Database Programming with PL/SQL

2-4: Using Scalar Data Types

Practice Activities

..Atributul %TYPE

daca eu am o coloana in tabel, gen first\_name care ii VARCHAR2(20)

atunci eu pot zice

DECLARE

element VARCHAR2(20)

sau

element employees.first\_name%TYPE si face match la tip

poate face match ori la database column type, ori la tipul oricarei alte variabile

sintaxa:

numeTabel.numeColoana%TYPE sau numeVariabila%TYPE

Vocabulary

|  |  |
| --- | --- |
| boolean | A datatype that stores one of the three possible values used for  logical calculations: TRUE, FALSE, or NULL. |
| %TYPE | Attribute used to declare a variable according to another  previously declared variable or database column. |

1. Declarations:

A. Which of the following variable declarations are valid?

|  |  |  |
| --- | --- | --- |
|  | Declaration | Valid (ja/nein) |
| a | number\_of\_students PLS\_INTEGER; | ja |
| b | STUDENT\_NAME VARCHAR(10)=Johnson; | nein(unde-s apostroafele) |
| c | stu\_per\_class CONSTANT NUMBER; | nein(stu\_per\_class must contain an initialization assignment) |
| d | Tomorrow DATE:=SYSDATE+1; | ja |

B. For the invalid declarations above, describe why they are invalid.

Write an anonymous block in which you declare and print (on the screen) each of the variables in 1A above, correcting the invalid declarations and adding information as needed

DECLARE

number\_of\_students PLS\_INTEGER;

stu\_per\_class CONSTANT NUMBER:=0;

STUDENT\_NAME VARCHAR(10):='Johnson';

Tomorrow DATE:=SYSDATE+1;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(number\_of\_students);

DBMS\_OUTPUT.PUT\_LINE(stu\_per\_class);

DBMS\_OUTPUT.PUT\_LINE(STUDENT\_NAME);

DBMS\_OUTPUT.PUT\_LINE(Tomorrow);

END;

2. Evaluate the variables in the following code. Answer the following questions about each variable.

Is it named well? Why or why not? If it is not named well, what would be a better name and why?

DECLARE

country\_name VARCHAR2(50); ii ok

median\_age nu are tip

NUMBER(6, 2);nu are identificator

BEGIN

SELECT country\_name, median\_age INTO country\_name, median\_age

FROM countries

WHERE country\_name = 'Japan';

DBMS\_OUTPUT.PUT\_LINE('The median age in '|| country\_name || ' is '

|| median\_age || '.');

END;

3. Change the declarations in #2 above so they use the %TYPE attribute.

country\_name countries.country\_name%TYPE;

median\_age countries.median\_age%TYPE

BEGIN

SELECT country\_name, median\_age INTO country\_name, median\_age

FROM countries

WHERE country\_name = 'Japan';

DBMS\_OUTPUT.PUT\_LINE('The median age in '|| country\_name || ' is '

|| median\_age || '.');

END;

4. In your own words, describe why using the %TYPE attribute is better than hard-coding data types.

Ne scapa de erori gen mystiping tipuri standard

Can you explain how you could run into problems in the future by hard-coding the data types of

the country\_name and median\_age variables in question 2?

daca de exemplu am country\_name si declar sa fie NUMBER in loc de VARCHAR2, atunci cand se copiaza din SELECT in country\_name o sa fie conflict de tipuri

5. Create the following anonymous block:

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello World');

END;

A. Add a declarative section to this PL/SQL block. In the declarative section, declare the following

variables:

•A variable named TODAY of datatype DATE. Initialize TODAY with SYSDATE.

•A variable named TOMORROW with the same datatype as TODAY. Use the %TYPE attribute to declare this variable.

DECLARE

today DATE := SYSDATE;

tomorrow today%TYPE;

B. In the executable section, initialize the TOMORROW variable with an expression that

calculates tomorrow’s date (add 1 to the value in TODAY). Print the value of TODAY and

TOMORROW after printing ‘Hello World’.

BEGIN

tomorrow:=today+1;

DBMS\_OUTPUT.PUT\_LINE(‘Hello World ’||today||’ ‘||tomorrow);

END;