
# Anonymous Block & Tipe Data

Disusun oleh:

Elis Hernawati,S.T, M.KOM – elishernawati@tass.telkomuniversity.ac.id

**Program Studi D3 Manajemen Informatika -Fakultas Ilmu Terapan**


# Tujuan Pembelajaran

Mahasiswa mampu mempersiapkan kerangka anonymous block lengkap dengan tipe datanya

# Pendahuluan

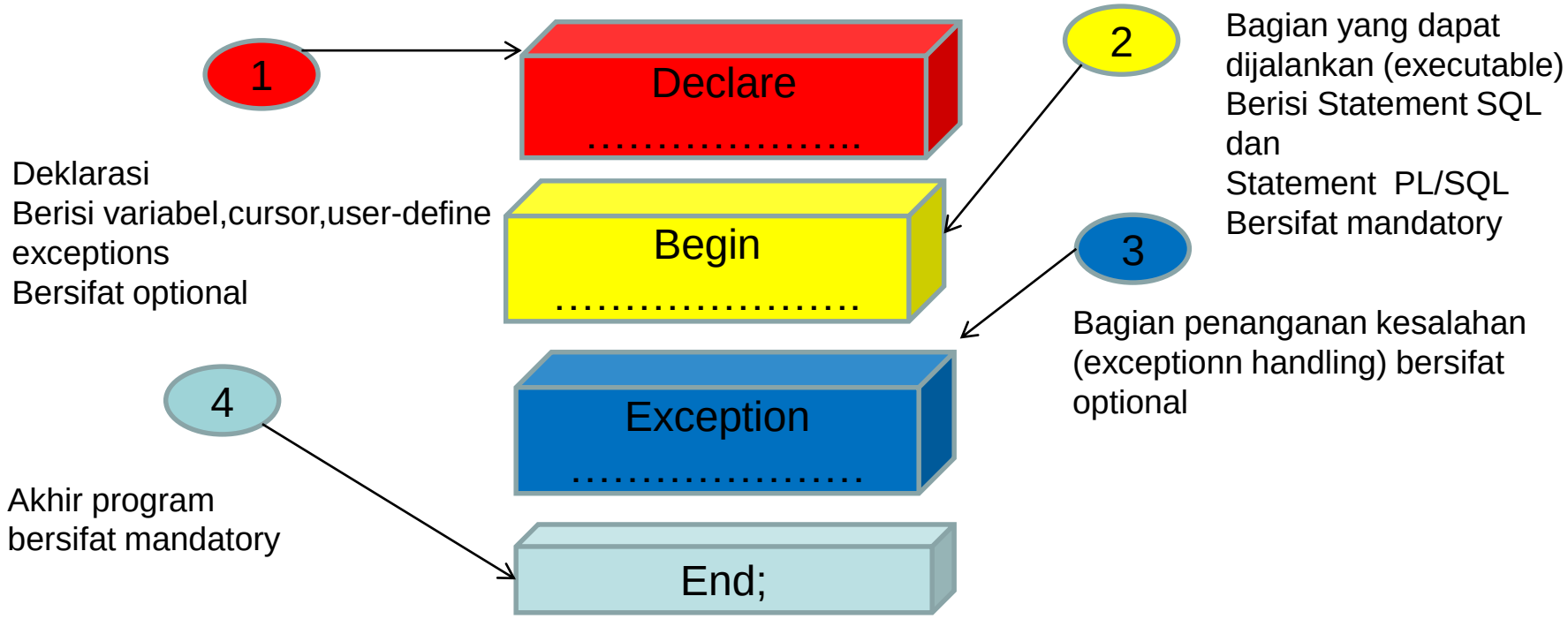
PL/SQL adalah singkatan dari Procedural Language / Structured Query Language, merupakan gabungan dari bahasa pemrograman prosedural dengan Sintaks SQL.


Type blok PL/SQL terdiri dari:

1. Blok non-modular PL/SQL atau dikenal dengan **anonymous block PL/SQ**L
2. Blok Modular

# Blok PL/SQL Non Modular (Anonymous Block)

- Anonymous Block adalah blok PL/SQL tidak bernama ,tidak menggunakan parameter dan tidak disimpan dalam database sehingga tidak bisa direferensi /dipanggil oleh blok PL/SQL lain
- Jika ingin menggunakan kode program yang sama anonymous block harus diketik dan dieksekusi kembali.
- Blok dalam PL/SQL Non Modular terdiri dari bagian -bagian, sebagai berikut :



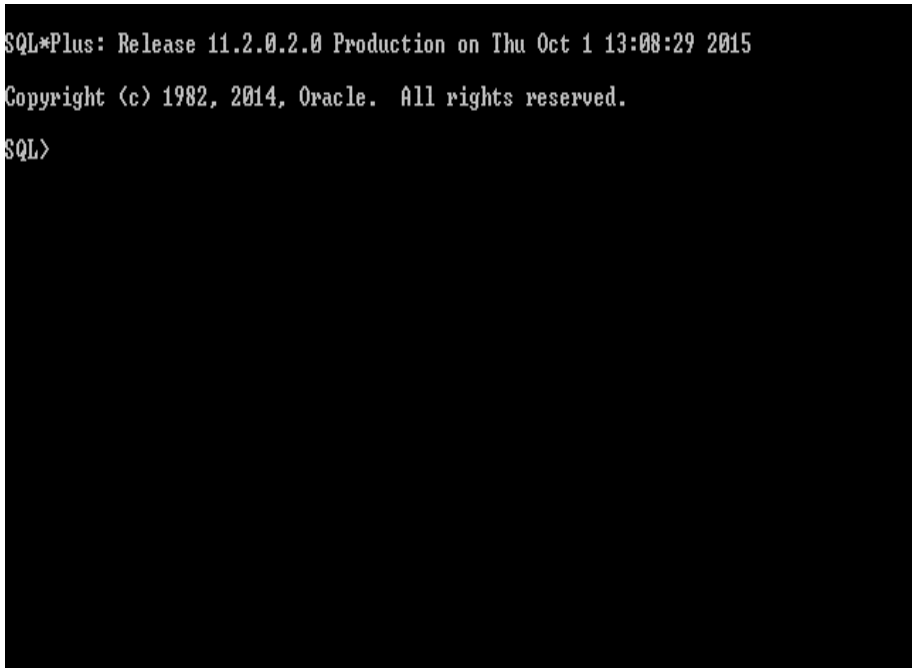
# Contoh 1: Anonymous Block PL/SQL

- Membuat sebuah Anonymous Block PL/SQL untuk menampilkan sebuah kalimat

**“SELAMAT DATANG DI PEMROGRAMAN BASIS DATA”**

- Jalankan SQLPLUS
- LOGIN KE SYSTEM ORACLE
- Ketik perintah berikut untuk menampilkan hasil output

  Set Serveroutput On
- Ketikkan Program Anonymous Block
- OUTPUT yang dihasilkan




DBMS_OUTPUT merupakan paket yang disediakan oleh Oracle PL / SQL dan PUT_LINE merupakan salah satu prosedur yang dikemas. Menampilkan nilai-nilai pada SQL Plus * terminal yang harus diaktifkan dengan SET SERVEROUTPUT ON terlebih dahulu. Untuk menjalankan kode sampel ini, login ke SQL * Plus. PL / SQL blok diakhiri dengan tanda garis miring / atau garis byitself.

# Deklarasi Variabel dan Tipe Data

- Bagian deklarasi variabel di antara kata kunci DECLARE dan BEGIN. Penamaan variabel tidal bersifat case sensitive. Tipe data variabel dapat berupa salah satu tipe data database Oracle atau tipe data built in PL/SQL.
- Sintaks:

**Identifier *typedata* [(presisi, skala)] [NOT NULL] [:=*iekspresi];***

ekspresi bisa merupakan literal, variabel yang lain atau sebuah ekspresi yang terdiri atas operator dan fungsi. Jika nilai inisial/awal tidak diberikan, maka suatu variabel akan diberikan nilai NULL untuk nilai inisialnya.

# Tipe Data Dasar

Contoh untuk data karakter:

**alamat VARCHAR2(20);**
**kodepos CHAR(5) := ‘40363’;**

Contoh untuk tipe data number:

**bonus NUMBER(7,2);**
**stok NUMBER NOT NULL := 0;**

Contoh untuk tipe data tanggal:

**tgllahir date;**

# Tipe Data pada PL/SQL

| Tipe Data | Keterangan |
|---|---|
| BOOLEAN | Data lojikal dengan nilai TRUE atau FALSE. |
| DATE | Data tanggal waktu. Nilai yang valid adalah antara 1 Januari 4712 SM sampai dengan 31 Desember 9999. |
| NUMBER [(*p* [,*s*])] | Tipe data numerik dengan *p* angka penting dan sejumlah *s* angka penting di belakang koma. Nilai *p* adalah integer dengan nilai maksimal 38 dan nilai *s* berada pada rentang -84 sampai dengan 127. Nilai *s* negatif berarti pembulatan sampai dengan $10^s$ terdekat. |
| FLOAT | Turunan dari NUMBER. Presisi sampai dengan 38 digit. |
| DOUBLE PRECISION | Sama dengan FLOAT. |
| REAL | Turunan dari number. Presisi sampai dengan 18 digit. |
| DEC [(*p* [,*s*])] | Sama dengan NUMBER [(*p* [,*s*])]. |
| DECIMAL [( *p* [,*s*])] | Sama dengan NUMBER [(*p* [,*s*])]. |
| NUMERIC [(*p* [,*s*])] | Sama dengan NUMBER [(*p* [,*s*])]. |
| INTEGER [(*n*)] | Sama dengan NUMBER [(*n*,0)]. |
| INT [(*n*)] | Sama dengan NUMBER [(*n*,0)]. |
| SMALLINT [(*n*)] | Sama dengan NUMBER [(*n*,0)]. |
| BINARY_INTEGER | Tipe variabel ini digunakan menyimpan nilai mulai dari -2.147.483.647 s/d 2.147.483.647 |
| NATURAL | Bagian dari binary integer, mampu menyimpan mulai dari 0 s/d 2.147.483.647. |
| NATURALN | Bagian dari binary integer, mampu menyimpan mulai dari 0 s/d 2.147.483.647. Tipe data ini tidak boleh bernilai NULL. |
| POSITIVE | Bagian dari binary integer, mampu menyimpan mulai dari 1 s/d 2.147.483.647 |
| POSITIVEN | Bilangan integer dengan rentang nilai 1 sampai dengan 2147483647. Tipe data ini tidak boleh bernilai NULL. |
| SIGNTYPE | Tipe data bilangan yang bernilai -1, 0 atau 1. |
| PLS_INTEGER | Bilangan integer dengan rentang nilai -2147483647 sampai 2147483647. |
| VARCHAR2(*n*) | Data karakter dengan panjang tidak tetap. Nilai *n* minimum sama dengan 1 dan maksimum sama dengan 32767 byte. |
| VARCHAR(*n*) | Sama dengan VARCHAR2(*n*). |
| CHAR [(*n*)] | Data karakter dengan panjang tetap sebesar *n* byte. Nilai *n* maksimum adalah 32767. Nilai *n* minimum dan juga nilai default adalah 1. |
| STRING(*n*) | Sama dengan VARCHAR2(*n*). |
| CHARACTER [(*n*)] | Sama dengan CHAR(*n*). |
| LONG [(*n*)] | Data karakter dengan panjang tidak tetap. Nilai *n* maksimum sama dengan 32760 byte. |
| NCHAR [(*n*)] | Data karakter dengan panjang tetap. Panjang maksimum sama dengan 32767 byte. maksimum bergantung pada *national character set* yang dipakai. Nilai default adalah 1. |
| NVARCHAR2(*n*) | Data karakter dengan panjang tidak tetap. Panjang maksimum sama dengan 32767 byte. Nilai *n* |

# Pendeklarasian Konstanta

Sintaks:

**Identifier CONSTANT typedate[(presisi,skala)] := ekspresi;**

Contoh:

```
pi CONSTANT NUMBER(9,2):=3.14;
```

# Type Data Reference

merupakan salah satu dari **macam-macam tipe data** pada pl/sql yang digunakan untuk mereferensikan tipe data pada **variabel** lain.

## 1. Attribut %TYPE

**%TYPE** adalah attribut yang digunakan untuk mendeklarasikan sebuah variable yang sesuai dengan:

- Definisi sebuah kolom database
- Deklarasi variable lain

Selain itu, %TYPE diawali dengan

- Tabel dan Kolom Database
- Nama dari Variabel yang dideklarasikan

**Deklarasi Variabel dengan %TYPE**

Syntax:

**identifier table.column_name%TYPE;**

Contoh :

namapegawai employees.last_name**%TYPE**;

## 2. Attribut %ROWTYPE

%**ROWTYPE** digunakan untuk mendeklarasikan sebuah variabel yang sesuai dengan tipe data sejumlah kolom pada **tabel** atau **view** di database. Beda halnya dengan %**TYPE** yang hanya bisa mereferensikan satu tipe kolom atau satu variabel saja.

**Keuntungan menggunakan %ROWTYPE :**

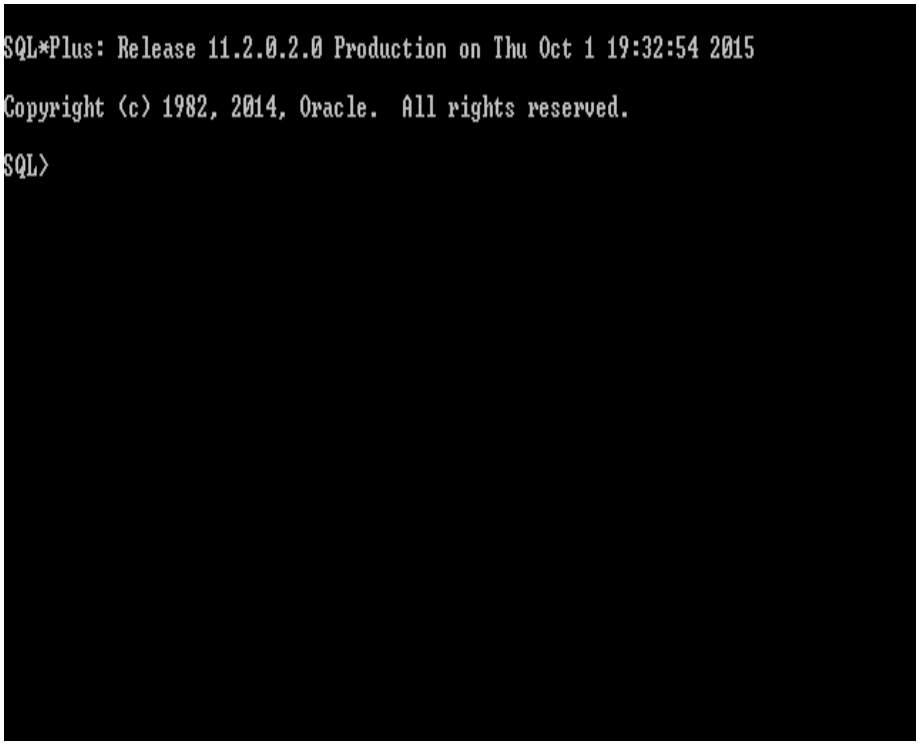
- Kita tidak perlu mengetahui berapa jumlah dan tipe data kolom pada tabel atau view yang ingin kita referensikan
- Adanya kemungkinan terjadinya perubahan jumlah dan tipe data kolom pada tabel atau view pada saat run time
- Cocok digunakan saat menerima data menggunakan SELECT * statement

Syntax :

DECLARE identifier reference%ROWTYPE;

# Contoh 2: Anonymous Block PL/SQL

- Dengan menggunakan tabel Employees dari Schema HR, dibuat anonymous blok menampilkan nama pegawai dan lama bekerja (dalam tahun) untuk pegawai dengan employee_id nya 180
- Jalankan SQLPlus
- Login Ke Schema HR
- Ketik perintah berikut untuk menampilkan hasil output **set serveroutput on**
- Ketikkan program ANONYMOUS BLOCK
- OUTPUT yang dihasilkan



# Composite Data Types

- Composite Data Types atau Tipe Data Komposit merupakan **Tipe data** yang dapat menampung banyak nilai, tidak seperti **tipe data scalar** yang hanya dapat menampung satu nilai
- Tipe Data Komposit ada 2 :

1. **RECORD**
2. **COLLECTION**
   a. INDEX BY TABLE
   b. NESTED TABLE
   c. VARRAY

# RECORD

- Record merupakan struktur data komposit atau gabungan dari beberapa field, baik yang bertipe sama maupun berbeda. Kita dapat menggunakan record untuk menyimpan **variabel-variabel** yang berhubungan, Misalanya record Alamat dimana di dalamnya terdapat field NamaJalan, NoJalan, NoRumah, Kota, kodePos dan lain-lain.
- Untuk mendefinisikan suatu variabel yang bertipe Record, kita menggunakan Sebuah Keyword yaitu RECORD

Syntax:

**TYPE** NamaRecord **IS RECORD (**

Field1 TipeData,

Field2 TipeData,

…

**);**

Contoh :

**TYPE** Alamat **IS RECORD** (

NamaJalan **VARCHAR2(100),**

NOJalan **NUMBER,**

NoRumah **VARCHAR2(5),**

Kota **VARCHAR2(20)**

**);**

**TYPE** PemilikRumah **IS RECORD**(

Nama **VARCHAR2(100),**

Domisili Alamat –menggunakan tipe record alamat

);

```
SET SERVEROUTPUT ON
DECLARE
TYPE Alamat IS RECORD (
NamaJalan VARCHAR2(100),
NOJalan NUMBER,
NoRumah VARCHAR2(5),
Kota VARCHAR2(20)
);
TYPE PemilikRumah  IS
RECORD(
Nama VARCHAR2(100),
Domisili Alamat
);
REC PemilikRumah;  -- REC merupakan nama
variabel
BEGIN
REC.Nama :='Riyansyah Iqbal Saputra' ;
REC.Domisili.NamaJalan := 'Jalan Delima';
REC.Domisili.NoJalan := 9;
REC.Domisili.NoRumah := '9B' ;
REC.Domisili.Kota := 'Jakarta' ;
DBMS_OUTPUT.PUT_LINE (REC.Nama);
DBMS_OUTPUT.PUT_LINE
(rec.domisili.namaJalan ||' ' ||
rec.domisili.noJalan || ' '||'No' ||' ' ||
rec.domisili.NoRumah || ' ' || rec.domisili.kota);
END;
```

Hasilnya :

*Riyansyah Iqbal Saputra*
*Jalan Delima 9 No 9B Jakarta*

# Index_By Table (Associative arrays )

- Associative arrays atau biasa disebut dengan tabel PL/SQL merupakan kumpulan key-value berpasang-pasangan, yang mana setiap key adalah unik dan digunakan untuk mencari nilai yang terkait di dalam array. Key dapat berupa integer atau string.
- Pemberian-pemberian nilai selanjutnya menggunakan key yang sama akan meng-update entry yang sama. Penting untuk memiliki key yang bernilai unik, baik dengan menggunakan primary key dari SQL table, atau dengan menggabungkan string-string secara bersama-sama untuk membentuk nilai unik.

Sebagai contoh, berikut ini adalah deklarasi dari tipe associative array, dan dua arrays

dengan tipe tersebut, menggunakan key-key berupa string-string:

```
DECLARE
TYPE population IS TABLE OF NUMBER -- Associative array type
INDEX BY VARCHAR2(64); -- indexed by string
city_population population; -- Associative array variable
i VARCHAR2(64); -- Scalar variable
BEGIN
-- Add elements (key-value pairs) to associative array:
city_population('Smallville') := 2000;
city_population('Midland') := 750000;
city_population('Megalopolis') := 1000000;
-- Change value associated with key 'Smallville':
city_population('Smallville') := 2001;
-- Print associative array:
i := city_population.FIRST; -- Get first element of array
WHILE i IS NOT NULL LOOP
DBMS_Output.PUT_LINE ('Population of ' || i || ' is ' || city_population(i));
i := city_population.NEXT(i); -- Get next element of array
END LOOP;
END;
```

# Note :

**Associative arrays** membantu kita merepresentasikan kumpulan data dengan ukuran yang berubah-ubah dengan pencarian cepat untuk elemen-elemen tunggal tanpa perlu mengetahui posisinya di dalam array dan tanpa harus mencari melalui seluruh elemen-elemen array. Hal ini seperti versi sederhana dari SQL table dimana kita dapat menampilkan nilai-nilai berdasarkan primary key. Namun pada Associatibe arrays kita tidak dapat melakukan operasi DML. Untuk pencarian data dengan penyimpanan sementara secara sederhana, associative array mengijinkan kita untuk menghindari penggunaan disk space dan network operation yang dibutuhkan oleh SQL tables.

# Referensi


1. Feurstein , Steven, and Bill Pribyl. Oracle PL/SQL Programming,  5th Edition. Sebastopol: O'Reilly Media,2009
2. Srivastava, Tulika, dan Glenn Stokol. Oracle Database 10g:Develop PL/SQL Program Units 2nd Edition. Boston: Oracle Publisher,2006
3. Urman, Scott, Ron Hardman, dan Michael Laughlin. Oracle Database 10g PL/SQL Programming. Boston:McGraw-Hill,2004.
4. Dedy Rahman Wijaya,S.T.,M.T.,OCA,Modul Praktikum Pemrograman Basis Data, Universitas Telkom, 2014