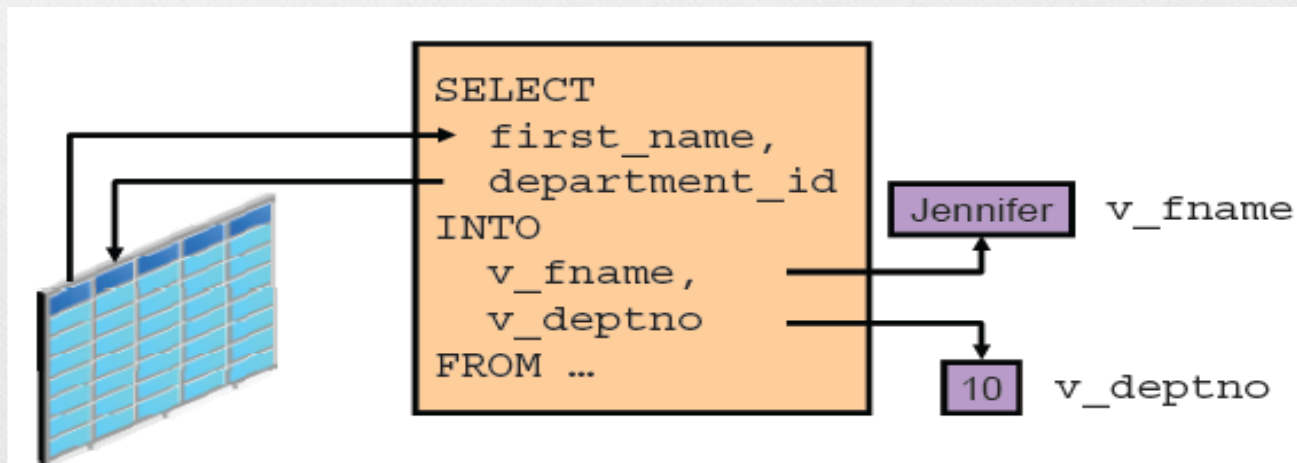


Variabile in PL/SQL

Utilizarea variabilelor

- Pentru stocarea temporara a datelor
- Manipularea informatiilor stocate
- (Re)Utilizarea in diferite contexte



Denumirea variabilelor

- Un nume de variabila:
 - Trebuie sa inceapa cu o litera
 - Poate sa includa litere si cifre
 - Poate sa includa caracterele speciale \$, _,#
 - Nu poate sa aiba o lungime mai mare de 30 de caractere
 - Nu poate sa includa cuvinte rezervate (ex: join, select etc)

Utilizarea variabilelor

- Variabilele sunt:
 - Declarate si optional initializate in sectiunea *DECLARE*.
 - Utilizate pentru a stoca valori in sectiunea executabila
 - Folosite drept parametrii ale subprogramelor
 - Utilizate drept valoare atribuita doar daca in prealabil au fost definite

Declararea si initializarea variabilelor

Sintaxa:

```
identifier [CONSTANT] datatype [NOT NULL]  
    [:= | DEFAULT expr];
```

Exemple:

```
DECLARE  
    v_hiredate      DATE;  
    v_location      VARCHAR2(13) := 'Atlanta';  
    v_deptno        NUMBER(2) NOT NULL := 10;  
    c_comm          CONSTANT NUMBER := 1400;
```

Declararea si initializarea variabilelor

Observatie:

```
DECLARE
    v_myName  VARCHAR(20);
BEGIN
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName );
    v_myName  := 'John';
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName );
END;
/
```

```
DECLARE
    v_myName VARCHAR2(20) := 'John';
BEGIN
    v_myName := 'Steven';
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
END;
/
```

Delimitatori in initializarea variabilelor

Exemplu:

```
DECLARE
    v_event VARCHAR2(15);
BEGIN
    v_event := q'!Father's day!';
    DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is :
    || v_event );
    v_event := q'[Mother's day]';
    DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is :
    || v_event );
END;
/
```

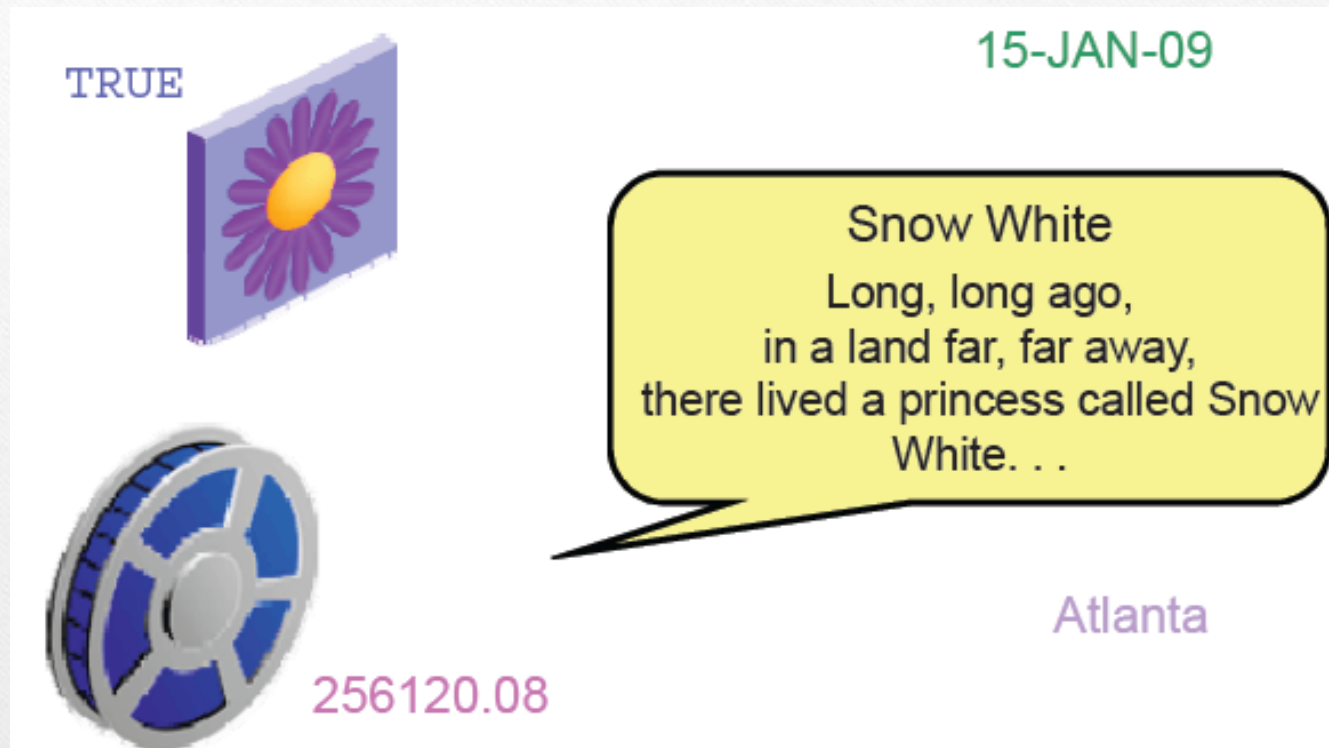
Output:

```
3rd Sunday in June is :Father's day
2nd Sunday in May is :Mother's day
```


Clasificarea variabilelor

- Variabile PL/SQL:
 - Scalare
 - Pointer
 - Colectii
 - Obiecte externe (adresa obiectelor LOB)
- Variabile de legatura (SQL – PL/SQL)
 - Utilizate pentru a pasa informatie intre cele 2 medii

Exemple tipuri de variabile



Info:

BOOLEAN, DATE, BLOB, VARCHAR2, CLOB, NUMBER, BFILE

Declararea variabilelor in PL/SQL

- Alegeti denumiri sugestive (x vs v_emp_id)
- Definiti cate o variabila pe linie
- Initializati variabilele care trebuie sa fie NOT NULL sau sunt declarate cu ajutorul CONSTANT
- Folositi pentru initializare “:=“ sau DEFAULT

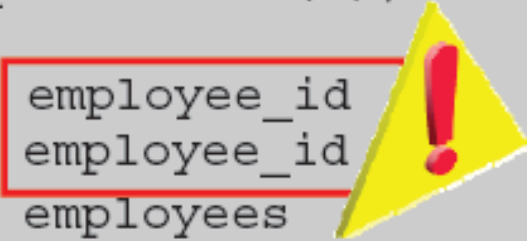
```
v_myName VARCHAR2 (20) := 'John';
```

```
v_myName VARCHAR2 (20) DEFAULT 'John';
```


Declararea variabilelor in PL/SQL

- Evitati sa denumiti variabilele cu numele coloanelor care vor furniza valorile

```
DECLARE
  employee_id NUMBER(6);
BEGIN
  SELECT
  INTO
  FROM
  WHERE
  END;
```



- Utilizati optiunea NOT NULL atunci cand o variabila trebuie sa aiba o valoare

```
pincode VARCHAR2(15) NOT NULL := 'Oxford';
```

Conventii privind denumirea variabilelor

PL/SQL Structure	Convention	Example
Variable	<code>v_variable_name</code>	<code>v_rate</code>
Constant	<code>c_constant_name</code>	<code>c_rate</code>
Subprogram parameter	<code>p_parameter_name</code>	<code>p_id</code>
Bind (host) variable	<code>b_bind_name</code>	<code>b_salary</code>
Cursor	<code>cur_cursor_name</code>	<code>cur_emp</code>
Record	<code>rec_record_name</code>	<code>rec_emp</code>
Type	<code>type_name_type</code>	<code>ename_table_type</code>
Exception	<code>e_exception_name</code>	<code>e_products_invalid</code>
File handle	<code>f_file_handle_name</code>	<code>f_file</code>

Variabile de tip scalar

```
DECLARE
```

```
  v_emp_job          VARCHAR2(9) ;
```

```
  v_count_loop       BINARY_INTEGER := 0;
```

```
  v_dept_total_sal    NUMBER(9,2) := 0;
```

```
  v_orderdate         DATE := SYSDATE + 7;
```

```
  c_tax_rate          CONSTANT NUMBER(3,2) := 8.25;
```

```
  v_valid             BOOLEAN NOT NULL := TRUE;
```

Declararea cu ajutorul %TYPE

- In unele situatii se doreste definirea unei variabile care sa permita intotdeauna salvarea datelor care se regasesc intr-o coloana a unui tabel (indiferent de modificarile pe care acesta le poate suferi)
- Se foloseste %type pentru:
 - definirea unei variabile de tipul `tabel.coloana%type;`
 - definirea unei variabile de tipul altei variabile `variabila%type;`

Declararea cu ajutorul %TYPE

- Sintaxa:

```
identifier      table.column_name%TYPE;
```

- Exemple:

```
...  
  v_emp_lname      employees.last_name%TYPE;  
...
```

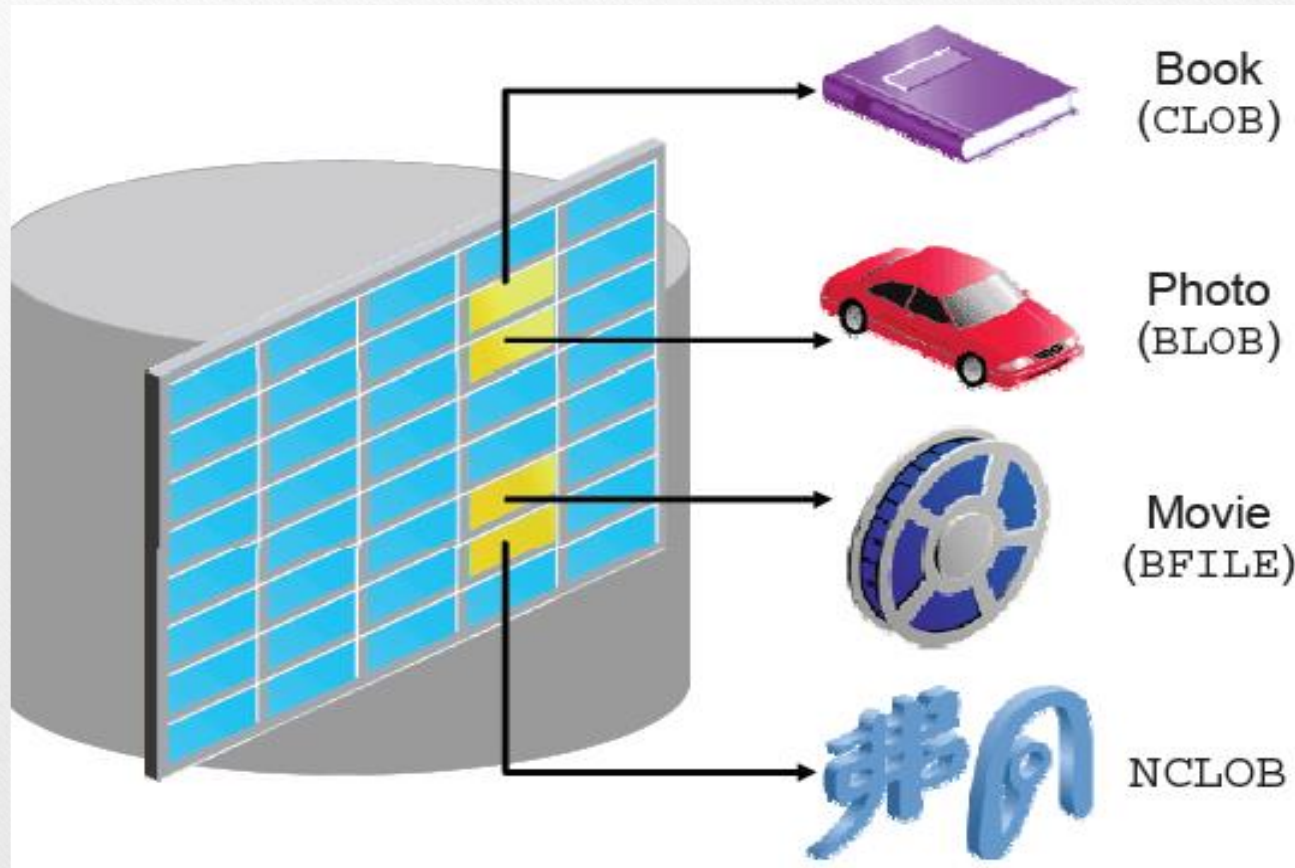
```
...  
  v_balance        NUMBER(7,2);  
  v_min_balance    v_balance%TYPE := 1000;  
...
```

Variabile de tip boolean in PL/SQL

- Unei variabile de tip boolean ii pot fi atribuite doar valorile TRUE, FALSE si NULL
- Pot fi folosite impreuna cu operatorii logici AND si OR
- Expresiile aritmetice, cele care prelucreaza tipuri de date caracter si date pot intoarce expresii de tip boolean


```
DECLARE
    flag BOOLEAN := FALSE;
BEGIN
    flag := TRUE;
END;
```


Tipuri de date LOB



Tipuri de date compuse

PL/SQL Record:

TRUE	23-DEC-98	ATLANTA	
------	-----------	---------	---

PL/SQL Collections:

1	SMITH	1	5000
2	JONES	2	2345
3	NANCY	3	12
4	TIM	4	3456

Diagram illustrating PL/SQL Collections:

The left collection is a **VARCHAR2** array (indexed 1 to 4) and a **PLS_INTEGER** array (indexed 1 to 4).

The right collection is a **NUMBER** array (indexed 1 to 4) and a **PLS_INTEGER** array (indexed 1 to 4).

Variabile de legatura

- Generalitati
 - Pot fi definite de mediu (environment)
 - Sunt denumite variabile host
 - NU sunt variabile globale
 - Sunt definite in SQLDeveloper cu ajutorul VARIABLE
 - Pot fi utilizate atat in SQL, cat si in PL/SQL
 - Pot fi accesate dupa terminarea blocului PL/SQL in care sunt utilizate
 - Sunt prefixate de “:”
 - Sunt afisate in SQL cu ajutorul lui PRINT

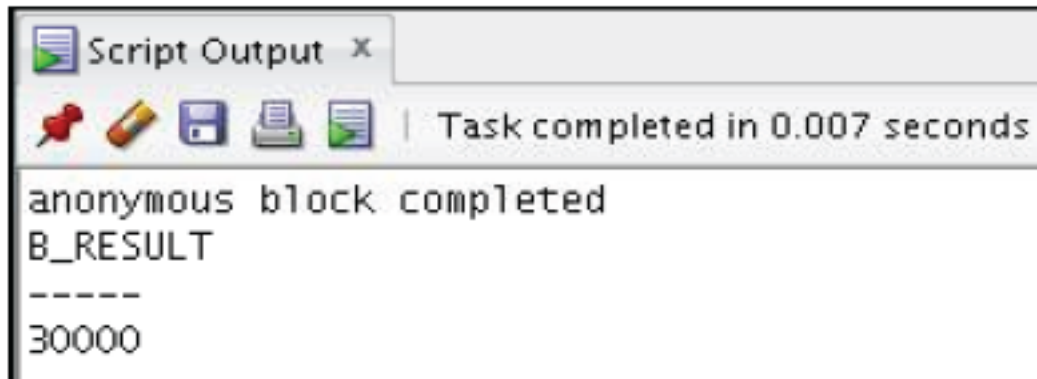
```
VARIABLE return_code NUMBER
```

```
VARIABLE return_msg VARCHAR2 (30)
```


Variabile de legatura

- Exemplu:

```
VARIABLE b_result NUMBER
BEGIN
  SELECT (SALARY*12) + NVL(COMMISSION_PCT,0) INTO :b_result
  FROM employees WHERE employee_id = 144;
END;
/
PRINT b_result
```

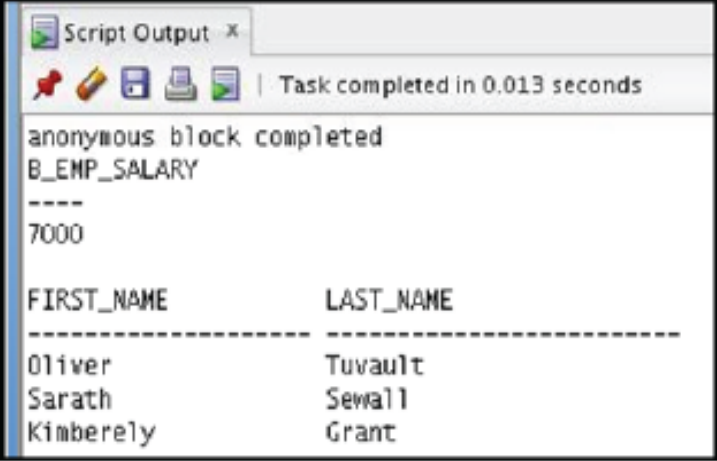


Variabile de legatura

- Exemplu:

```
VARIABLE b_emp_salary NUMBER
BEGIN
    SELECT salary INTO :b_emp_salary
    FROM employees WHERE employee_id = 178;
END;
/
PRINT b_emp_salary
SELECT first_name, last_name
FROM employees
WHERE salary=:b_emp_salary;
```

Output →



Script Output - x

Task completed in 0.013 seconds

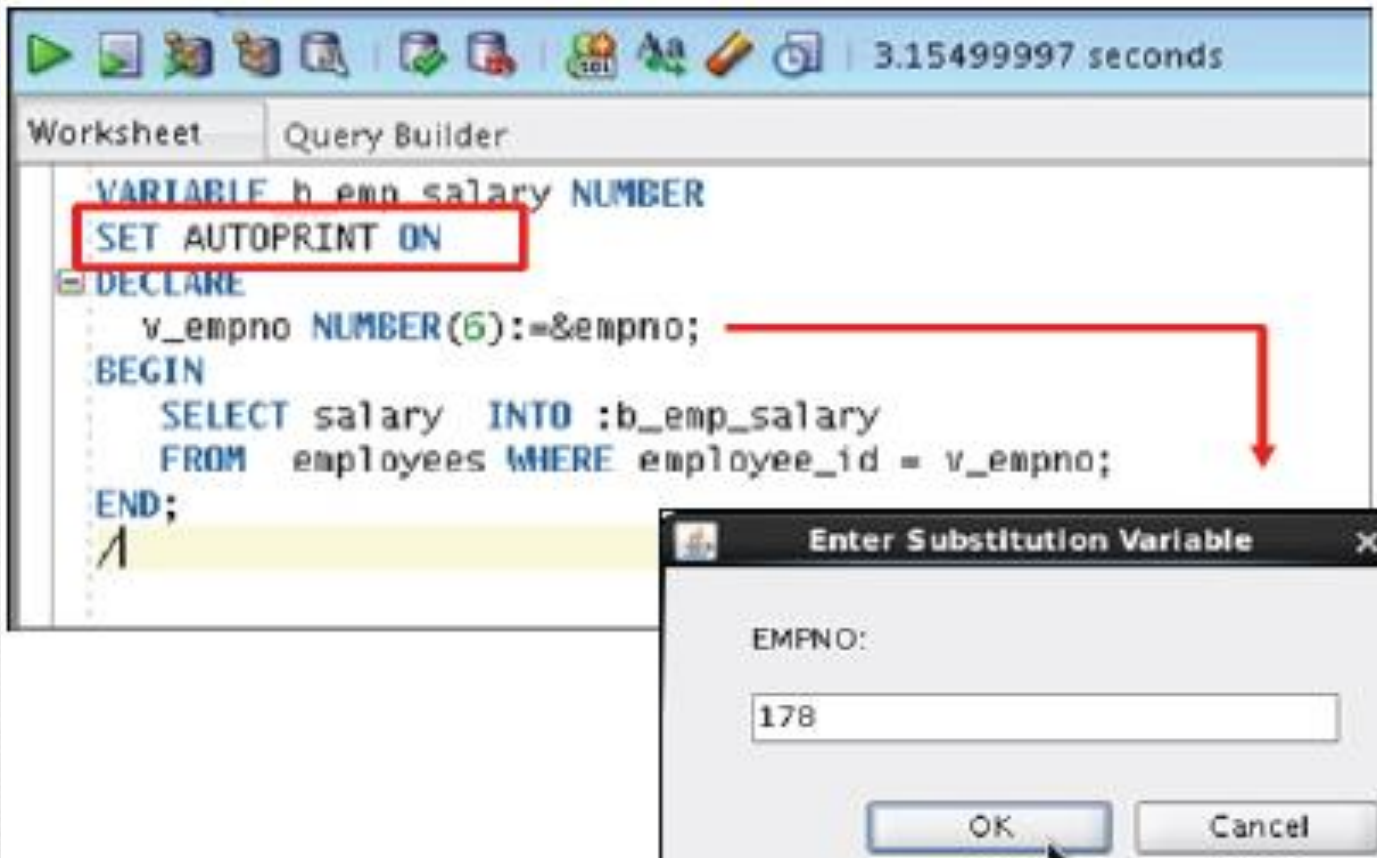
anonymous block completed
B_EMP_SALARY

7000

FIRST_NAME	LAST_NAME
Oliver	Tuvault
Sarath	Sewall
Kimberely	Grant

Variabile de legatura - AUTOPRINT

- Exemplu:



Variabile case sensitive?

- In situatii cu totul speciale este necesara definirea unor variabile case sensitive sau care sa contina spatii:

```
"begin date" DATE;
```

```
"end date" DATE;
```

```
"exc thrown" BOOLEAN DEFAULT TRUE;
```

- Aceste variabile trebuie sa fie intotdeauna referite cu ajutorul "*var*"
- NU se recomanda (utilizarea este greoaie)

Formatarea unui bloc anonim

- Selectati butonul Format (CTRL+F7) pentru a formata blocul:

The diagram illustrates the process of formatting an anonymous SQL block. It consists of three main components:

- Initial Code (Left):** A code editor window showing the following SQL code:

```
DECLARE
v_fname VARCHAR2(20);
BEGIN
select first_name into v_
WHERE employee_id=100;
END;
```

A green circle with the number '1' is placed over the word 'v_'.
A red arrow points from this code to the 'Format' menu item in the context menu.
- Context Menu (Middle):** A right-click context menu is displayed with the following options:
 - Cut (Ctrl-X)
 - Copy (Ctrl-C)
 - Paste (Ctrl-V)
 - Select All (Ctrl-A)
 - Debug (Ctrl+Shift-F10)
 - Refactoring
 - Format (Ctrl-F7)** (highlighted with a green circle '2')
 - Advanced Format (Ctrl+Shift-F)
 - Code Template
 - Popup Describe (Shift-F4)
 - Open Declaration
- Formatted Code (Right):** The same code editor window after formatting, showing the code with proper indentation:

```
DECLARE
  v_fname VARCHAR2(20);
BEGIN
  SELECT first_name
  INTO v_fname
  FROM employees
  WHERE employee_id = 100;
END;
```

A green circle with the number '3' is placed over the word 'v_fname'.

Comentarea codului

- Prefixarea liniei cu “--”
- Plasarea blocului intre simbolurile /* si */

```
DECLARE
...
v_annual_sal NUMBER (9,2);
BEGIN
/* Compute the annual salary based on the
   monthly salary input from the user */
v_annual_sal := monthly_sal * 12;
--The following line displays the annual salary
DBMS_OUTPUT.PUT_LINE(v_annual_sal);
END;
```


Funcții SQL in PL/SQL

- Intr-un bloc PL/SQL sunt disponibile toate funcțiile single-row
- NU sunt disponibile funcțiile grup și funcția DECODE

```
v_desc_size INTEGER(5);  
v_prod_description VARCHAR2(70):='You can use this  
product with your radios for higher frequency';  
  
-- get the length of the string in prod_description  
v_desc_size:= LENGTH(v_prod_description);
```

```
v_tenure:= MONTHS_BETWEEN (CURRENT_DATE, v_hiredate);
```

Utilizarea secventelor in PL/SQL

1. Incepand cu 11g este posibila utilizarea secventelor intr-un bloc:

```
DECLARE
    v_new_id NUMBER;
BEGIN
    v_new_id := my_seq.NEXTVAL;
END;
```

2. Inainte de 11g:

```
DECLARE
    v_new_id NUMBER;
BEGIN
    SELECT my_seq.NEXTVAL INTO v_new_id FROM Dual;
END;
```

Conversia tipurilor de date (implicite, explicite)

```
-- implicit data type conversion  
v_date_of_joining DATE:= '02-Feb-2000';
```

```
-- error in data type conversion  
v_date_of_joining DATE:= 'February 02,2000';
```

```
-- explicit data type conversion  
v_date_of_joining DATE:= TO_DATE('February  
02,2000','Month DD, YYYY');
```

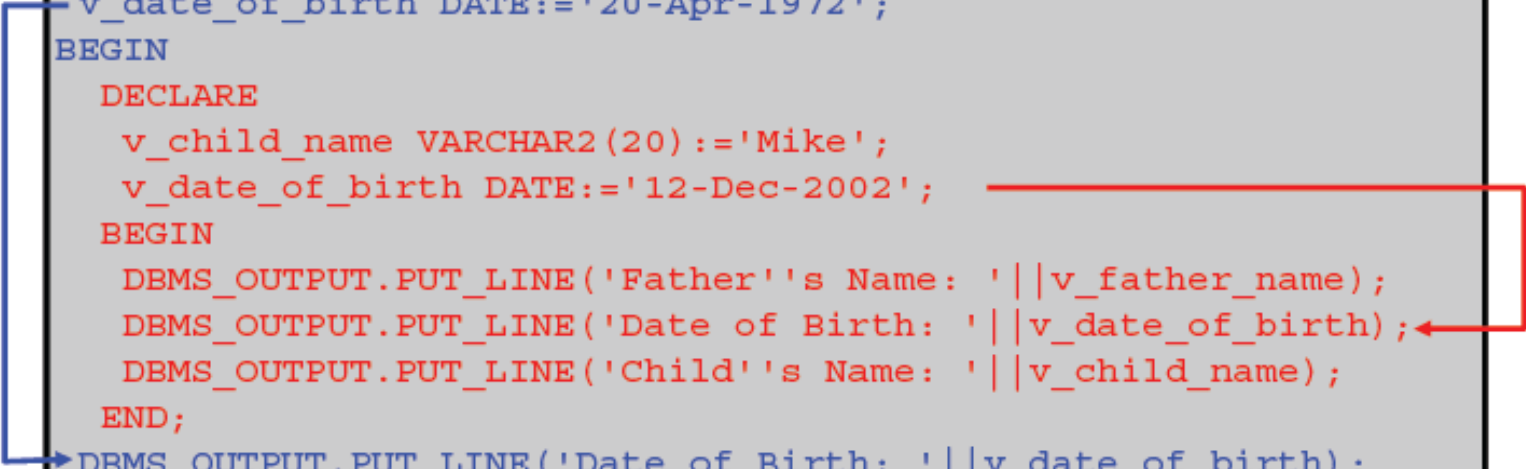

Utilizarea variabilelor in cadrul blocurilor imbricate

```
DECLARE
  v_outer_variable VARCHAR2(20):='GLOBAL VARIABLE';
BEGIN
  DECLARE
    v_inner_variable VARCHAR2(20):='LOCAL VARIABLE';
  BEGIN
    DBMS_OUTPUT.PUT_LINE(v_inner_variable);
    DBMS_OUTPUT.PUT_LINE(v_outer_variable);
  END;
  DBMS_OUTPUT.PUT_LINE(v_outer_variable);
END;
```

Output: LOCAL VARIABLE
GLOBAL VARIABLE
GLOBAL VARIABLE

Utilizarea variabilelor in cadrul blocurilor imbricate

```
DECLARE
  v_father_name VARCHAR2(20):='Patrick';
  v_date_of_birth DATE:='20-Apr-1972';
BEGIN
  DECLARE
    v_child_name VARCHAR2(20):='Mike';
    v_date_of_birth DATE:='12-Dec-2002';
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Father's Name: ' || v_father_name);
    DBMS_OUTPUT.PUT_LINE('Date of Birth: ' || v_date_of_birth);
    DBMS_OUTPUT.PUT_LINE('Child's Name: ' || v_child_name);
  END;
  DBMS_OUTPUT.PUT_LINE('Date of Birth: ' || v_date_of_birth);
END;
/
```

A diagram illustrating nested block scopes. A blue arrow points from the 'BEGIN' line of the outer block to the 'BEGIN' line of the inner block. A red arrow points from the 'END;' line of the inner block to the 'END;' line of the outer block. Another red arrow points from the 'END;' line of the inner block to the 'DBMS_OUTPUT.PUT_LINE' statement in the outer block, indicating that the inner block's scope ends before the outer block's execution continues.

Exercitii

Determinati valorile variabilelor v_message, v_total_comp, v_comm, outer.v_comm la pozitiile indicate:

```
BEGIN <<outer>>
DECLARE
  v_sal      NUMBER(7,2) := 60000;
  v_comm     NUMBER(7,2) := v_sal * 0.20;
  v_message  VARCHAR2(255) := ' eligible for commission';
BEGIN
  DECLARE
    v_sal      NUMBER(7,2) := 50000;
    v_comm     NUMBER(7,2) := 0;
    v_total_comp NUMBER(7,2) := v_sal + v_comm;
  BEGIN
    → v_message := 'CLERK not' || v_message;
      outer.v_comm := v_sal * 0.30;
  END;
→ v_message := 'SALESMAN' || v_message;
END;
END outer;
/
```