(avg(salary)) INTO avgsal FROM employees;
    RETURN (CEIL(avgsal*0.1));
END MIN_KENAIKAN_GAJI;
```

4. Gunakan QUERY select untuk menampilkan perbandingan gaji saat ini dengan gaji yang yangdiharapkan dengan memanfaatkan fungsi `MIN_KENAIKAN_GAJI`!

```
SELECT first_name||' '||last_name  NAME, salary,
salary+MIN_KENAIKAN_GAJI  EXPECTED_SALARY
FROM employees;
```

9.3.4.3 Pengamatan

1. Jelaskan apa yang terjadi jika query berikut dijalankan?

```
UPDATE employees
   SET salary = salary + MIN_KENAIKAN_GAJI;
```

   Mengapa hal tersebut bisa terjadi?

2. Jelaskan perbedaan dari fungsi CEIL dan FLOOR pada fungsi diatas?
3. Apa yang terjadi jika pada QUERY no.4 pernyataan '`EXPECTED_SALARY`' dihapus? Apa kegunaan dari pernyataan '`EXPECTED_SALARY`' dan '`NAME`' ?

9.4 TEST AKHIR

1. Tuliskan perintah untuk menampilkan semua FUNGSI yang telah anda buat sebelumnya!
2. Buatlah fungsi untuk menampilkan bilangan kelipatan 7 dengan nilai antara 1 hingga 50!
3. Buatlah fungsi untuk menampilkan nama NEGARA (dari tabel COUNTRIES) yang termasuk dalam REGIONS ASIA dan diakhiri dengan huruf 'a'!
   (*hint : gunakan fungsi substr)

Contoh output :
Australia
China
India
Malaysia

4. Buat fungsi untuk menampilkan deret aritmatika dengan selisih 3 untuk nilai antara 10 dan 100 Contoh output :
10+13+16+19+22+25+28+31+34+37+40+43+46+49+52+55+58+61+64+67+70+73+76+79+82+85+88+91+94+97+100

5. Buatlah fungsi untuk menampilkan nama-nama kota terurut menurun (descending) pada lokasi-lokasi departemen (database HR) yang berada di benua EROPA (region_name = 'EUROPE')?
Contoh Output :
Veniche
Utrecht
Stretford
Roma
Oxford
Munich
London
Geneva
Bern

6. Perhatikan Stored Function berikut,

```
Create or replace function PLUSKU return varchar2 is
i number;
j number;
begin
for i in 1..10 loop
for j in 1..10 loop
IF (i=1) or (j=1) or (i=10) or (j=10) THEN
        DBMS_OUTPUT.PUT('*');
        ELSE
        DBMS_OUTPUT.PUT('_');
        END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE(' ');
end loop;
end;
```

Jelaskan dan tuliskan apa keluaran dari function ‘PLUSKU’?

7. Buatlah fungsi untuk menampilkan jumlah huruf k dari kata “KUKU KAKI KAKEKKU KAKU KAKU”!
8. Jelaskan manfaat dan kegunaan dari FUNCTION tanpa parameter?

10 BAB X STORED FUNCTION DENGAN PARAMETER

10.1 IDENTITAS

Kajian

Stored Function dengan parameter

Topik

1. Konsep dan struktur strored function dengan parameter
2. Manajemen stored function

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu menjelaskan Konsep dan struktur Stored Function.
2. Mampu membuat dan mengeksekusi Stored Function dengan parameter.
3. Mampu melakukan manajemen Store Function (Create, Create or Replace, Drop)

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Tugas Pendahuluan: 20%
2. Jurnal Hasil Pengamatan: 40%
3. Tugas Akhir: 40%

10.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Jelaskan mengapa tidak dapat menjalankan *SELECT Statement* yang mengandung DML pada stored function?
2. Apa perintah untuk menghapus sebuah function?
3. Buat Stored Function untuk menambah jabatan pada tabel JOBS dengan masukan sbb
   JOB_ID = IT_MAN
   JOB_TITLE = IT Manager
   MIN_SALARY = 9000
   MAX_SALARY = 12000
   Stored Function tersebut akan mengembalikan status 'berhasil' jika berhasil memasukkan nilai tersebut kedalam tabel JOBS!

4. Perhatikan function berikut,

```
CREATE OR REPLACE FUNCTION SISA_BAGI(A number, B number) RETURN
number IS
BEGIN
RETURN(MOD(A,B));
END SISA_BAGI;
/
```

   Tuliskan dalam dua cara yang berbeda bagaimana mengakses function SISA_BAGI diatas dan tunjukkan apa outputnya?

10.3 PRAKTIK

10.3.1 Pengenalan Oracle Stored Function

Stored Function dengan parameter merupakan block PL/SQL yang bisa menerima masukan parameter, menjalankan aksi tertentu dan mengembalikan nilai. Stored function disimpan secara permanen dalam database. Struktur dari procedure dan function dengan parameter hampir sama, akan tetapi setiap function harus mengembalikan nilai (paling tidak mempunyai sebuah pernyataan RETURN) pada block yang memanggil function tersebut. Sintaks untuk membuat function dengan parameter sebagai berikut:

```
CREATE OR REPLACE FUNCTION name
  [(parameter1 [mode1]  datatype1
   (parameter2 [mode2]  datatype2, ...)]
RETURN datatype
IS|AS
  [local declarations]
BEGIN
  executable statements
RETURN expressions;
[EXCEPTION
  exception handlers]
END [name];
```

Tipe-tipe mode yang ada adalah IN, OUT, dan IN OUT. Meskipun mode OUT dan IN OUT dapat digunakan pada pembuatan sebuah function akan tetapi bukan sebuah cara yang bagus karena pada function bisa menggunakan RETURN untuk mengembalikan nilai. Mode IN adalah default pada saat pembuatan sebuah function.

Cara mengakses stored function pada ekspresi SQL dapat diletakkan pada :

a) Pernyataan SELECT atau pada klausa dari sebuah query
b) Pernyataan kondisi WHERE atau HAVING
c) Klausa pada query yaitu pada CONNECT BY, START WITH, ORDER BY, dan GROUP BY
d) Pernyataan INSERT pada saat mengisi VALUES
e) Pernyataan UPDATE pada saat mengisi SET

10.3.2 Stored Function Dengan Parameter 1

Pada bagian ini dipelajari mengenai pembuatan stored function dengan parameter sederhana

10.3.2.1 Soal

Buatlah fungsi untuk menampilkan nama pegawai dengan masukan EMPLOYEE_ID.

10.3.2.2 Langkah Penyelesaian

1. Buatlah stored function `NAMA_PEGAWAI` menggunakan statement CREATE OR REPLACE dan parameter masukan ID pegawai!

```
CREATE OR REPLACE FUNCTION NAMA_PEGAWAI (id
employees.employee_id%type)
```

2. Gunakan VARCHAR2 sebagai tipe data untuk nilai balikan dalam menampilkan nama pegawai!

```
RETURN VARCHAR2 IS
```

3. Pada bagian deklarasi dan body dari function, buatlah pernyataan SQL untuk menampilkan nama pegawai dari id yang diketahui

```
      a VARCHAR2(100);
      temp VARCHAR2(50);
  BEGIN
      SELECT first_name||' '||last_name into a from employees
      where employee_id = id;
      RETURN a;
  END NAMA_PEGAWAI;
/
```

4. Jalankan stored function `NAMA_PEGAWAI` menggunakan perintah EXECUTE!

```
  EXECUTE DBMS_OUTPUT.PUT_LINE(NAMA_PEGAWAI (204));
```

10.3.2.3 Pengamatan

1. Jelaskan apa kegunaan dari variabel `a?` apa yang terjadi jika dihapus?
2. Jelaskan apa kegunaan %type? Ubah %type menjadi tipe data lain yang sesuai sehingga function diatas tetap bisa dijalankan!
3. Jalankan fungsi diatas dengan anonymous block dengan bantuan inputan dari bind variabel !

10.3.3 Stored Function Dengan Parameter 2

Pada bagian ini dipelajari mengenai pembuatan stored function dengan parameter dan memanggilnya menggunakan SQL *statement*

10.3.3.1 Soal

Buatlah fungsi RATA2_GAJI_JABATAN untuk menghitung rata-rata gaji untuk sebuah jabatan dari pegawai dengan masukan EMPLOYEE_ID. Kemudian jalankan perintah SQL untuk menampilkan nama, id pegawai, dan gaji pegawai yang lebih tinggi dari rata-rata gaji untuk sebuah jabatan menggunakan fungsi RATA2_GAJI_JABATAN.

10.3.3.2 Langkah Penyelesaian

1. Buatlah stored function `RATA2_GAJI_JABATAN` menggunakan statement CREATE OR REPLACE dan parameter masukan ID pegawai!

```
CREATE OR REPLACE FUNCTION RATA2_GAJI_JABATAN (id
employees.employee_id%type)
```

2. Gunakan NUMBER sebagai tipe data untuk nilai balikan dalam menampilkan nilai tengah (rata-rata) dari gaji pegawai!

```
RETURN NUMBER IS
```

3. Pada bagian deklarasi dan body dari function, buatlah pernyataan SQL untuk menampilkan rata-rata gaji dari sebuah jabatan pegawai yang id pegawainya diketahui

```
    temp NUMBER;
BEGIN
    SELECT (MIN_SALARY+MAX_SALARY)/2 into temp from employees
    a, jobs b where a.job_id=b.job_id and a.employee_id = id;
    RETURN temp;
END RATA2_GAJI_JABATAN;
/
```

4. Jalankan stored function `RATA2_GAJI_JABATAN` menggunakan SQL statement!

```
Select first_name, salary, RATA2_GAJI_JABATAN(employee_id)
RATA2_GAJI_JABATAN from employees where salary >=
RATA2_GAJI_JABATAN(employee_id);
```

10.3.3.3 Pengamatan

1. Jelaskan apa yang terjadi jika pernyataan yang digarisbawahi berikut dihapus?

```
Select first_name, salary, RATA2_GAJI_JABATAN(employee_id)
RATA2_GAJI_JABATAN from employees where salary >=
RATA2_GAJI_JABATAN(employee_id);
```

2. Jalankan perintah SQL untuk menampilkan nama pegawai berserta gajinya dimana gaji tersebut lebih kecil dibandingkan gaji rata-rata jabatan(memanfaatkan fungsi `RATA2_GAJI_JABATAN`)!
3. Jika tanda buka dan tutup kurung pada fungsi `RATA2_GAJI_JABATAN` dihapus,

```
SELECT (MIN_SALARY+MAX_SALARY)/2 into temp from employees a,
jobs b where a.job_id=b.job_id and a.employee_id = id
```

Apakah berpengaruh terhadap hasil akhir fungsi tersebut?

10.3.4 Stored Function Dengan Parameter 3

Pada bagian ini dipelajari mengenai pembuatan stored function dengan parameter dan memanggilnya menggunakan Anonymous Block PL/SQL

10.3.4.1 Soal

Buatlah fungsi LINGKARAN untuk menghitung penjumlahan antara luas dan keliling dari Lingkaran.

10.3.4.2 Langkah Penyelesaian

1. Buatlah stored function LINGKARAN menggunakan statement CREATE OR REPLACE dan parameter masukan jari jari!

```
CREATE OR REPLACE FUNCTION LINGKARAN (jari number)
```

2. Gunakan number sebagai tipe data untuk nilai balikan dalam menampilkan luas dan keliling dari Lingkaran!

```
RETURN NUMBER IS
```

3. Pada bagian deklarasi dan body dari function, buatlah pernyataan SQL untuk menampilkan rata-rata gaji dari sebuah jabatan pegawai yang id pegawainya diketahui

```
      temp_luas NUMBER;
      temp_kell NUMBER;
  BEGIN
      temp_luas := 22/7*jari*jari;
      temp_kell := 2*22/7*jari;
      return(temp_luas+temp_kell);
  END LINGKARAN;
/
```

4. Deklarasikan Bind Variable untuk membentuk variabel jari-jari

```
Variable jari2 number;
```

5. Jalankan stored function LINGKARAN menggunakan anonymous block PL/SQL dan bind variabel jari jari!

```
  Declare
  temp number;
  Begin
  :jari2 := 7;
  temp := LINGKARAN(:jari2);
  DBMS_OUTPUT.PUT_LINE('Lingkaran dengan jari jari '|| :jari2
  ||' memliki jumlah luas dan keliling sebesar '|| temp);
  End;
```

10.3.4.3 Pengamatan

1. Hapus Function LINGKARAN yang telah dibuat, kemudian buat kembali fungsi LINGKARAN tanpa menggunakan bantuan variabel lokal (temp_kell maupun temp_luas)?
2. Gunakan perintah EXECUTE dalam menggunakan fungsi LINGKARAN!
3. Buat kembali anonymous block PL/SQL untuk memanfaatkan fungsi LINGKARAN tanpa menggunakan bantuan variabel lokal (temp)? {tanpa ada declare}

10.4 TEST AKHIR

1. Tuliskan perintah untuk menampilkan JUMLAH semua function dan procedure yang telah anda buat sebelumnya!
2. Buatlah function yang mengembalikan nilai sesuai dengan persamaan matematika $f(x) = x^3 + x^2 + 1$, jika inputan dari function tersebut adalah x!
3. Buatlah function dengan mode parameter IN untuk menampilkan output sbb jika nilai masukan=5!

```
*____
 _*___
_ ____
*****
   ___*
___ _ _
    ___*
_____
```

4. Buatlah Function untuk menghitung nilai faktorial dari sebuah bilangan yang menjadi inputan

   Input : 5

   Output : 120

   5x4x3x2x1 = 120

   Input : 6

   Output : 720

   6x5x4x3x2x1 = 720
5. Buatlah stored function untuk menghitung pajak dari gaji yang diterima pegawai yaitu sebesar 2% dengan nilai masukan EMPLOYEE_ID. Kemuadian buat Anonymous block yang memanfaatkan function tersebut untuk menampilkan nama, id dan gaji bersih setelah dkurangi pajak dari seluruh pegawai! {sesuai dengan database HR}
6. Buatlah function untuk menghitung konversi dari rupiah (sebagai nilai masukan) menjadi dollar (sebagai nilai keluaran, pembulatan kebawah). Asumsi 1 dollar = Rp 12.000,-

   Input : 12500

   Output : 1 dollar

   Input : 50000

   Output : 4 dollar
7. Apakah bisa menggunakan bind variable di dalam body sebuah stored function? Jelaskan!
8. Apa kelebihan dari block PL/SQL yang dibuat jika didalamnya menggunakan stored function?

11 BAB XI PENGENALAN PACKAGE

11.1 IDENTITAS

Kajian

Package

Topik

1. Konsep dan struktur package.
2. Komponen Public dan Private pada Package.
3. Membangun package specification dan package body.
4. Manajemen Package.

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu menjelaskan Konsep dan struktur Package.
2. Mampu membangun package.

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Tugas Pendahuluan: 20%
2. Jurnal Hasil Pengamatan: 40%
3. Tugas Akhir: 40%

11.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Jelaskan perbedaan antara stored procedure/ function dengan package!
2. Jelaskan apa yang dimaksud dengan package specification dan package body, jelaskan juga hubungan antara dua komponen tersebut!
3. Jelaskan tentang public dan private component!
4. Perhatikan contoh package specification berikut:

```
CREATE OR REPLACE PACKAGE comm_pkg IS
std_comm NUMBER := 0.10; --initialized to 0.10
PROCEDURE reset_comm(new_comm NUMBER);
END comm_pkg;
/
```

   Apa yang terjadi jika procedure reset_comm tidak ada pada package body!

5. Perhatikan contoh package body berikut:

```
CREATE OR REPLACE PACKAGE BODY comm_pkg IS
FUNCTION validate(comm NUMBER) RETURN BOOLEAN IS
max_comm employees.commission_pct%type;
BEGIN
SELECT MAX(commission_pct) INTO max_comm
FROM employees;
RETURN (comm BETWEEN 0.0 AND max_comm);
END validate;
PROCEDURE reset_comm (new_comm NUMBER) IS BEGIN
IF validate(new_comm) THEN
std_comm := new_comm; -- reset public var
ELSE RAISE_APPLICATION_ERROR(
-20210, 'Bad Commission');
END IF;
END reset_comm;
END comm_pkg;
```

   Apa yang terjadi jika function validate tidak ada pada package specification?

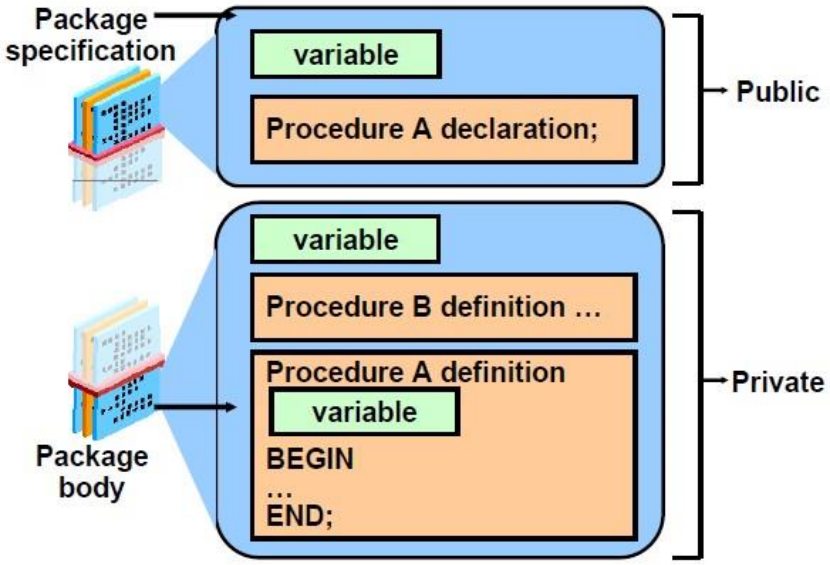
11.3 PRAKTIK

11.3.1 Pengenalan Oracle Package

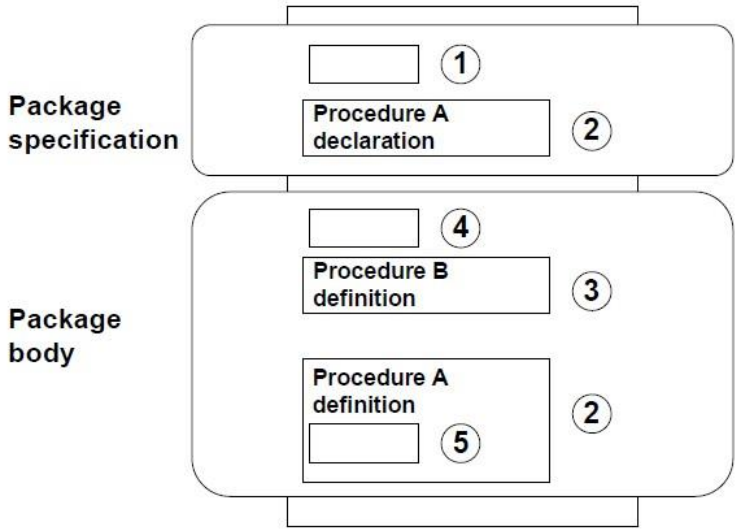
Package merupakan pengelompokan dari script PL/SQL yang di dalamnya dapat berupa procedure, function, cursor, deklarasi variabel, dan tipe data. Dengan menggunakan package kita mendapatkan beberapa keuntungan. Keuntungan yang pertama adalah source code yang kita buat akan lebih rapi karena telah terkelompok ke dalam satu paket. Kedua adalah ketika kita memiliki beberapa procedure dan function yang sering kita panggil dengan mengelompokkannya dalam suatu package maka performansi sistem akan meningkat karena isi package akan diload hanya pada saat pertama kali ke dalam memory. Setelah itu isi dari package tidak perlu diload lagi dari disk sehingga bottleneck akan dapat dihindari. PL/SQL Package memiliki 2 komponen penyusun utama, yaitu :

1. Package Specification

2. Package Body



Package Specification bersifat public, maksudnya adalah ketika package diakses dari luar bagian inilah sebenarnya yang melakukan interaksi bukan bagian Package Body. Mari kita telaah komponen Package lebih mendalam.



Keterangan Gambar :

1. Public (Global) Variabel
2. Public Procedure
3. Private Procedure
4. Private Variable
5. Local Variable

Dari deskripsi gambar di atas diketahui bahwa yang bersifat public adalah no 1 & 2 pada bagian specification. Public disini maksudnya ada ketika package dieksekusi di lingkungan SQL*Plus atau PL/SQL lain bagian inilah yang melakukan interaksi. Package Specification akan meneruskan permintaan dari luar ke Package Body, dari Package Body hasil akan ditampilkan ke layar atau dikembalikan lagi ke Package Specification untuk pemrosesan selanjutnya.

Sintaks untuk membuat package specification sebagai berikut:

```
CREATE [OR REPLACE] PACKAGE package_name IS|AS
public type and variable declarations
subprogram specifications
END [package_name];
```

Sintaks untuk membuat package body sebagai berikut:

```
CREATE [OR REPLACE] PACKAGE BODY package_name IS|AS
private type and variable declarations
subprogram bodies
[BEGIN initialization statements]
END [package_name];
```

11.3.2 Package 1

Pada bagian ini dipelajari mengenai pembuatan package sederhana

11.3.2.1 Soal

Buatlah package bernama PKG_ARITMATIKA yang berisi 4 function yaitu function untuk penambahan, pengurangan, perkalian, dan pembagian. Masing-masing function tadi memiliki dua buah parameter, nilai balikan dari function adalah hasil dari perhitungan dua buah parameter tsb!

11.3.2.2 Langkah Penyelesaian

1. Buatlah package specification untuk mendefinisikan masing-masing function!

```
CREATE OR REPLACE PACKAGE PKG_ARITMATIKA IS
FUNCTION penambahan(a NUMBER, b NUMBER) RETURN NUMBER;
…
…
…
…
END pkg_aritmatika;
/
```

2. Buatlah package body untuk mendefinisikan isi masing-masing function!

```
CREATE OR REPLACE PACKAGE BODY PKG_ARITMATIKA IS
FUNCTION penambahan(a NUMBER, b NUMBER) RETURN NUMBER IS
BEGIN
RETURN (a+b);
END penambahan;
```

```
…
END pkg_aritmatika;
```

3. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
4. Jalankan function-function yang ada pada package tersebut dengan menggunakan anonymous block!

```
BEGIN
DBMS_OUTPUT.PUT_LINE(pkg_aritmatika.penambahan(56,89));
…
…
END;
/
```

11.3.2.3 Pengamatan

1. Apa yang terjadi jika nama procedure atau function berbeda antara yang ada di package specification dan package body?
2. Apa yang terjadi jika function pembagian tidak kita buat pada package body?
3. Tuliskan perintah untuk melihat isi dari package specification!
4. Tuliskan perintah untuk melihat isi dari package body!

11.3.3 Package 2

Pada bagian ini dipelajari mengenai pembuatan package untuk mengelola data pada tabel departments

11.3.3.1 Soal

Buatlah package specification dan package body bernama PKG_DEPARTMENT. Di dalam PKG_DEPARTMENT terdapat:

1. procedure ADD_DEPARTMENT, merupakan procedure untuk menambah data department
2. procedure UPD_DEPARTMENT, merupakan procedure untuk mengubah data department
3. procedure DEL_DEPARTMENT, merupakan procedure untuk menghapus data department
4. function GET_DEPARTMENT, merupakan function untuk mengambil data department

11.3.3.2 Langkah Penyelesaian

1. Buatlah package specification untuk mendefinisikan masing-masing function!

```
CREATE OR REPLACE PACKAGE PKG_DEPARTMENT IS
PROCEDURE del_department(dep_id departments.department_id%type);
…
…
…
…
END pkg_department;
/
```

2. Buatlah package body untuk mendefinisikan isi masing-masing procedure dan function!

```
CREATE OR REPLACE PACKAGE BODY PKG_DEPARTMENT IS
```

```
…
…
PROCEDURE  del_department(dep_id  departments.department_id%type)
IS
BEGIN
…
DELETE departments WHERE department_id=dep_id;
…
END del_department;
…
…
END pkg_department;
/
```

3. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
4. Jalankan function-function yang ada pada package tersebut dengan menggunakan anonymous block!

```
BEGIN
pkg_department.del_department(10);
END;
/
```

11.3.3.3 Pengamatan

1. Apa yang terjadi jika tipe data parameter dep_id diganti dengan NUMBER?
2. Apa yang terjadi jika ketika kita jalankan pkg_department package body-nya belum dibuat?
3. Hapuslah package menggunakan perintah DROP PACKAGE BODY, jelaskan pada yang terjadi!
4. Hapuslah package menggunakan perintah DROP PACKAGE, jelaskan pada yang terjadi!

11.3.4 Package 3

Pada bagian ini dipelajari mengenai pembuatan package yang memiliki local & global variabel.

11.3.4.1 Soal

Buatlah sebuah package bernama PKG_VOLUME. Di dalam package ini terdapat dua buah function sebagai berikut:

1. Function VOLUME_BALOK, function ini akan mengembalikan nilai volume balok. Function VOLUME_BALOK memiliki dua parameter yaitu panjang dan lebar.
2. Function VOLUME_TABUNG, function ini akan mengembalikan nilai volume tabung. Function VOLUME_BALOK memiliki satu parameter yaitu jari_jari serta memiliki satu local variabel yaitu phi yang bernilai 3,14.

Di dalam PKG_VOLUME ini terdapat satu global variabel yaitu tinggi. Gunakan parameter dan variabel yang ada tsb untuk menghitung volume balok dan tabung ketika function dalam package ini dipangggil!

11.3.4.2 Langkah Penyelesaian

1. Buatlah package specification untuk mendefinisikan masing-masing function!

```
CREATE OR REPLACE PACKAGE PKG_VOLUME IS
```

```
tinggi NUMBER := 9;
FUNCTION volume_balok(panjang NUMBER, lebar NUMBER) RETURN
NUMBER;
FUNCTION volume_tabung(jari_jari NUMBER) RETURN NUMBER;
END pkg_volume;
/
```

2. Buatlah package body untuk mendefinisikan isi masing-masing procedure dan function!

```
CREATE OR REPLACE PACKAGE BODY PKG_VOLUME IS
…
…
…
END pkg_volume;
```

3. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
4. Jalankan function-function yang ada pada package tersebut dengan menggunakan anonymous block!

```
begin
dbms_output.put_line(pkg_volume.volume_balok(5,6));
dbms_output.put_line(pkg_volume.volume_tabung(10));
end;
/
```

11.3.4.3 Pengamatan

1. Jalankan perintah dibawah ini untuk melihat nilai dari global variabel tinggi!

```
BEGIN
DBMS_OUTPUT.PUT_LINE(pkg_volume.tinggi);
END;
/
```

Apakah nilai global variabel tinggi tersebut juga dapat diubah menggunakan anonymous block? jelaskan! Jika bisa dirubah tuliskan perintah untuk merubah nilai dari global variabel tinggi tersebut menjadi 15!

2. Jalankan perintah dibawah ini untuk melihat nilai dari variabel phi!

```
BEGIN
DBMS_OUTPUT.PUT_LINE(pkg_volume.volume_tabung.tinggi);
END;
/
```

Apa yang terjadi? Jelaskan!

3. Apakah nilai variabel phi pada function volume_tabung tersebut dapat dirubah? jelaskan! Jika bisa dirubah tuliskan perintah untuk merubah nilai dari variabel phi tersebut menjadi 22/7!

11.4 TEST AKHIR

1. Buatlah package bernama PKG_LUAS_BANGUN. Package ini berisi empat buah function sebagai berikut:
   a. Function LUAS_PERSEGI, mengembalikan luas dari persegi
   b. Function LUAS_SEGITIGA, mengembalikan luas dari segitiga.
   c. Function LUAS_LINGKARAN, mengembalikan luas lingkaran.
   d. Function LUAS_BELAHKETUPAT, mengembalikan luas belah ketupat.
2. Buatlah package bernama PKG_EMPJOB yang berisi procedure dan function sebagai berikut:
   a. Procedure NEW_JOB, berfungsi untuk menambahkan data baru di tabel JOBS
   b. Procedure UPD_JOBSAL, berfungsi untuk merubah gaji minimum dan maksimum di tabel jobs.
   c. Function GET_YEARS_SERVICE, berfungsi untuk mengembalikan lama bekerja dari seorang pegawai.
   d. Function GET_JOB_COUNT, berfungsi untuk mengembalikan jumlah job yang pernah dijalani oleh seorang pegawai dari awal hingga saat ini (data diambil dari tabel employees dan job_history).

12 BAB XII PACKAGE LANJUT

12.1 IDENTITAS

Kajian

Package

Topik

1. Overloading Package,
2. Package Initialization Block
3. Persistent State

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu menjelaskan dan mendemokan konsep overloading subprogram
2. Mampu menjelaskan dan mendemokan cara membangun initialization block.
3. Mampu menjelaskan dan mendemokan persistent state package.

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Tugas Pendahuluan: 20%
2. Jurnal Hasil Pengamatan: 40%
3. Tugas Akhir: 40%

12.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Jelaskan apa yang dimaksud dengan konsep overloading subprogram pada package! berikan contoh!
2. Jelaskan apa yang dimaksud dengan initialization block pada package! Berikan contoh!
3. Jelaskan apa yang dimaksud dengan persistent state package! Berikan contoh!

12.3 PRAKTIK

12.3.1 Konsep Package Lanjut

Dalam Oracle PL/SQL dimungkinkan untuk membuat dua atau lebih subprogram dengan nama yang sama. Fitur ini disebut dengan overloading. Fitur ini akan sangat berguna ketika kita membuat subprogram yang dapat menerima beberapa jenis tipe data. Sebagai contoh fungsi bawaan TO_CHAR. Fungsi ini dapat menerima beberapa jenis tipe data yang berbeda seperti NUMBER dan DATE. Beberapa pertimbangan dalam melakukan overloading:

- Membuat dua atau lebih subprogram dengan nama yang sama namun memiliki jenis atau jumlah parameter yang berbeda.
- Membuat alternatif dalam melakukan pencarian data dengan beberapa kriteria pencarian. Sebagai contoh ketika kita akan membuat mekanisme pencarian data pegawai berdasarkan NIP atau berdasarkan nama. Secara lojik pencarian kurang lebih akan sama namun memiliki parameter yang berbeda.
- Melakukan penambahan fungsionalitas tanpa merubah source code yang sudah ada.

Catatatan: subprogram yang berdiri sendiri tidak dapat dilakukan overloading.

Jika kita menggunakan package maka kita dapat melakukan inisialisasi terhadap variabel pada saat pertama kali kita memanggil package tersebut. Di dalam package body terdapat sebuah blok yang dipergunakan untuk melakukan inisialisasi variabel yang disebut sebagai initialization block:

```
CREATE OR REPLACE PACKAGE BODY package_name {IS | AS}
...
BEGIN
initialization_code;
END [package_name];
```

Secara default masing-masing session memiliki nilai variabel yang berbeda-beda, hal inilah yang disebut sebagai persistent state of package.

12.3.2 Overloading Subprogram

Pada bagian ini akan dipraktekkan tentang overloading subprogram pada package

12.3.2.1 Soal

Modifikasi package PKG_DEPARTMENT yang sudah anda buat sebelumnya sebagai berikut dengan menambahkan beberapa procedure dan function sbb:

a. Lakukan overloading untuk procedure GET_DEPARTMENT yang memiliki parameter dep_name sehingga kondisi yang digunakan untuk mengambil data di tabel departments berdasarkan department_name.

b. Lakukan overloading untuk procedure GET_DEPARTMENT yang memiliki parameter man_id dan loc_id sehingga kondisi yang digunakan untuk mengambil data di tabel departments berdasarkan manager_id dan location_id.
c. Lakukan overloading untuk procedure DEL_DEPARTMENT yang memiliki parameter dep_name sehingga kondisi yang digunakan untuk menghapus data berdasarkan pada department_name.
d. Lakukan overloading untuk procedure DEL_DEPARTMENT yang memiliki parameter loc_id sehingga kondisi yang digunakan untuk menghapus data berdasarkan pada location_id.

12.3.2.2 Langkah Penyelesaian

1. Buatlah package specification untuk mendefinisikan function GET_DEPARTMENT dan DEL_DEPARTMENT yang akan dilakukan overload!

```
CREATE OR REPLACE PACKAGE PKG_DEPARTMENT IS
…
…
FUNCTION   get_department(dep_id   departments.department_id%type)
RETURN departments%ROWTYPE;
FUNCTION                                    get_department(dep_name
departments.department_name%type) RETURN departments%ROWTYPE;
FUNCTION    get_department(man_id    departments.manager_id%type,
loc_id departments.location_id%type) RETURN departments%ROWTYPE;
PROCEDURE del_department(dep_id departments.department_id%type);
PROCEDURE                                   del_department(dep_name
departments.department_name%type);
END pkg_department;
/
```

2. Buatlah package body untuk mendefinisikan isi tiga function get_department dan dua function del_department!

```
CREATE OR REPLACE PACKAGE BODY PKG_DEPARTMENT IS
…
…
FUNCTION   get_department(dep_id   departments.department_id%type)
RETURN departments%ROWTYPE IS
CURSOR cur IS SELECT * FROM departments WHERE department_id =
dep_id;
dep departments%ROWTYPE;
BEGIN
OPEN cur;
FETCH cur INTO dep;
CLOSE cur;
RETURN dep;
END get_department;
```

```
FUNCTION                                        get_department(dep_name
departments.department_name%type) RETURN departments%ROWTYPE IS
CURSOR cur IS SELECT * FROM departments WHERE department_name =
dep_name;
dep departments%ROWTYPE;
BEGIN
OPEN cur;
FETCH cur INTO dep;
CLOSE cur;
RETURN dep;
END get_department;

FUNCTION    get_department(man_id    departments.manager_id%type,
loc_id  departments.location_id%type)  RETURN  departments%ROWTYPE
IS
CURSOR cur IS SELECT * FROM departments WHERE manager_id=man_id
AND location_id=loc_id;
dep departments%ROWTYPE;
BEGIN
OPEN cur;
FETCH cur INTO dep;
CLOSE cur;
RETURN dep;
END get_department;
…
…
END pkg_department;
/
```

3. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
4. Jalankan function-function yang ada pada package tersebut dengan menggunakan anonymous block!

```
BEGIN
DBMS_OUTPUT.PUT_LINE(pkg_department.get_department(20).department
_name);
DBMS_OUTPUT.PUT_LINE(pkg_department.get_department('Marketing').m
anager_id);
DBMS_OUTPUT.PUT_LINE(pkg_department.get_department(205,1700).depa
rtment_name);
END;
/
```

12.3.2.3 Pengamatan

1. Jika misalkan ditambahkan lagi function GET_DEPARTMENT dengan parameter loc_id dengan tipe sama seperti kolom location_id. Apa yang terjadi?jelaskan!
2. Jika misalkan ditambahkan lagi function GET_DEPARTMENT berparameter dep_id dengan tipe data CHAR. Apa yang terjadi?jelaskan!
3. Jika misalkan ditambahkan lagi function GET_DEPARTMENT berparameter dep_id dengan nilai balikan VARCHAR2. Apa yang terjadi?jelaskan!

12.3.3 Initialization Block

Pada bagian ini akan dipraktekkan tentang initialization block pada package.

12.3.3.1 Soal

Buatlah tabel yang bernama RATES yang memiliki tiga kolom yaitu CURRENCY, PERIOD, dan RATE. Tabel ini akan menyimpan nilai kurs mata uang terhadap dollar amerika (USD). Kemudian isikan data ke tabel RATES dengan nilai kolom CURRENCY='IDR', PERIOD=tanggal, dan RATE='12380'.
Tugas anda adalah membuat package bernama PKG_SALARY yang di dalamnya berisi sbb:

1. Global variabel bernama "kurs" yang bertipe NUMBER untuk menyimpan kurs mata uang terhadap dollar amerika.
2. Function CONVERT, fungsi ini akan mengembalikan nilai hasil konversi sebuah mata uang terhadap dollar amerika. Function CONVERT memiliki satu parameter input yaitu val_currency. Function CONVERT ini akan melakukan konversi berdasarkan nilai variabel "kurs" yang dikalikan dengan parameter val_currency.
3. Procedure PRINT_SALARY, prosedur ini akan mencetak daftar gaji seluruh pegawai dalam nilai rupiah dengan memanggil function CONVERT.

Di dalam package PKG_SALARY tersebut anda harus membuat initialization block. Pada initialization block ini dilakukan inisialisasi variabel "kurs" dengan mengambil nilai RATE pada tabel RATES yang berlaku.

12.3.3.2 Langkah Penyelesaian

1. Buatlah DDL untuk tabel RATES

```
CREATE TABLE RATES
   (   CURRENCY VARCHAR2(10),
       PERIOD DATE,
       RATE NUMBER,
        CONSTRAINT PK_RATES PRIMARY KEY (CURRENCY, PERIOD) ENABLE
   ) ;
```

2. Isikan data berikut ke tabel RATES

```
INSERT INTO RATES VALUES('IDR', '21-JAN-14', '12860');
```

3. Buat package specification

```
CREATE OR REPLACE PACKAGE PKG_SALARY IS
kurs NUMBER:=0;
FUNCTION CONVERT(val_currency NUMBER) RETURN NUMBER;
PROCEDURE PRINT_SALARY;
END PKG_SALARY
```

```
/
```

4. Buat package body yang terdapat initialization block di dalamnya

```
CREATE OR REPLACE PACKAGE BODY PKG_SALARY IS
FUNCTION CONVERT(val_currency NUMBER) RETURN NUMBER IS
BEGIN
RETURN(kurs*val_currency);
END CONVERT;

PROCEDURE PRINT_SALARY IS
CURSOR cur IS SELECT employee_id, first_name||' '||last_name as
name, CONVERT(salary) as sal FROM employees;
BEGIN
DBMS_OUTPUT.PUT_LINE('DAFTAR GAJI PEGAWAI DALAM RUPIAH');
…
…
…
END PRINT_SALARY;

BEGIN
SELECT rate INTO kurs FROM rates;
END PKG_SALARY;
/
```

5. Jalankan perintah SET SERVEROUT ON untuk memastikan hasil muncul pada saat dieksekusi!
6. Jalankan function dan procedure yang ada pada package tersebut menggunakan anonymous block!

```
BEGIN
pkg_salary.print_salary;
END;
/
```

12.3.3.3 Pengamatan

1. Jika kurs adalah private/local variable kemudian kita melakukan inisialisasi menggunakan initialization block kira-kira apa yang terjadi!

12.3.4 Persistent Stage

Pada bagian ini akan dipraktekkan tentang persistent stage pada package.

12.3.4.1 Soal

Dalam soal ini masih menggunakan PKG_SALARY pada soal sebelumnya. Untuk mempraktekkan persistent stage pada package bukalah dua session yang berbeda untuk mengakses schema HR dengan membuka dua SQL*Plus (loginlah pada masing-masing SQL*Plus menggunakan akun HR). Dengan menggunakan pkg_salary anda diminta untuk mensimulasikan bagaimana persistent stage dalam package!

12.3.4.2 Langkah Penyelesaian

Jalankanlah langkah-langkah berikut pada masing-masing session

	Session Pertama	Nilai Variabel Kurs	Session Kedua	Nilai Variabel Kurs

1	BEGIN
	DBMS_OUTPUT.PUT_LINE(pkg_salary.convert(5));
	END;
		12860	BEGIN
			DBMS_OUTPUT.PUT_LINE(pkg_salary.convert(5));
			END;
				12860

2		12860	UPDATE rates SET rate='13000' WHERE currency='IDR'	12860

3		12860	BEGIN
			DBMS_OUTPUT.PUT_LINE(pkg_salary.convert(5));
			END;
				13000

4	UPDATE rates SET rate='12500' WHERE currency='IDR'	12860	COMMIT;	13000

5	BEGIN
	DBMS OUTPUT.PUT LINE(pkg salary.convert(5));
	END;
		12500		13000

6	ROLLBACK;	12500		13000

7	BEGIN
	DBMS_OUTPUT.PUT_LINE(pkg_salary.convert(5));
	END;
		13000		13000

1. Pada session yang pertama dan kedua melakukan pemanggilan function CONVERT sehingga nilai variabel kurs terinisialisasi sesuai isi nilai dalam tabel RATES yaitu 12860.
2. Pada session kedua dilakukan update nilai kolom RATE menjadi 13000 namun nilai varibel kurs masih 12860.
3. Session kedua melakukan pemanggilan fungsi CONVERT sehingga nilai varibel kurs berubah menjadi 13000 namun nilai kurs pada session pertama masih 12860 karena perubahan varibel kurs bersifat persisten (hanya berlaku pada masing-masing session).
4. Meskipun session yang kedua melakukan commit nilai kurs pada session pertama masih tetap 12680. Pada saat yang hampir bersamaan session pertama menjalankan perintah update nilai kolom RATE menjadi 12500.
5. Session pertama melakukan pemanggilan fungsi CONVERT lagi sehingga nilai varibel kurs berubah sesuai isi kolom RATE yaitu 12500.
6. Session pertama membatalkan perubahan dengan menjalankan perintah ROLLBACK.
7. Session pertama memanggil fungsi CONVERT sehingga nilai variabel kurs berubah sesuai isi RATE terakhir yang telah di-COMMIT oleh session kedua yaitu 13000.

12.3.4.3 Pengamatan

1. Apa yang terjadi jika dijalankan perintah COMMIT pada session pertama setelah perintah UPDATE?

12.4 TEST AKHIR

1. Lihat lagi PKG_VOLUME yang sudah anda buat pada praktikum sebelumnya. Tugas anda adalah melakukan **overloading function** VOLUME_BALOK dan VOLUME_TABUNG dengan menambahkan parameter tinggi pada kedua function tersebut!
2. Tambahkan global variabel cur_date dan cur_currency dalam package PKG_SALARY. Kemudian inisialisasi cur_date menggunakan tanggal saat ini dan cur_currency dengan

nilai 'IDR'. Lakukan modifikasi pada cursor `SELECT rate INTO kurs FROM rates` dengan **mengkondisikan nilai rate sesuai dengan tanggal pada cur_date dan currency sesuai dengan cur_currency**!

3. Perhatikan skenario pada soal ketiga. Tugas anda adalah **melakukan modifikasi kecil** package specification PKG_SALARY supaya perubahan variabel kurs berlaku untuk session pertama dan kedua (nilai varibel kurs berlaku sama pada masing-masing session)!

13 BAB XIII UTL_FILE

13.1 IDENTITAS

Kajian

UTL_FILE

Topik

1. UTL_FILE
2. REGEXP_SUBSTR

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu menjelaskan dan mendemokan penggunaan UTL_FILR
2. Mampu mendemokan cara membangun program yang menggunakan UTL_FILE
3. Mampu menjelaskan dan mendemokan penggunaan REGEXP_SUBSTR
4. Mampu mendemokan cara membangun program yang menggunakan UTL_FILE dan REGEXP_SUBSTR

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Tugas Pendahuluan: 20%
2. Jurnal Hasil Pengamatan: 40%
3. Tugas Akhir: 40%

14 BAB XIV TRIGGER

14.1 IDENTITAS

Kajian

Trigger

Topik

3. Statement Level Trigger
4. Row Level Trigger
5. Manajemen Trigger

Referensi

5. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
6. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
7. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
8. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

5. Mampu menjelaskan dan mendemokan statement dan row level trigger
6. Mampu mendemokan cara membangun statement dan row level trigger
7. Mampu menjelaskan dan mendemokan manajemen trigger (enable, disable, remove)
8. Mampu mendemokan cara pengujian trigger

Lama Kegiatan Praktikum

3. Pertemuan Terbimbing : 1 x 120 menit
4. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

4. Tugas Pendahuluan: 20%
5. Jurnal Hasil Pengamatan: 40%
6. Tugas Akhir: 40%

14.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Sebutkan dan jelaskan jenis-jenis trigger!
2. Jelaskan manfaat dan kegunaan trigger!
3. Sebutkan batasan dari trigger!

14.3 PRAKTIK

14.3.1 Konsep Trigger Dasar

Trigger merupakan jenis Blok PL/SQL yang berhubungan/ berasosiasi dengan table, view, schema, atau database. Trigger dieksekusi secara implisit ketika terjadi suatu event. Adapun jenis-jenis trigger sebagai berikut:

- Application trigger: dibangkitkan ketika terjadi event pada aplikasi (Oracle Form Developer)
- Database Trigger: dibangkitkan ketika terjadi event pada data (ex: DML) atau event pada system (ex: logon, shutdown, dll) pada schema database.

Pada praktikum ini kita akan fokus pada DML Trigger. DML trigger merupakan jenis trigger yang dieksekusi ketika terjadi operasi DML (insert, update, delete). DML trigger dibagi menjadi dua sbb:

- Statement Trigger
  - Type trigger secara default
  - Dieksekusi sekali ketika terjadi trigger event
  - Akan selalu dieksekusi walaupun tidak ada row data yang berubah
- Row Trigger
  - Dieksekusi sebanyak row data yang berubah
  - Tidak akan dieksekusi jika tidak ada row data yang berubah
  - Ditunjukkan oleh statement FOR EACH ROW

14.3.2 Statement Trigger 1

Pada bagian ini akan dipraktekkan tentang statement trigger sederhana

14.3.2.1 Soal

Buatlah statement trigger bernama TRG_DML_REGIONS_WARNING yang mengoutputkan warning jika terjadi operasi DML pada tabel regions. Misalkan ketika terjadi penambahan di tabel regions maka akan dikeluarkan peringatan “Terjadi Penambahan data di tabel REGIONS!!!” ataupun jika terjadi perubahan data maka akan dikeluarkan peringatan “Terjadi Perubahan data di tabel REGIONS!!!”

14.3.2.2 Langkah Penyelesaian

1. Definisinikan nama trigger yang akan dibuat!

```
CREATE OR REPLACE TRIGGER TRG_DML_REGIONS_WARNING
```

2. Tentukan trigger timing dan event-nya

```
AFTER INSERT OR UPDATE OR DELETE ON REGIONS
```

3. Deklarasikan variabel yang dibutuhkan

```
DECLARE
```

```
warning VARCHAR2(100);
```

4. Tuliskan isi/ body dari trigger

```
BEGIN
IF INSERTING THEN
      warning:='Terjadi penambahan data di tabel regions!';
ELSIF UPDATING THEN
      warning:='Terjadi perubahan data di tabel regions!';
ELSIF DELETING THEN
      warning:='Terjadi penghapusan data di tabel regions!';
END IF;
DBMS_OUTPUT.PUT_LINE(warning);
END;
/
```

14.3.2.3 Pengamatan

1. Lakukan operasi DML yang merubah satu data di tabel regions, misalkan `UPDATE regions SET region_name='Indonesia' WHERE region_id=4;` kira-kira berapa jumlah warning yang akan ditampilkan? Jelaskan alasannya!
2. Lakukan operasi DML yang merubah lebih dari 1 data, misalkan `UPDATE regions SET region_name='Indonesia';` kira-kira berapa jumlah warning yang akan ditampilkan? Jelaskan alasannya!
3. Lakukan penghapusan data di tabel regions lalu buat kembali trigger tersebut dengan terlebih dahulu mengubah timing-nya menjadi BEFORE. Kemudian lakukan lagi penghapusan data di tabel regions dan jelaskan perbedaannya!

14.3.3 Statement Trigger 2

Pada bagian ini akan dipraktekkan tentang statement trigger yang lebih kompleks

14.3.3.1 Soal

Buatlah trigger bernama TRG_RESTRICT_SALARY_UPDATE untuk mencegah perubahan data gaji/ salary di tabel employees pada saat hari dan jam kerja. Asumsikan hari kerja adalah senin s.d jumat dan jam kerja 08.00 s/d 17.00.

14.3.3.2 Langkah Penyelesaian

1. Definisinikan nama trigger yang akan dibuat!

```
CREATE OR REPLACE TRIGGER TRG_RESTRICT_SALARY_UPDATE
```

2. Tentukan trigger timing dan event-nya

```
…… UPDATE OF …… ON EMPLOYEES
```

3. Tuliskan isi/ body dari trigger

```
BEGIN
IF (TO_CHAR(SYSDATE,'DY') IN ('MON','TUE','WED','THU','FRI')) AND
(TO_CHAR(SYSDATE,'HH24') BETWEEN '08' AND '17') THEN
      RAISE_APPLICATION_ERROR(-20504,'Tidak boleh melakukan
perubahan gaji pegawai pada hari dan jam kerja!!!');
```

```
END IF;
END;
/
```

14.3.3.3 Pengamatan

1. Lakukan perubahan data salary di tabel employees pada saat jam kerja. Jelaskan apa yang terjadi!
2. Lakukan juga perubahan data salary secara masal pada saat jam kerja. Kira-kira berapa jumlah warning yang ditampilkan dan jelaskan juga alasannya?
3. Rubahkan hari dan jam pada komputer anda sehingga seolah-olah waktu di komputer anda menunjukkan di luar hari atau jam kerja kemudian lakukan perubahan data salary di tabel employees. Jelaskan apa yang terjadi!

14.3.4 Row Trigger 1

Pada bagian ini akan dipraktekkan tentang row trigger sederhana

14.3.4.1 Soal

Modifikasi statement trigger TRG_DML_REGIONS_WARNING pada soal sebelumnya sehingga menjadi row trigger. Tampilkan juga detail perubahan datanya (data lama dan baru) dengan memanfaatkan referensi (OLD dan NEW)!

14.3.4.2 Langkah Penyelesaian

1. Definisinikan nama trigger yang akan dibuat!

```
CREATE OR REPLACE TRIGGER TRG_DML_REGIONS_WARNING
```

2. Tentukan trigger timing dan event-nya

```
AFTER INSERT OR UPDATE OR DELETE ON REGIONS
```

3. Tambahkan perintah FOR EACH ROW setelah trigger timing dan event

```
FOR EACH ROW
```

4. Deklarasikan variabel yang dibutuhkan

```
DECLARE
warning VARCHAR2(100);
```

5. Tuliskan isi/ body dari trigger

```
BEGIN
IF INSERTING THEN
      warning:='Terjadi penambahan data di tabel regions, dengan
nilai region_id='||:NEW.region_id||' dan
region_name='||:NEW.region_name;
ELSIF UPDATING THEN
      warning:='Terjadi perubahan data di tabel regions, dengan
nilai lama region_id='||:OLD.region_id||',
region_name='||:OLD.region_name||' nilai baru
region_id='||:NEW.region_id||', region_name='||:NEW.region_name;
```

```
ELSIF DELETING THEN
      warning:='Terjadi penghapusan data di tabel regions untuk
region_id='||:OLD.region_id||' dan region_name='||:OLD.region_name;
END IF;
DBMS_OUTPUT.PUT_LINE(warning);
END;
/
```

14.3.4.3 Pengamatan

1. Lakukan operasi DML yang merubah satu data di tabel regions, misalkan `UPDATE regions SET region_name='Indonesia' WHERE region_id=4;` kira-kira berapa jumlah warning yang akan ditampilkan? Jelaskan alasannya!
2. Lakukan operasi DML yang merubah lebih dari 1 data, misalkan `UPDATE regions SET region_name='Indonesia';` kira-kira berapa jumlah warning yang akan ditampilkan? Jelaskan alasannya!
3. Lakukan penghapusan data di tabel regions lalu buat kembali trigger tersebut dengan terlebih dahulu mengubah timing-nya menjadi BEFORE. Kemudian lakukan lagi penghapusan data di tabel regions dan jelaskan perbedaannya!

14.3.5 Row Trigger 2

Pada bagian ini akan dipraktekkan tentang row trigger yang lebih kompleks

14.3.5.1 Soal

Buatlah trigger bernama TRG_CHECK_SAL_RANGE yang akan dijalankan sebelum dilakukan perubahan pada kolom min_salary dan max_salary pada tabel jobs untuk setiap baris. Jika pada saat dilakukan update nilai pada kolom tersebut ternyata nilainya keluar pada range gaji di tabel employees maka trigger akan membangkitkan exception!

14.3.5.2 Langkah Penyelesaian

1. Definisinikan nama trigger yang akan dibuat!

```
CREATE OR REPLACE TRIGGER TRG_CHECK_SAL_RANGE
```

2. Tentukan trigger timing dan event-nya

```
BEFORE UPDATE OF min_salary, max_salary ON jobs
```

3. Tambahkan perintah FOR EACH ROW setelah trigger timing dan event

```
FOR EACH ROW
```

4. Deklarasikan variabel yang dibutuhkan

```
DECLARE
minsal employees.salary%TYPE;
maxsal employees.salary%TYPE;
e_invalid_salrange EXCEPTION;
```

5. Tuliskan isi/ body dari trigger

```
BEGIN
```

```
SELECT MIN(salary), MAX(salary) INTO minsal, maxsal
FROM employees
WHERE job_id = :NEW.job_id;
IF (minsal < :NEW.min_salary) OR (maxsal > :NEW.max_salary) THEN
RAISE e_invalid_salrange;
END IF;
EXCEPTION
WHEN e_invalid_salrange THEN
RAISE_APPLICATION_ERROR(-20550,'Update gagal karena nilai salary
diluar range!');
END check_sal_range;
/
```

14.3.5.3 Pengamatan

1. Lakukan perubahan nilai max_salary untuk job_id 'IT_PROG' menjadi 5000. Jelaskan apa yang terjadi?
2. Apakah bisa timing BEFORE pada trigger tersebut diganti menjadi AFTER?jelaskan!
3. Apa yang terjadi jika statement FOR EACH ROW dihilangkan?

14.4 TEST AKHIR

1. Buatlah statement trigger yang bernama TRG_RESTRICT_DEL_EMPLOYEE untuk mencegah penghapusan data-data employees pada hari kerja (senin s/d jumat) antara jam 09.00 s/d 17.00!
2. Buatlah sebuah tabel bernama REGION_HISTORY yang memiliki 6 kolom yaitu OLD_REGION_ID, OLD_REGION_NAME, NEW_REGION_ID, NEW_REGION_NAME, CHANGE_TIME, DESCRIPTION. Kemudian modifikasi trigger TRG_DML_REGIONS_WARNING sehingga ketika terjadi operasi DML akan tercatat nilai lama, nilai baru, waktu perubahan, dan deskripsinya (insert, update, delete)!

15 BAB XV TRIGGER LANJUT

15.1 IDENTITAS

Kajian

Trigger

Topik

1. System event trigger
2. Pemanggilan procedure dari trigger
3. Mutating table

Referensi

1. Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol : O'Reilly Media, 2009.
2. Srivastava, Tulika, and Glenn Stokol. Oracle Database 10g: Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher, 2006.
3. Urman, Scott, Ron Hardman, and Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston: McGraw-Hill, 2004.
4. Rosenzweig, Benjamin and Elena Silvestrova Rakhimov. Oracle PL/SQL by Example, 4th Edition. Boston:Addison-Wesley.

Kompetensi Utama

1. Mampu membuat database or system event trigger
2. Mampu melakukan pemanggilan procedure dari trigger
3. Mampu menjelaskan dan mendemokan mutating table pada trigger

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 2 x 120 menit

Parameter Penilaian

1. Tugas Pendahuluan: 20%
2. Jurnal Hasil Pengamatan: 40%
3. Tugas Akhir: 40%

15.2 TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan ini, jika sudah selesai, kumpulkan kepada asisten praktikum anda. Waktu pengerjaan maksimal 15 menit.

1. Sebutkan dan jelaskan jenis-jenis event yang dapat membangkitkan trigger!
2. Apa yang dimaksud dengan CALL statement?
3. Apa yang dimaksud dengan mutating table?
4. Jelaskan mengenai pemanfaatan database trigger untuk Auditing dan Security!

15.3 PRAKTIK

15.3.1 Konsep System Event Trigger

Selain dipicu oleh perintah-perintah DML, database trigger juga dapat dipicu oleh perintah DDL maupun system event pada level schema dan database. Sebagai contoh kita dapat membuat trigger yang akan dipicu pada saat kita membuat (create), mengubah (alter), dan menghapus (drop) tabel. Berikut ini adalah beberapa event yang dapat memicu dieksekusinya trigger:

- Perintah DDL pada objek yang ada pada suatu schema atau database.
- Pada saat seorang user melakukan login atau logoff pada schema atau database
- Pada saat database shutdown atau startup
- Pada saat terjadi error

Kemampuan trigger untuk membaca beberapa event yang sudah disebutkan diatas akan sangat membantu bagi kita untuk menerapkan suatu aturan/ rule dengan tujuan keamanan dan auditing.

Kemampuan lain dari trigger adalah melakukan pemanggilan procedure menggunakan perintah CALL. Perintah CALL ini memungkinkan kita untuk memanggil sebuah stored procedure atau procedure dari sebuah package dengan mudah dan singkat. Sangat disarankan untuk membuat trigger dengan isi sederhana/ singkat karena akan mempermudah kita untuk melakukan pengelolaan subprogram. Untuk itu proses yang dilakukan oleh trigger disarankan untuk dipindahkan/ ditaruh pada subprogram seperti stored procedure dan package.

Salah satu hal penting yang harus diperhatikan pada saat kita menggunkan trigger adalah penanganan error mutating table. Error mutating table ini terjadi pada saat kita membuat row trigger dimana pada isi/ body dari trigger yang kita buat terdapat proses yang melibatkan objek (biasanya tabel) yang sama dengan objek tabel pada event pemicunya. Trigger berhasil dibuat namun pada saat tereksekusi akan terjadi error mutating table karena secara default trigger tidak diijinkan untuk membaca data yang tidak konsisten saat terjadi operasi DML. Transaksi yang terjadi pada trigger biasanya merupakan transaksi yang bergantung pada transaksi lain sehingga ketika terjadi kegagalan pada transaksi induk maka transaksi yang ada pada trigger akan di-rollback. Beberapa cara yang dapat dilakukan untuk mengatasi error mutating table antara lain:

1. Menggunakan statement trigger jika memungkinkan
2. Menyimpan nilai summary data yang dibutuhkan pada trigger body ke dalam tabel yang terpisah dengan memastikan nilai summary data tersebut tetap up-to-date.
3. Menyimpan summary data yang dibutuhkan pada package. Solusi ini dapat dilakukan jika kita mengunakan statement trigger dengan timing BEFORE
4. Menggunakan compound trigger. Compoud trigger merupakan fitur baru pada oracle 11g yang memungkinkan kita untuk mengkombinasikan lebih dari satu timing pada satu trigger. Bahkan kita dapat membuat row trigger dan statement trigger dalam satu trigger.

15.3.2 DDL Trigger

Pada bagian ini akan dipraktekkan tentang DDL trigger. Trigger ini akan dijalankan ketika terjadi operasi DDL seperti CREATE, ALTER, dan DROP.

15.3.2.1 Soal

Buatlah trigger bernama TRG_DDL_LOG untuk mencatatat operasi-operasi DML yang ada pada schema HR. catatlah operasi DML ke dalam tabel yang bernama ddl_log(object_type, object_name, object_owner, operation_date)!

15.3.2.2 Langkah Penyelesaian

1. Buat terlebih dahulu tabel ddl_log

```
CREATE TABLE ddl_log (
object_type VARCHAR2(20),
object_name VARCHAR2(30),
object_owner VARCHAR2(30),
operation_desc VARCHAR2(30),
operation_date DATE);
```

2. Definisikan nama trigger yang akan dibuat!

```
CREATE OR REPLACE TRIGGER TRG_DDL_LOG
```

3. Tentukan trigger timing dan event-nya

```
AFTER CREATE OR DROP OR ALTER ON SCHEMA
```

4. Deklarasikan variabel yang dibutuhkan

```
DECLARE
sys_event VARCHAR2(50);
```

5. Tuliskan isi/ body dari trigger

```
BEGIN
sys_event:= ora_sysevent;
INSERT INTO ddl_log VALUES(ORA_DICT_OBJ_TYPE, ORA_DICT_OBJ_NAME,
ORA_DICT_OBJ_OWNER, sys_event, SYSDATE);
END;
/
```

15.3.2.3 Pengamatan

1. Buatlah sebuah tabel, kemudian lihat isi data pada tabel ddl_log untuk memastikan proses ddl tercatat!
2. Tambahkan kolom atau ubah struktur data pada tabel yang sudah anda buat, kemudian lihat isi data pada tabel ddl_log untuk memastikan proses ddl tercatat!
3. Hapuslah tabel yang sudah anda buat, kemudian lihat isi data pada tabel ddl_log untuk memastikan proses ddl tercatat!
4. Apa yang terjadi jika `ON SCHEMA` diganti dengan `ON DATABASE`?
5. Apa yang terjadi jika trigger event-nya ditambah dengan `OR TRUNCATE`?

15.3.3 Trigger On System Event

Pada bagian ini akan dipraktekkan tentang trigger yang dipicu karena system event, misalkan: setelah user logon, sebelum logoff, setelah database startup, dll.

15.3.3.1 Soal

Buatlah trigger bernama TRG_SCHEMA_LOGIN. Trigger ini akan mencatat user yang login pada schema HR, waktu login, dan keterangan loginnya. Terlebih dahulu buatlah tabel access_log (user_id, log_date, action) untuk menyimpan data akses schema HR.

15.3.3.2 Langkah Penyelesaian

1. Terlebih dahulu buatlah tabel access_log!

```
CREATE TABLE access_log(
user_id VARCHAR2(20),
log_date DATE,
action VARCHAR2(50)
);
```

2. Definisikan nama trigger yang akan dibuat!

```
CREATE OR REPLACE TRIGGER TRG_SCHEMA_LOGIN
```

3. Tentukan trigger timing dan event-nya

```
AFTER LOGON ON SCHEMA
```

4. Tuliskan isi/ body dari trigger

```
BEGIN
INSERT INTO access_log(user_id,log_date,action)
VALUES (USER, SYSDATE, 'Logging on');
END;
/
```

15.3.3.3 Pengamatan

1. Logout-lah dari schema HR, kemudian login lagi ke schema HR. Lihat tabel access_log apa yang terjadi? (perhatikan: jika tabel access_log berisi data berarti TRG_SCHEMA_LOGIN berhasil dijalankan)
2. Apa yang terjadi jika timingnya diganti dengan AFTER LOGOFF? Jelaskan!

15.3.4 Call Statement pada Trigger

Pada bagian ini akan dipraktekkan mengenai pemanggilan procedure pada trigger menggunakan perintah CALL .

15.3.4.1 Soal

Modifikasi `TRG_RESTRICT_SALARY_UPDATE` yang sudah anda buat sebelumnya dengan memindahkan body-nya menjadi stored procedure `PROC_RESTRICT_SALARY`. Kemudian panggil stored procedure tersebut di dalam trigger dengan menggunakan perintah CALL.

15.3.4.2 Langkah Penyelesaian

1. Buatlah terlebih dahulu stored procedure `PROC_RESTRICT_SALARY`!

```
CREATE OR REPLACE PROCEDURE PROC_RESTRICT_SALARY IS
BEGIN
```

```
IF  (TO_CHAR(SYSDATE,'DY')  IN  ('MON','TUE','WED','THU','FRI'))  AND
(TO_CHAR(SYSDATE,'HH24') BETWEEN '08' AND '17') THEN
     RAISE_APPLICATION_ERROR(-20504,'Tidak     boleh     melakukan
perubahan gaji pegawai pada hari dan jam kerja!!!');
END IF;
END;
/
```

2. Buat ulang trigger TRG_RESTRICT_SALARY_UPDATE dengan memanggil stored procedure PROC_RESTRICT_SALARY menggunakan perintah CALL!

```
CREATE OR REPLACE TRIGGER TRG_RESTRICT_SALARY_UPDATE
BEFORE UPDATE OF SALARY ON EMPLOYEES
CALL proc_restrict_salary
/
```

15.3.4.3 Pengamatan

1. Coba lakukan update salary di tabel employees pada hari dan jam kerja!jelaskan apa yang terjadi!
2. Pada soal diatas dilakukan pemanggilan satu stored procedure pada trigger, coba jelaskan cara pemanggilan banyak stored procedure pada trigger menggunakan perintah CALL!

15.3.5 Mutating Table

Pada bagian ini akan dipraktekkan bagimana proses mutating table dan cara mengatasinya.

15.3.5.1 Soal

Buatlah trigger bernama TRG_TES_MUTATING dimana trigger ini akan menampilkan jumlah data yang terdapat pada tabel regions setelah dilakukan operasi DML di tabel regions!

15.3.5.2 Langkah Penyelesaian

1. Definisikan nama trigger yang akan dibuat!

```
CREATE OR REPLACE TRIGGER TRG_TES_MUTATING
```

2. Tentukan trigger timing dan event-nya

```
AFTER INSERT ON REGIONS
```

3. Tambahkan statement FOR EACH ROW

```
FOR EACH ROW
```

4. Deklarasikan variabel yang dibutuhkan

```
DECLARE
jumlah NUMBER;
```

5. Tuliskan isi/ body dari trigger

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Insert berhasil…');
SELECT COUNT(*) INTO jumlah FROM REGIONS;
```

```
DBMS_OUTPUT.PUT_LINE('Jumlah data di tabel regions = '||jumlah);
END;
/
```

6. Jalankan perintah update sebagai berikut!

```
INSERT INTO REGIONS VALUES(5, 'Indonesia');
```

Ketika perintah tersebut dijalankan pasti akan terjadi error sbb:

insert into regions values(5,'indonesia')
*****
ERROR at line 1:
ORA-04091: table HR.REGIONS is mutating, trigger/function may not see it
ORA-06512: at "HR.TRG_TES_MUTATING", line 4
ORA-04088: error during execution of trigger 'HR.TRG_TES_MUTATING'

Error tersebut terjadi karena tabel regions menjadi objek pada trigger event dan sekaligus menjadi sumber data pada cursor yang ada pada trigger body. Terdapat beberapa cara untuk mengatasi kasus diatas:

1. Menggunakan STATEMENT TRIGGER

```
CREATE OR REPLACE TRIGGER TRG_TES_MUTATING
AFTER INSERT ON REGIONS
DECLARE
jumlah NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('Insert berhasil...');
SELECT COUNT(*) INTO jumlah FROM REGIONS;
DBMS_OUTPUT.PUT_LINE('Jumlah data di tabel regions =
'||jumlah);
END;
/
```

2. Menggunakan COMPOUND TRIGGER

Compound trigger merupakan fitur baru yang mulai ada pada oracle 11g. Jenis trigger ini memungkinkan kita untuk menggunakan lebih dari satu timing dalam sebuah trigger.

```
CREATE OR REPLACE TRIGGER TRG_TES_MUTATING
FOR INSERT ON REGIONS
COMPOUND TRIGGER
jumlah NUMBER;
AFTER EACH ROW IS
BEGIN
DBMS_OUTPUT.PUT_LINE('Insert berhasil…');
END AFTER EACH ROW;

AFTER STATEMENT IS
BEGIN
```

```
SELECT COUNT(*) INTO jumlah FROM REGIONS;
DBMS_OUTPUT.PUT_LINE('Jumlah data di tabel regions =
'||jumlah);
END AFTER STATEMENT;
END;
/
```

15.3.5.3 Pengamatan

1. Jalankan solusi pertama dan kedua dan jelaskan hasilnya!
2. Apa yang terjadi jika bagian AFTER STATEMENT diganti dengan BEFORE STATEMENT?

15.4 TEST AKHIR

1. Perhatikan trigger TRG_DDL_LOG yang sudah anda buat. Trigger tersebut hanya berlaku untuk operasi DDL pada schema HR saja. Tugas anda adalah menjelaskan langkah-langkah yang harus dilakukan supaya trigger tersebut dapat berlaku/ berjalan untuk semua schema yang ada pada sebuah database!
2. Buatlah sebuah stored procedure bernama PROC_BYE. Procedure ini akan mengisi kolom action pada tabel access_log dengan kalimat “Oracle PL/SQL is Awesome!!!, Thanks for our lecturer for all…”. Panggillah stored procedure ini melalui trigger bernama TRG_BYE **menggunakan perintah CALL** yang akan dijalankan ketika sebelum LOGOFF pada schema HR!

16 DAFTAR PUSTAKA

[1] Feuerstein, Steven, and Bill Pribyl. Oracle PL/SQL Programming, 5th Edition. Sebastopol: O'Reilly Media, 2009.

[2] Benjamin Rosenzweig, Elena Silvestrova. Oracle PL/SQL by Example 3rd edition. Prentice Hall PTR, 2003.

[3] Michael Rosenblum, Paul Dorsey. Oracle PL/SQL for Dummies. Wiley Publishing, Inc., 2006.

[4] -.PL/SQL Framework. http://cisku.com/2013/07/18/plsql-framework/ (accessed January 2014).

[5] -.Contoh PL/SQL Sederhana. http://cisku.com/2013/07/18/contoh-plsql-sederhana/ (accessed January 2014).

[6] -.Hitung Diskon Pembelian Barang. http://cisku.com/2013/07/18/hitung-diskon-pembelian-barang/ (accessed January 2014).