# Presentasi UAS BDL

Presented by Noval Nugraha | 1 TIE

# Sebelum itu... Kenapa BDL...?

Sebelum masuk ke pembahasan materi, saya ingin menyampaikan beberapa alasan subjektif saya mengenai pembelajaran BDL sejauh ini secara singkat. Selama saya mempelajari basis data sejauh ini, dan bertemu dengan mata kuliah BDL (Basis Data Lanjut) serta mengerjakan project collabs. Saya sadar betapa pentingnya data, terlebih bila datanya akan disajikan dan dikelola untuk suatu keperluan baik pribadi maupun sebuah kepentingan kelompok. Adanya pembelajaran basis data bagi saya sungguh membantu sekali bagi mahasiswa untuk bisa mengelola data yang akan menjadi informasi pengerjaan suatu project bagi mereka. Terimakasih telah membaca alasan subjektif saya ini.

# Bahasan...

01. Index & View

02. Sequence & Synonym

03. PL/SQL & Procedure

04. Function & Cursor

05. Exception & Trigger

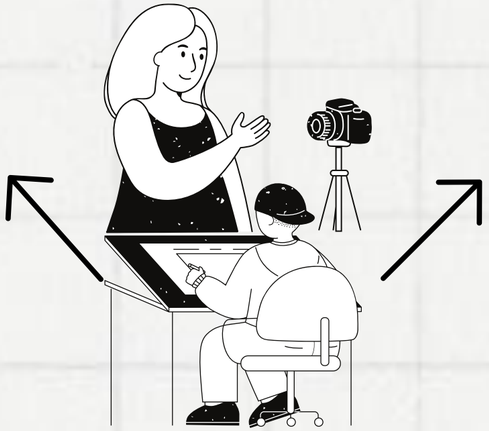
# 1. Index & View

Index pada dasarnya merupakan sebuah query yang berguna untuk mempercepat pengambilan suatu data pada tabel di database tertentu. Index bisa melakukan hal tersebut sebab saat implementasinya seperti mencari data di suatu tabel database, Index tidak perlu memindai setiap baris yang ada di dalam tabel.

View, apa pernah terlintas di kepala kita kenapa untuk mengolah suatu data yang memiliki query begitu panjang tidak bisa dibuat lebih singkat saja...?
Peran View di sini akan memecahkan masalah kita tanpa perlu mengetikkan query yang panjang berulang kali (tidak efektif), adanya view kita bisa mempersingkat query panjang itu dengan sebuah penamaan tetapi query view ini hanya bisa untuk dilihat saja.

# Contoh

## Index & View



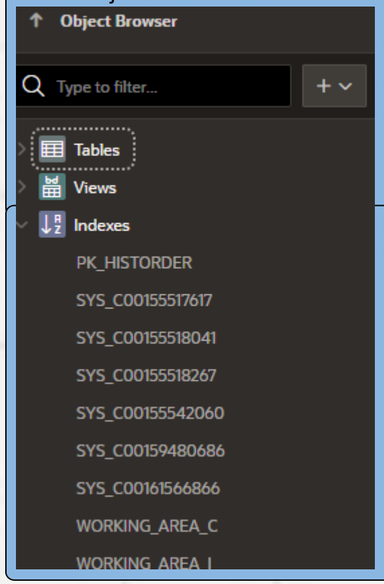
```
33  select
34  index_name, index_type, status, table_name
35  from all_indexes
36  where table_name = 'CUSTOMER' OR
37  table_name = 'ORDERS' OR
38  table name = 'AGENTS' order by table name asc
```

Results | Explain | Describe | Saved SQL | History

| INDEX_NAME | INDEX_TYPE | STATUS | TABLE_NAME |
|---|---|---|---|
| WORKING_AREA_I | NORMAL | VALID | AGENTS |
| SYS_C00155517617 | NORMAL | VALID | AGENTS |
| SYS_C00156408407 | NORMAL | VALID | CUSTOMER |
| WORKING_AREA_C | NORMAL | VALID | CUSTOMER |
| SYS_C00155518041 | NORMAL | VALID | CUSTOMER |
| SYS_C00156408406 | NORMAL | VALID | CUSTOMER |
| SYS_C00155518267 | NORMAL | VALID | ORDERS |

7 rows returned in 3.23 seconds  Download

Anda bisa mengecek dulu apa saja index yang ada di database kita, bisa menggunakan query di atas sebelum akan menambahkan index baru atau ketika anda menggunakan apex oracle anda bisa saja langsung mengunjungi menu object browser dan bisa menemukan list dari index anda di sana.




Lihatlah gambar di samping dengan mengakses object browser di apex oracle, anda dapat mengetahui langsung index anda di sana. Object browser akan sangat berguna sekali untuk pembahasan kita kedepannya, sebab di sini setiap index, view, tabel dan informasi mengenai database anda bisa dilihat.

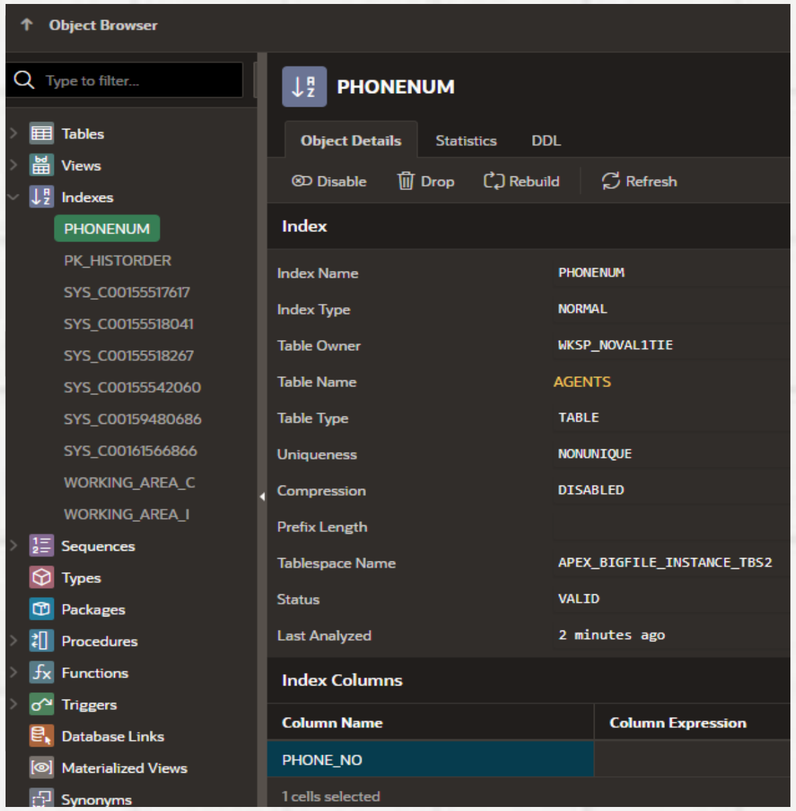
```
1  create index
2  PhoneNum on
3  agents (phone_no)
```

Results | Explain | Describe

Index created.

0.05 seconds

Membuat sebuah Index, contohnya bisa dilihat di samping. Membuat sebuah index PhoneNum untuk kolom Phone_no yang berada di tabel agents. Anda bisa mengecek lagi di object browser apakah index yang anda buat tadi sudah jadi dan bila anda mau menghapusnya cukup drop kan saja.



```sql
select c.cust_name, c.working_area as cust_w_area, a.agent_name,
a.working_area as agent_w_area, o.ord_date, o.ord_amount, o.advance_amount
from customer c inner join orders o on c.cust_code = o.cust_code
inner join agents a on o.agent_code = a.agent_code
where a.working_area = 'New York' or a.working_area = 'Mumbai'
```

Results | Explain | Describe | Saved SQL | History

| CUST_NAME | CUST_W_AREA | AGENT_NAME | AGENT_W_AREA | ORD_DATE | ORD_AMOUNT | ADVANCE_AMOUNT |
|---|---|---|---|---|---|---|
| Micheal | New York | Alford | New York | 07/15/2020 | 3000 | 1000 |
| Bolt | New York | Alford | New York | 08/15/2020 | 3500 | 2000 |
| Sasikant | Mumbai | Mukeshi | Mumbai | 04/20/2020 | 2500 | 700 |
| Ramesh | Mumbai | Mukeshi | Mumbai | 07/20/2020 | 500 | 100 |
| Ramesh | Mumbai | Mukeshi | Mumbai | 07/20/2020 | 3500 | 1500 |
| Ramesh | Mumbai | Mukeshi | Mumbai | 06/29/2020 | 1200 | 400 |
| Albert | New York | Alford | New York | 07/10/2020 | 1000 | 300 |
| Avinash | Mumbai | Mukeshi | Mumbai | 06/10/2020 | 5000 | 600 |
| Avinash | Mumbai | Mukeshi | Mumbai | 09/16/2020 | 500 | 100 |
| Avinash | Mumbai | Mukeshi | Mumbai | 06/24/2020 | 500 | 100 |

Bagaimana pendapat anda saat melihat Query ini..?
Panjang sekali bukan..?
Query view akan menjalankan tugasnya di sini untuk membuat query lebih efektif, cukup tambahkan seperti di bawah..

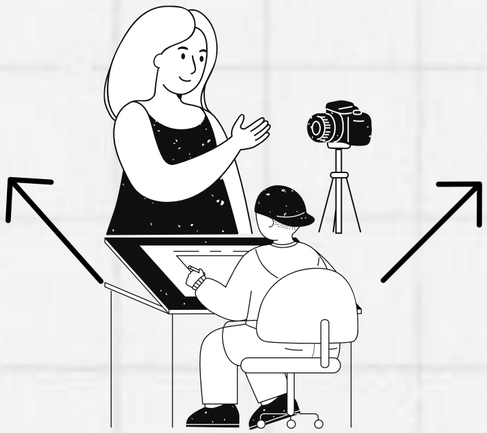
```sql
CREATE OR REPLACE VIEW
"TIGARELASI" ("CUST_NAME", "CUST_W_AREA", "AGENT_NAME", "AGENT_W_AREA", "ORD_DATE", "ORD_AMOUNT", "ADVANCE_AMOUNT") AS
select c.cust_name, c.working_area as cust_w_area, a.agent_name,
a.working_area as agent_w_area, o.ord_date, o.ord_amount, o.advance_amount
from customer c inner join orders o on c.cust_code = o.cust_code
inner join agents a on o.agent_code = a.agent_code
where a.working_area = 'New York' or a.working_area = 'Mumbai'
```

Results | Explain | Describe | Saved SQL | History

Statement processed.

0.06 seconds

Create or Replace View, lalu tambahkan ke query yang panjang tadi maka view sudah dibuat di database anda



# Contoh

## Index & View

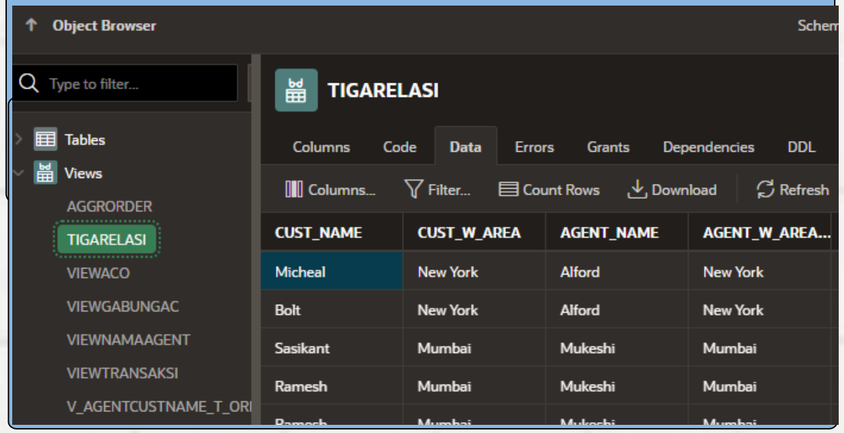
```sql
select * from tigarelasi
```

Results | Explain | Describe | Saved SQL | History

| CUST_NAME | CUST_W_AREA | AGENT_NAME | AGENT_W_AREA | ORD_DATE | ORD_AMOUNT | ADVANCE_AMOUNT |
|---|---|---|---|---|---|---|
| Micheal | New York | Alford | New York | 07/15/2020 | 3000 | 1000 |
| Bolt | New York | Alford | New York | 08/15/2020 | 3500 | 2000 |
| Sasikant | Mumbai | Mukeshi | Mumbai | 04/20/2020 | 2500 | 700 |
| Ramesh | Mumbai | Mukeshi | Mumbai | 07/20/2020 | 500 | 100 |
| Ramesh | Mumbai | Mukeshi | Mumbai | 07/20/2020 | 3500 | 1500 |
| Ramesh | Mumbai | Mukeshi | Mumbai | 06/29/2020 | 1200 | 400 |
| Albert | New York | Alford | New York | 07/10/2020 | 1000 | 300 |
| Avinash | Mumbai | Mukeshi | Mumbai | 06/10/2020 | 5000 | 600 |
| Avinash | Mumbai | Mukeshi | Mumbai | 09/16/2020 | 500 | 100 |
| Avinash | Mumbai | Mukeshi | Mumbai | 06/24/2020 | 500 | 100 |

10 rows returned in 0.03 seconds Download

Bagaimana??
Query yang dijalankan lebih simpel dan anda tidak perlu mengetikkan query sepanjang tadi berulang kali, cukup gunakan view sepeti anda akan menggunakan tabel untuk menyortid kolom-kolom terkait



# 2. Sequence & Synonym

Sequence memiliki fungsi yang cukup berguna ketika anda akan membuat sebuah penomoran secara otomatis. Secara teknis sequence akan melakukan penomoran langsung ke kolom yang menjadi primary key di sebuah tabel, pada sequence anda bisa menentukan primary key dan pengisiannya juga bisa diatur sesuai dengan query yang anda siap kan.

Synonym, secara fungsi query synonym memiliki cara kerja yang cukup mirip seperti view untuk membuat pekerjaan anda dalam mengolah data lebih efektif. Intinya Synonym berperan sebagai nama alternatif yang dibuat untuk sebuah tabel, view, sequence, procedure dan semua object browser di dalam apex oracle.

# EXP:

## Creating Sequence

```
38  create sequence seq4
39      increment by 1
40      start with 200306
41      minvalue 200301
42      maxvalue 200306
43      nocache
```

Results | Explain | Describe | Saved SQL | History

Sequence created.

0.04 seconds

## Penjelasan...

Membuat Sequence, cukup ketikkan query di samping untuk membuat nama seperti contoh seq4, increment by [1] berarti setiap data yang ditambahkan ke dalam tabel akan dijumlahkan sebanyak 1, lalu start with adalah mulai dari mana data akan ditambahkan serta minvalue & maxvalue merupakan ketentuan nilai paling kecil dan paling besar dari sequence tersebut.

## Penggunaan..

```
48  insert into orders values (seq4.NEXTVAL,
49      5500,
50      900,
51      CURRENT_DATE,
52      'C00012',
53      'A011',
54      'SOD'
55  )
```

Results | Explain | Describe | Saved SQL | History

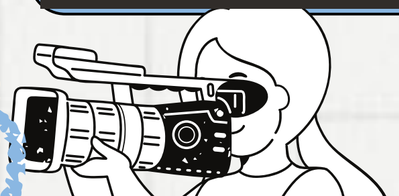
1 row(s) inserted.

0.01 seconds

## Penjelasan

Query disamping ketika dieksekusi akan menambahkan data yang ada di bawahnya dengan ketentuan dari sequence yang kita buat yaitu seq4. Ceklah tabel orders maka kita bisa melihat data yang kita tambahkan dengan ketentuan sequence tadi. Akan terjadi error bila kita menambahkan data karena seq4 memulai input data dari 200306 dan diincrement 1 lalu jika kita gunakan akan error.

| | | | | | | |
|---|---|---|---|---|---|---|
| 200306 | 5500 | 900 | 07/10/2024 | C00012 | A011 | SOD |



# EXP:

## Creating Synonym

```
103    create synonym PESANAN
104    for orders
```

Results | Explain | Describe | Sa

```
Synonym created.

0.03 seconds
```

## Penjelasan...

Pembuatan synonym itu sederhana sekali, anda hanya perlu mencari sebuah tabel, view, atau object browser lainnya dari database anda untuk dijadikan nama alternatifnya. Pada contoh disamping ada tabel orders yang dibuat nama lainnya yakni pesanan. Penggunaannya sama saja nantinya ketika kita akan menggunakan tabel orders, hanya saja anda bisa menggunakan nama lain yaitu PESANAN.

## Another example....

```
103    Create View
104        lihatKomis
105        as
106        select commission from agents
107
108    Create synonym
109        gaji for lihatKomis
```

Results | Explain | Describe | Saved SQL | Hi

```
Statement processed.

0.04 seconds
```

## Penjelasan

Query disamping adalah salah satu contoh lain dari penggunaan synonym yaitu ketika diterapkan sebagai alternatif nama dari sebuah view. Pada query ini bisa dilihat ada sebuah view yang baru dibuat dengan nama lihatKomis yang mana isinya adalah untuk melihat isi kolom commission dari tabel agents. Setelah ditentukan view apa yang akan digunakan, anda bisa membuat synonym dari view tersebut, di samping menjadikan view lihatKomis memiliki synonym gaji. Anda bisa menggunakan synonym yang telah anda buat setelahnya cukup jalankan perintah select * from seperti biasanya.

# 3.PL/SQL & Procedure

PL/SQL memiliki kepanjangan dari Procedural Language yang memiliki kegunaan menggabungkan bahasa pemrograman dengan query dari SQL. Layaknya bahasa pemrograman pada umumnya anda dapat menemukan variabel, kondisi if maupun kondisi looping pada implementasinya.

Procedure pada PL/SQL memiliki fungsi yang lebih spesifik. Kerja procedure sama layaknya PL/SQL akan adanya beberapa tingkat kode saat akan menjalankannya. Adanya DECLARE part untuk membuat variabel(opsional), bagian executeable yang akan berisi kondisi berstatement.

# EXP:

## PL/SQL

```
9   DECLARE
10      nama agents.agent_name%TYPE;
11      salary agents.commission%type;
12  BEGIN
13      nama := 'Phillip Dosen';
14      salary := 75;
15      DBMS_OUTPUT.PUT_LINE('Halo ' || nama);
16      DBMS_OUTPUT.PUT_LINE('Komisi anda saat ini berjumlah = ' || salary);
17  END
```

Results | Explain | Describe | Saved SQL | History

```
Halo Phillip Dosen
Komisi anda saat ini berjumlah = 75

Statement processed.

0.00 seconds
```

## Penjelasan...

Implementasi secara sederhana bisa dilihat pada contoh di samping. Sama halnya seperti membuat program sederhana, menambahkan sebuah variabel pada bagian declare ada nama dan salary yang mana tipe data dari kedua variabel tersebut diambil sesuai tipe data kolom yang dimaksud.
Anda bisa melanjutkan ke pembuatan nilai variabel dan pemanggilan print pada oracle dengan cara DBMS_OUTPUT.PUT_LINE dan panggilah nilai yang dibuat tadi.

## Procedure

```
87  CREATE OR REPLACE PROCEDURE showAgents(ac IN CHAR) IS
88      dataAgents agents%ROWTYPE;
89  BEGIN
90      select * into dataAgents from agents
91      where agent_code = ac;
92      DBMS_OUTPUT.PUT_LINE(
93          dataAgents.agent_code ||','||
94          dataAgents.agent_name ||','||
95          dataAgents.country
96      );
97      DBMS_OUTPUT.PUT_LINE('komis = '||dataAgents.commission);
98  END;
99  begin
100     showagents('A007');
101 end;
```

Results | Explain | Describe | Saved SQL | History

```
A007  ,Ramasundar                       ,INDIA
komis = .27

Statement processed.
```

## Penjelasan

Contoh pembuatan procedure di oracle Apex, lihatlah gambar di samping terdapat procedure yang diberi nama showAgents di dalamnya terdapat parameter ac bertipe data char. IS memiliki maksud bahwa kita akan membuat sebuah variabel dataAgents yang mana bisa memiliki tipe data yang ada di kolom-kolom tabel agents.
Mulai ke blok Begin yang memilih semua kolom dari tabel agents yang nilainya akan cocok dengan parameter ac tadi. Ketika cocok maka akan menampilkan isi variabel dataAgents yang berupa ada agent_code, agent_name, country dan commission.

# 4. Function & Cursor

Function jika disandingkan dengan Procedure itu keduanya memiliki peran yang sama-sama untuk menjalankan tugas yang spesifik, pembedanya adalah di procedure tidak memiliki nilai yang akan dikembalikan sedangkan Function harus memiliki nilai yang akan dikembalikan. Nilai yang dikembalikan ini ditandakan dengan kata “RETURN”.

Cursor merupakan sebuah alat yang disediakan untuk pemrograman SQLberguna untuk mengakses dan dapat memanipulasi hasil dari query secara baris perbaris. Jenis dari cursor ada implicit yang bisa menangani operasi DML seperti insert, update, delete dan ada cursor explicit.

# EXP:

## Function

```
1   Declare
2       nilai1 number(2);
3   FUNCTION evenOdd(nilai in number)
4   return VARCHAR2 IS
5       hasil VARCHAR2(15);
6   BEGIN
7       IF (MOD(nilai,2)=0) THEN
8           hasil := 'EVEN';
9       else
10          hasil := 'ODD';
11      END IF;
12      Return hasil;
13  END;
14  Begin
15      nilai1 := 26;
16      dbms_output.put_line(nilai1 || ' is ' || evenOdd(nilai1));
17  end;
```

Results | Explain | Describe | Saved SQL | History

```
26 is EVEN

Statement processed.
```

## Penjelasan...

Function, sama seperti procedure anda akan membuat sebuah statement logika. Gambar di samping adalah contoh penggunaan dari function. Dimulai dari pembuatan variabel dengan tipe data number dan panjang datanya 2 (bila panjangnya kurang dari 2 akan menyebabkan error). Function pasti akan ada nilai yang dikembalikannya maka bisa dilihat di samping program akan mengembalikan sebuah tipe data bertipe varchar2 dari variabel hasil yang memiliki panjang data varchar2 sepanjang 15. Lalu logika dijalankan di begin, seperti yang dilihat pada blok terakhir statement logika program diterapkan pengembalian nilai RETURN.

## Implicit Cursor

```
134 BEGIN
135     INSERT INTO pegawai (pegawai_id, firstName, lastName, divisi_id, gaji)
136     VALUES (4, 'Blade', 'Walker', 4, 3500000);
137
138     DBMS_OUTPUT.PUT_LINE('Pegawai ditambahkan.');
139 END;
```

Results | Explain | Describe | Saved SQL | History

```
Pegawai ditambahkan.

1 row(s) inserted.
```

| PEGAWAI_ID | FIRSTNAME | LASTNAME | DIVISI_ID | GAJI |
|---|---|---|---|---|
| 1 | Mark | Aoki | 4 | 5000000 |
| 2 | Steve | Aoki | 4 | 6000000 |
| 3 | Lynx | Mordor | 4 | 7000000 |
| 4 | Blade | Walker | 4 | 3500000 |

## Penjelasan

Untuk Contoh di samping menggunakan implicit Cursor yakni dengan cursor yang memanipulasi DML Insert. Ketika cursor dijalankan dan data dimasukkan maka akan menampilkan keterangan sesuai dengan print DBMS_OUTPUT.PUT_LINE-nya. Cek lagi tabel tersebut untuk melihat apa data yang dimasukkan tadi sudah terinput dengan benar.

Untuk menghapus menggunakan Implicit Cursor bisa dilihat disamping

```
143 BEGIN
144     DELETE FROM pegawai
145     WHERE pegawai_id = 4;
146
147     DBMS_OUTPUT.PUT_LINE('pegawai dipecat.');
148 END;
```

Results | Explain | Describe | Saved SQL | History

```
pegawai dipecat.

1 row(s) deleted.

0.00 seconds
```

# EXP:

## Explicit Cursor

```
105  CREATE OR REPLACE FUNCTION hitungGaji(f_divisi_id IN NUMBER) RETURN NUMBER IS
106      vTotalGaji NUMBER := 0;
107      CURSOR employee_cursor IS
108          SELECT gaji
109          FROM pegawai
110          WHERE divisi_id = f_divisi_id;
111      v_gaji pegawai.gaji%TYPE;
112  BEGIN
113      OPEN employee_cursor;
114
115      LOOP
116          FETCH employee_cursor INTO v_gaji; -- Mengambil baris berikutnya dari cursor
117          EXIT WHEN employee_cursor%NOTFOUND; -- Keluar dari loop jika tidak ada lagi baris
118
119          vTotalGaji := vTotalGaji + v_gaji; -- Menambahkan gaji ke total gaji
120      END LOOP;
121
122      CLOSE employee_cursor;
123
124      RETURN vTotalGaji; -- Mengembalikan total gaji
125  END;
```

Results | Explain | Describe | Saved SQL | History

Function created.

## Penjelasan...

Function di samping adalah function yang berfungsi untuk menghitung gaji dari keseluruhan pegawai divisi 4. Cara kerja functionnya adalah cursor akan men-select gaji dari tabel pegawai dan mencari divisi id-nya. Setelah itu akan melakukan perulangan untuk v_gaji sampai datanya sudah tidak ditemukan lagi, bila sudah habis maka vTotalGaji akan berjalan dan melakukan penambahan dari semua gaji yang sudah diloop tadi. Akhirnya akan mengembalikan nilai (RETURN) dari vTotalGaji

Contoh hasil(Dilakukan pengecekan terlebih dahulu):

```
134  select firstname, gaji from pegawai
```

Results | Explain | Describe | Saved SQL | History

| FIRSTNAME | GAJI |
|---|---|
| Mark | 5000000 |
| Blade | 3500000 |
| Steve | 6000000 |
| Lynx | 7000000 |

4 rows returned in 0.01 seconds Download

```
127  DECLARE
128      vTotalGaji NUMBER;
129  BEGIN
130      vTotalGaji := get_total_salary(4);
131      DBMS_OUTPUT.PUT_LINE('Total salary in department 4: ' || vTotalGaji);
132  END;
```

Results | Explain | Describe | Saved SQL | History

Total salary in department 4: 21500000

Statement processed.

# 5. Exception & Trigger

Exception digunakan di PL/SQL untuk menangani atau menangkap sebuah kesalahan yang terjadi saat melakukan eksekusi bagian blok kode PL/SQL. Adanya Exception akan memudahkan anda dan mengelola kesalahan yang terjadi dengan cara yang terkontrol.

Trigger merupakan sebuah program PL/SQL yang akan jalan secara otomatis ketika terjadi sesuatu kepada sebuah tabel atau view. Trigger penggunaannya dengan tujuan untuk pemeliharaan integritas data, menjalankan logika bisnis dan penanganan perubahan data otomatis.

# EXP:

## Exception

```
180 declare
181     a_code agents.agent_code%type := 'A013';
182     a_name agents.agent_name%type;
183 begin
184     select  agent_code,agent_name Into
185     a_code, a_name from agents where
186     agent_code = a_code;
187     DBMS_OUTPUT.PUT_LINE ('Agent Code: '|| a_code);
188     DBMS_OUTPUT.PUT_LINE ('Agent Name: '|| a_name);
189 EXCEPTION
190     WHEN no_data_found THEN
191     DBMS_OUTPUT.PUT_LINE ('NO DATA FOUND!');
192     WHEN others THEN
193     DBMS_OUTPUT.PUT_LINE ('ERROR!');
194 END
195
```

Results | Explain | Describe | Saved SQL | History

```
Agent Code: A013
Agent Name: Badu Atai

Statement processed.
```

## Penjelasan...

Query Exception di samping berguna untuk memanggil sebuah print bila ada data yang tidak ditemukan atau error. Namun dalam kasus ini data ditemukan maka akan menampilkan info dari agent code & name.

## If there is no data found..

```
180 declare
181     a_code agents.agent_code%type := '00009';
182     a_name agents.agent_name%type;
183 begin
184     select  agent_code,agent_name Into
185     a_code, a_name from agents where
186     agent_code = a_code;
187     DBMS_OUTPUT.PUT_LINE ('Agent Code: '|| a_code);
188     DBMS_OUTPUT.PUT_LINE ('Agent Name: '|| a_name);
189 EXCEPTION
190     WHEN no_data_found THEN
191     DBMS_OUTPUT.PUT_LINE ('NO DATA FOUND!');
192     WHEN others THEN
193     DBMS_OUTPUT.PUT_LINE ('ERROR!');
194 END
195
```

Results | Explain | Describe | Saved SQL | History

```
NO DATA FOUND!

Statement processed.
```

# EXP:

## Trigger

```
11  CREATE OR REPLACE TRIGGER tINSCust
12  AFTER INSERT ON customer
13  FOR EACH ROW
14  BEGIN
15      dbms_output.put_line('Trigger tINSCust terpicu. Info:');
16      dbms_output.put_line('Cust Code: ' || :NEW.CUST_CODE);
17      dbms_output.put_line('Cust Name: ' || :NEW.CUST_NAME);
18  END;
```

Results | Explain | Describe | Saved SQL | History

```
Trigger created.
```

## Penjelasan...

Fungsi utama di trigger ini adalah ketika dia melakukan sebuah eksekusi dari memanipulasi data DML maka akan menampilkan sebuah trigger atau memicu sebuah statement yang telah disediakan di BEGIN tersebut. Contoh eksekusi ada di bawah:

## When triggered:

```
20  INSERT all
21  into customer values ('test', 'nopel', 'test', 'test', 'test', 2, 0.1, 12000.00, 91000.00, 111000.00, 'saucy', 'A012')
22  select * from dual
```

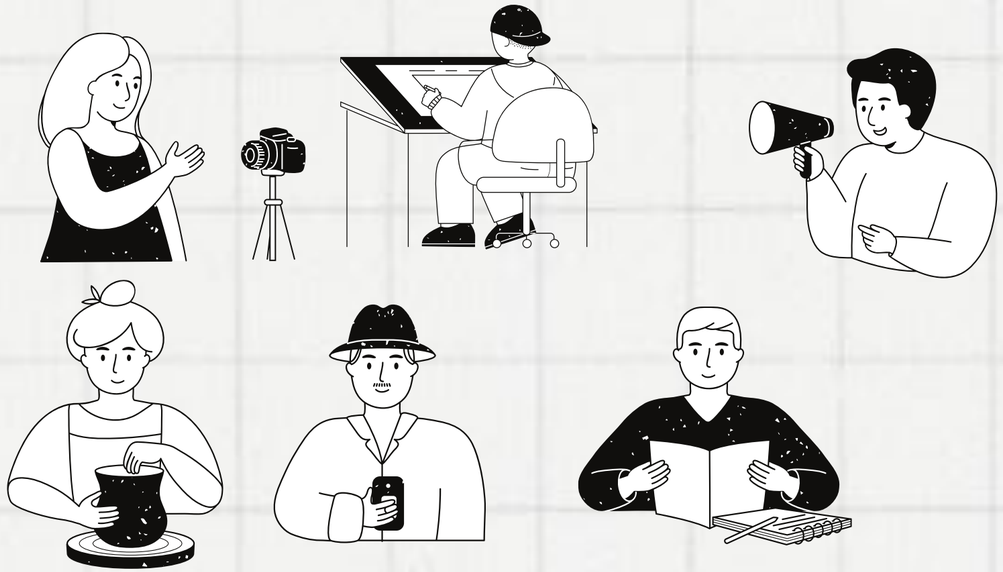
Results | Explain | Describe | Saved SQL | History

```
Trigger tINSCust terpicu. Info:
Cust Code: test
Cust Name: nopel

1 row(s) inserted.
```

# BDL Setelahnya...

Telah berakhir pembahasan dari beberapa materi untuk presentasi UAS BDL ini. Selama melihat dan membaca pembahasannya secara seksama kita bisa mendapatkan pemahaman lebih lagi fitur dan kegunaan dari Apex oracle dan ragam query yang tersedia untuk SQL yang dapat mempermudah kita untuk melakukan pengelolaan data.



# Terimakasih Banyak!