Tehnici de programare cu baze de date

#2 PL/SQL
Variabile şi tipuri de date

Adrian Runceanu
www.runceanu.ro/adrian

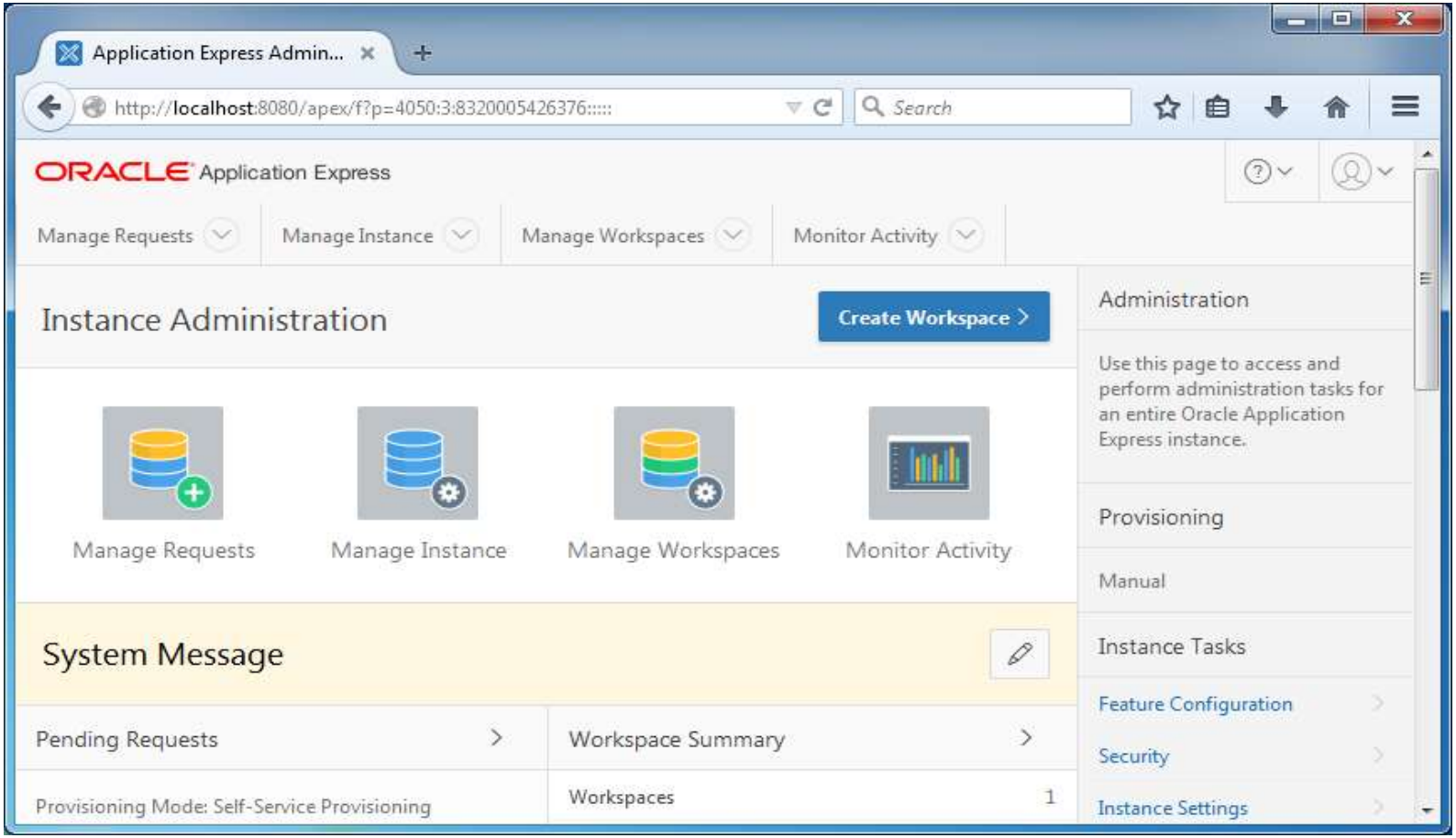
2022

Curs 2

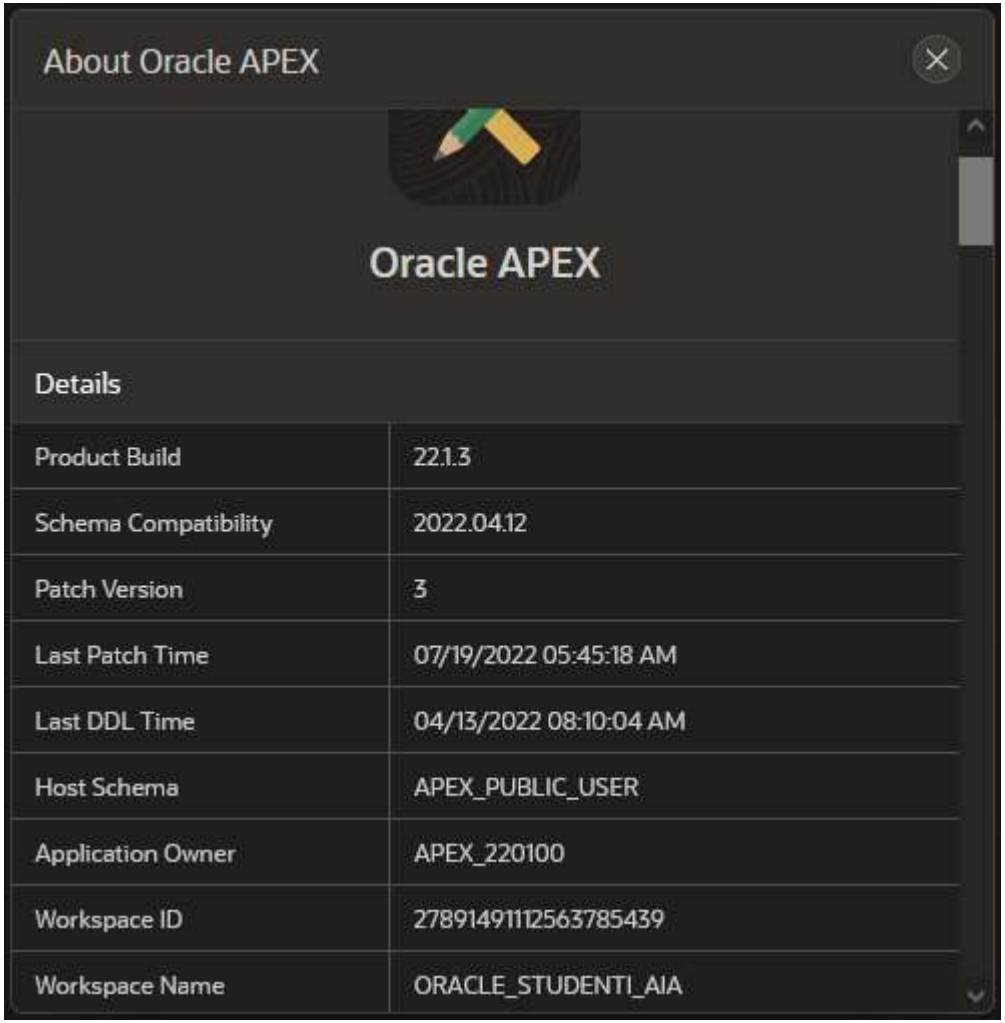
Variabile şi tipuri de date în PL/SQL

Cuprins

Oracle APEX 22.1.3 este o aplicatie web bazata pe un browser ce ofera componentele mediului de lucru SQL si PL/SQL

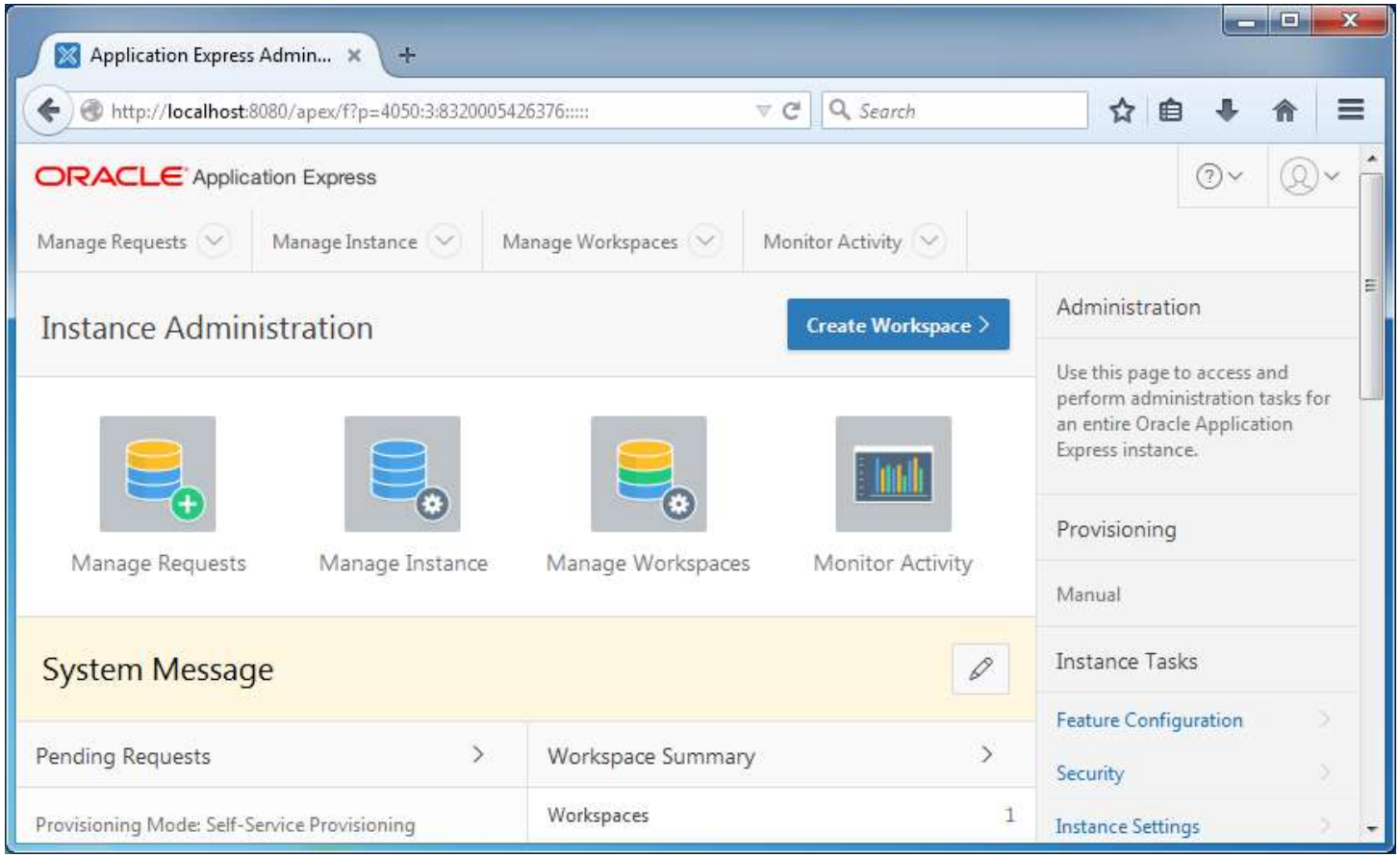


Oracle APEX 22.1.3 este o aplicatie web bazata pe un browser ce ofera componentele mediului de lucru SQL si PL/SQL



Cand va logati la Oracle Application Express si selectati SQL Workshop puteti alege sa folositi:

1. Optiunea SQL Commands – pentru a folosi editorul de comenzi SQL
2. Optiunea SQL Script – pentru a lucra cu editorul de scripturi



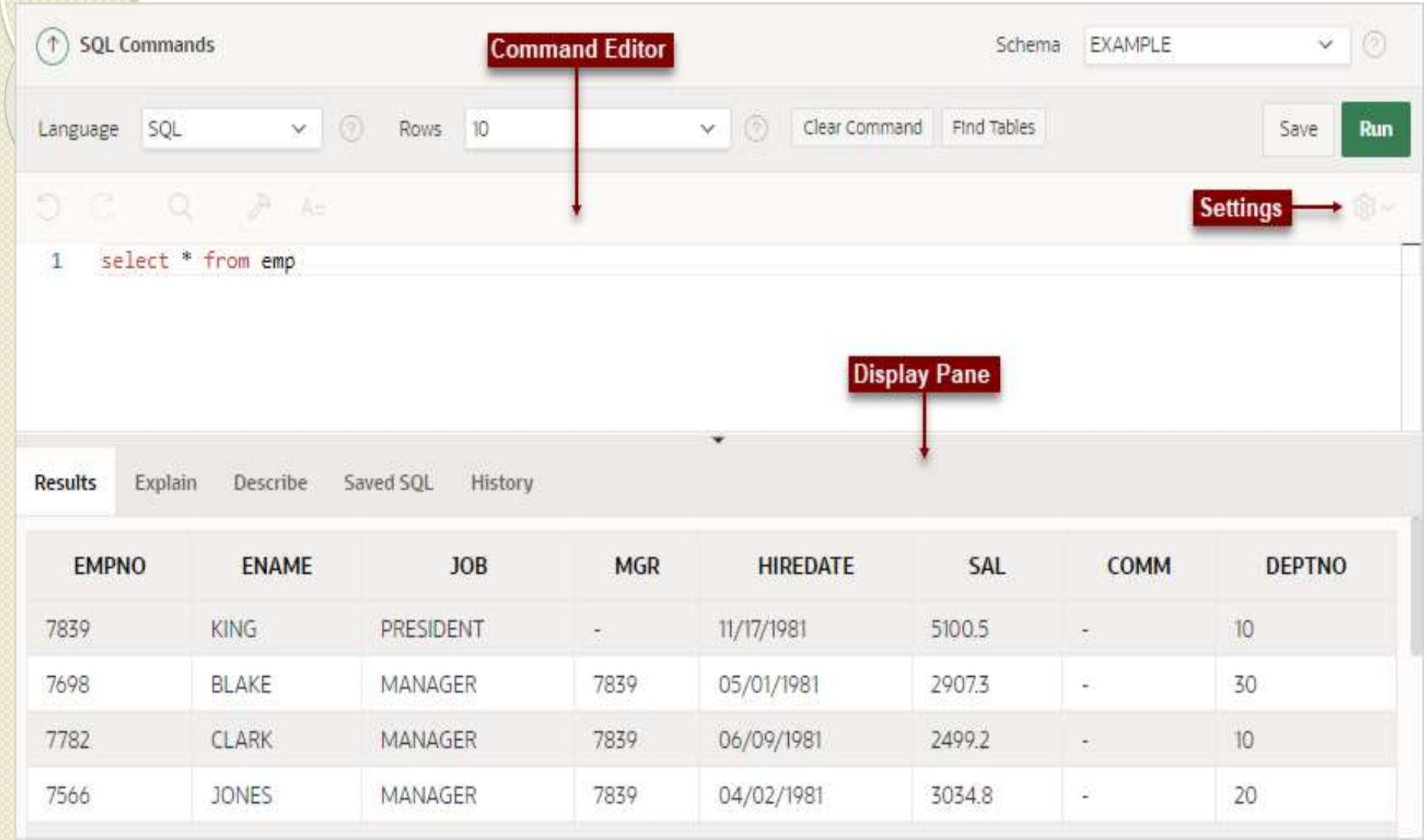
Informatii utile despre APEX:

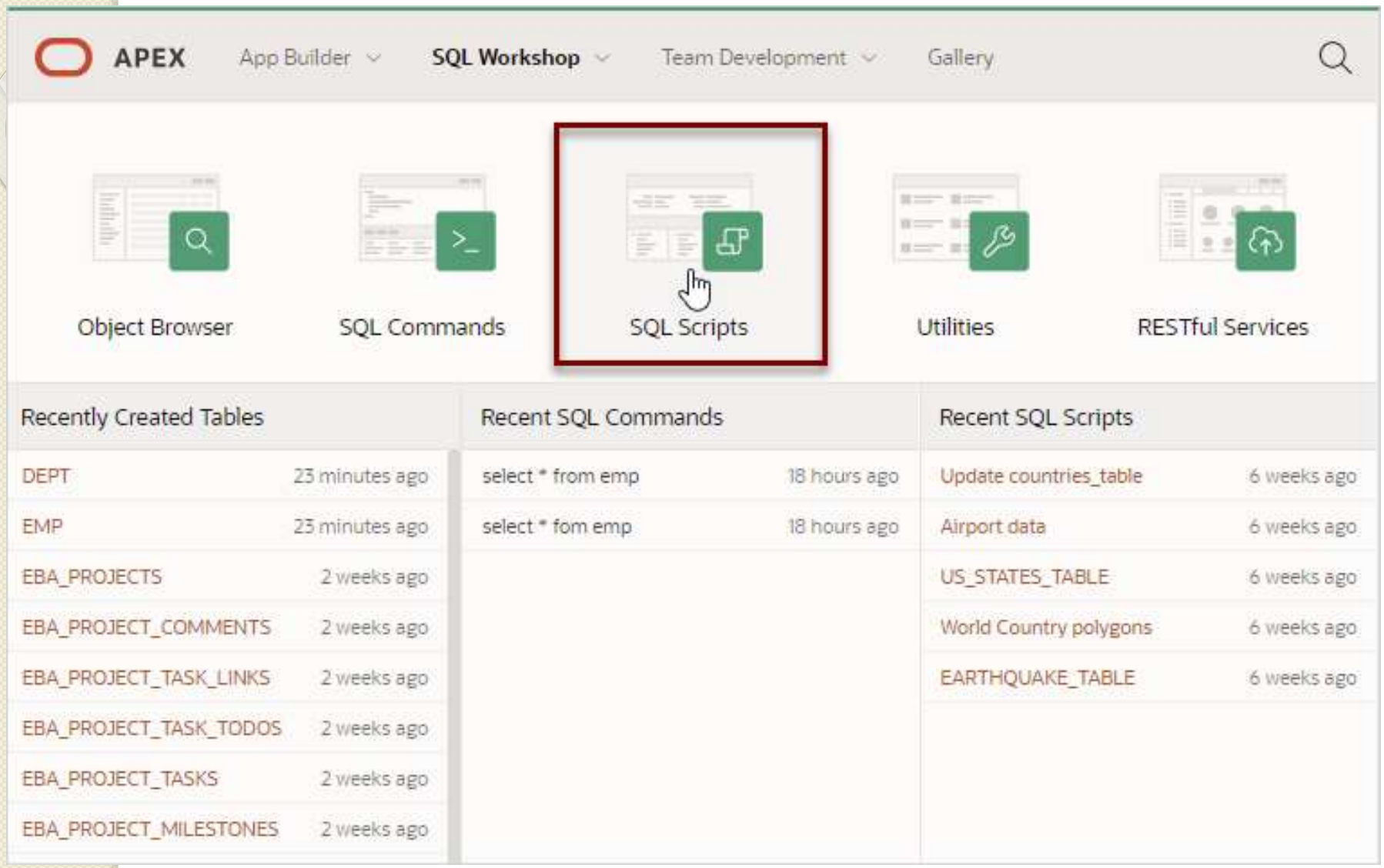
1. Oracle APEX Tutorial for Beginners (APEX 5.0):
http://o7planning.org/en/10345/oracle-apex-tutorial-for-beginners

2.http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/devdays2012/apexp1_lab/apexp1_lab.html

Oracle APEX Release 22.1
3.https://docs.oracle.com/en/database/oracle/apex/22.1/

- Se poate folosi SQL Commands pentru a introduce si executa o singura instructiune **SQL** sau un singur bloc **PL/SQL**.
- Un *script SQL poate contine una sau mai multe instructiuni **SQL**, unul sau mai multe blocuri **PL/SQL***.
- In aceasta situatie se foloseste SQL Scripts.





Instructiunea DBMS_OUTPUT.PUT_LINE

- Instructiunea **DBMS_OUTPUT.PUT_LINE** este foarte utilizata deoarece ne permite sa afisam rezultatele pentru a verifica daca blocurile ruleaza corect.

- Putem afisa:

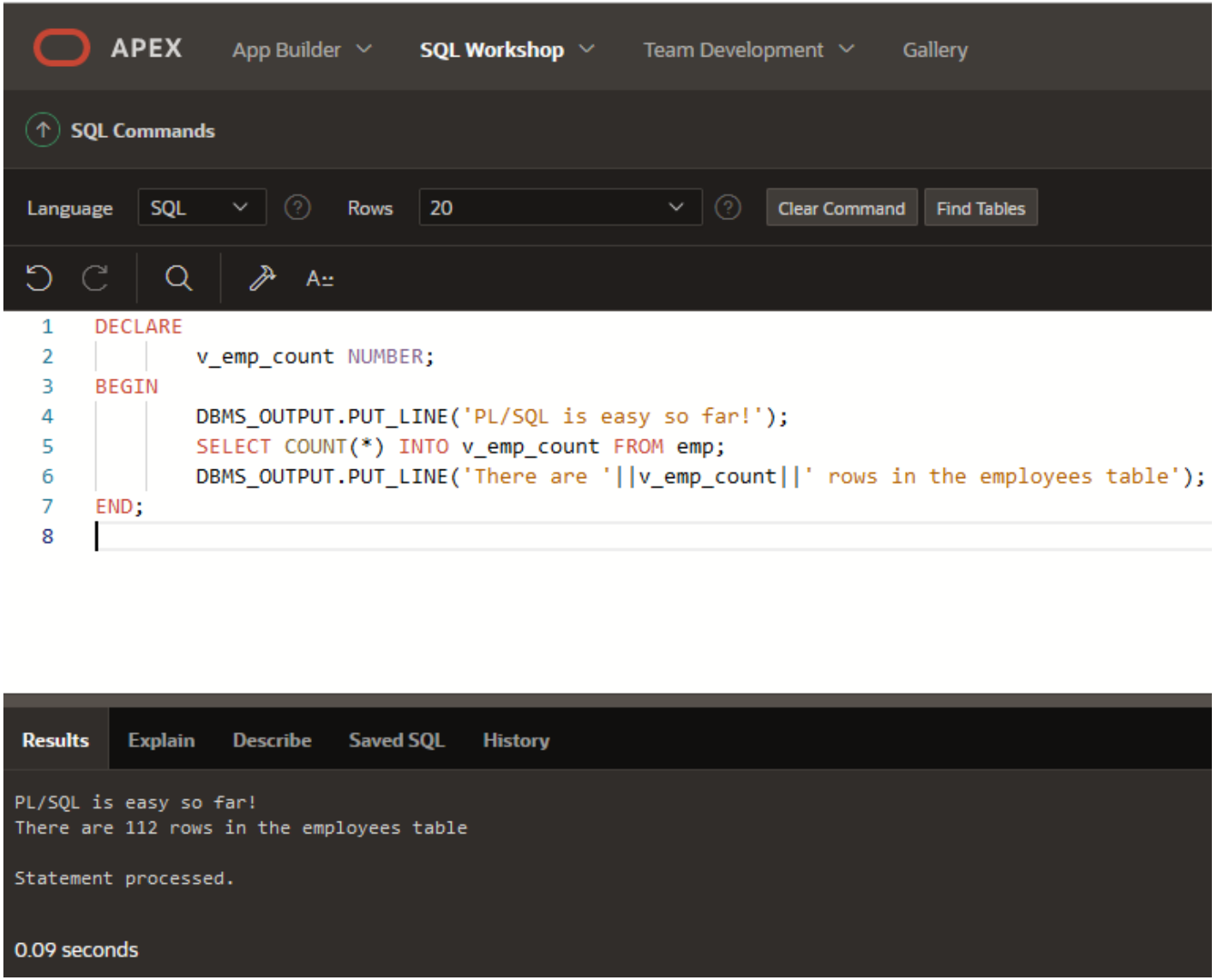
1. *un sir de caractere la un moment dat*
2. *concatena mai multe siruri de caractere intr-unul singur*

Instructiunea DBMS_OUTPUT.PUT_LINE

Exemplu:

```
DECLARE
        v_emp_count NUMBER;
BEGIN
        DBMS_OUTPUT.PUT_LINE('PL/SQL is easy
   so far!');
        SELECT COUNT(*) INTO v_emp_count
   FROM emp;
        DBMS_OUTPUT.PUT_LINE('There are
   '||v_emp_count||' rows in the employees table');
END;
```

Instructiunea DBMS_OUTPUT.PUT_LINE



Sintaxa unui bloc PL/SQL

- Un bloc anonim **PL/SQL** se compune din sectiuni si are sintaxa următoare:

Bloc anonim

```
DECLARE
   declaratii de variabile
BEGIN
   cod program
EXCEPTION
   cod tratare exceptii
END;
```

- Dacă blocul contine o procedură memorată în baza de date, sintaxa sa este următoarea:

Subprogram memorat de tip procedură

```
CREATE OR REPLACE PROCEDURE "nume"
(lista_parametri)
IS
   declaratii variabile
BEGIN
   cod program
EXCEPTION
   cod tratare exceptii
END;
```

Cuprins

2. Folosirea variabilelor in PL/SQL

- Vom studia declararea si initializarea variabilelor in sectiunea declarativa a unui bloc **PL/SQL**

- In **PL/SQL** se pot declara variabile care apoi pot fi folosite in instructiunile SQL si in cele procedurale.

2. Folosirea variabilelor in PL/SQL

Variabilele se utilizeaza pentru:

1. Stocarea temporara a datelor
2. Manipularea valorilor retinute
3. Refolosire

Manipularea variabilelor in PL/SQL

Variabilele sunt:

- Declarate si initializate in partea declarativa
- Folosite, si li se atribuite valori in partea executabila

Variabilele pot fi:

- Transmise ca parametri subprogramelor **PL/SQL**
- Folosite pentru a retine rezultatele unui subprogram **PL/SQL**

Declararea variabilelor

- Toate *variabilele **PL/SQL** trebuie declarate in partea declarativa* inainte de a fi referite de catre blocul **PL/SQL**

- Scopul unei unei declarari este de a aloca spatiu de memorie pentru o valoare, specificarea tipului de date si denumirea zonei de memorie pentru a putea fi folosita.

- Variabilele se pot declara in partea declarativa a oricarui bloc, subprogram si pachet **PL/SQL**

Declararea variabilelor – sintaxa

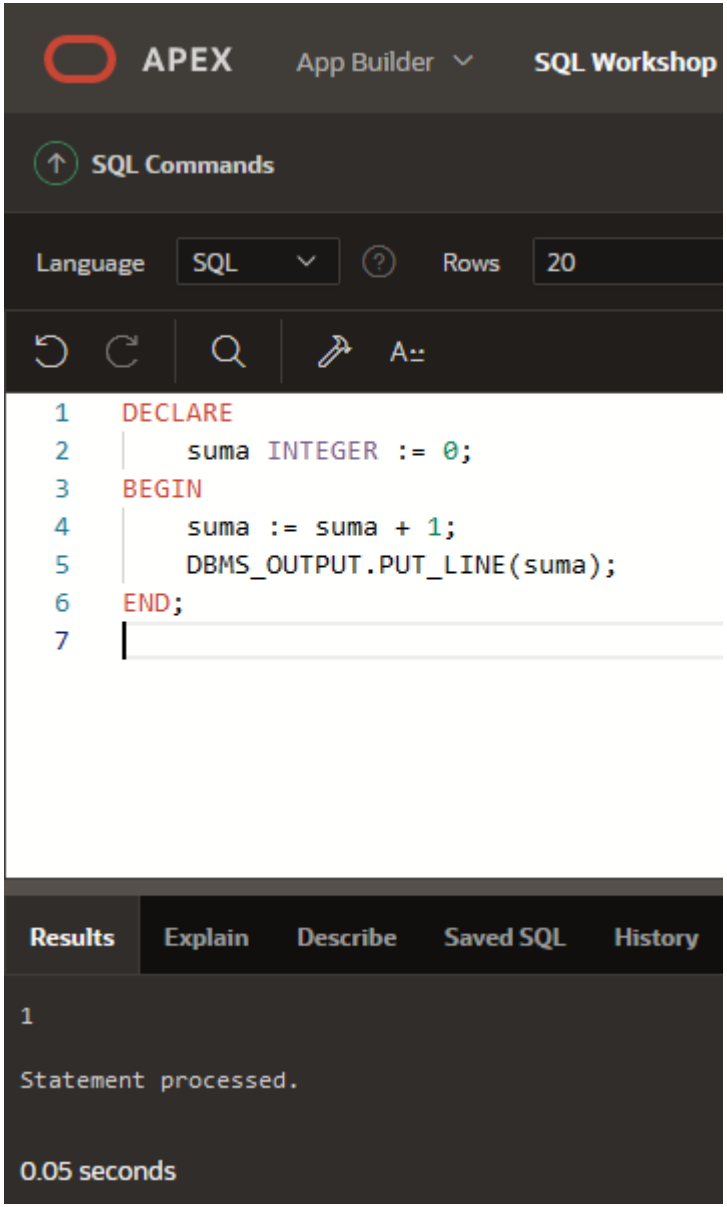
Identificator [CONSTANT] tip de date [NOT NULL] [:=expresie | DEFAULT expresie];

Initializarea variabilelor

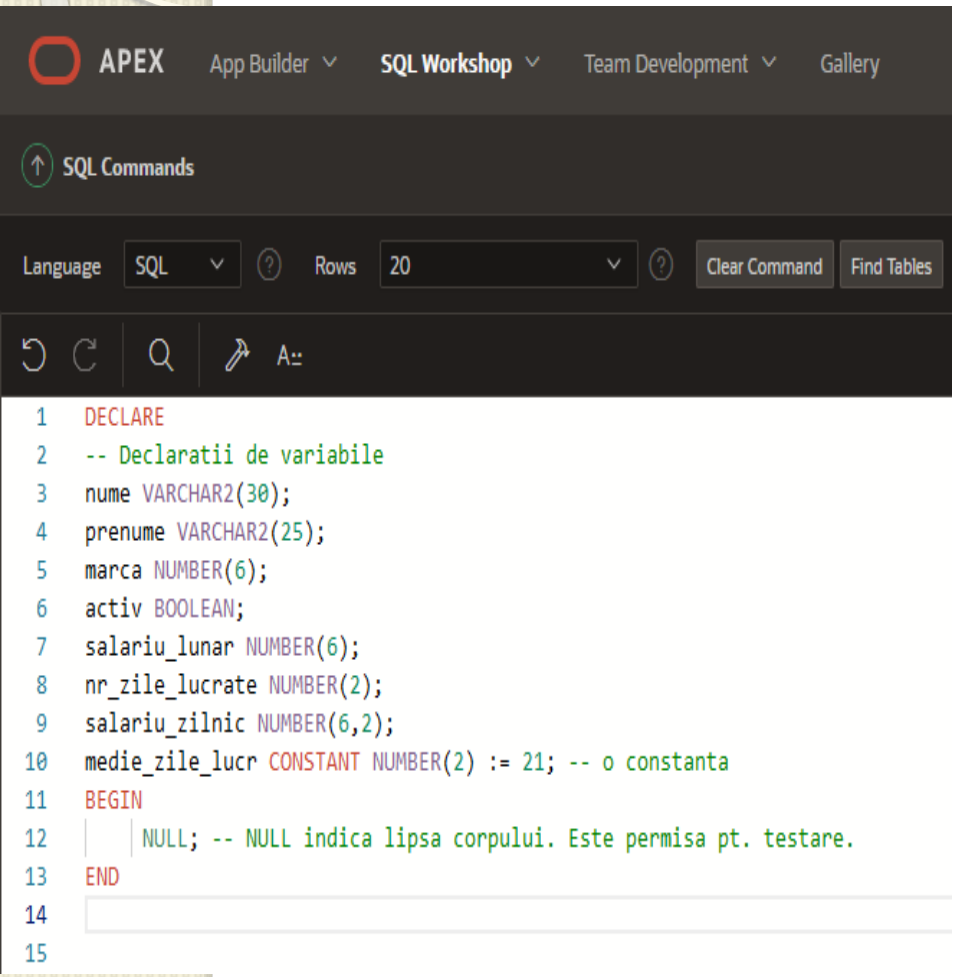
- Variabilelor li se asociaza o locatie de memorie in sectiunea DECLARE.
- Variabilelor li se atribuie o valoare la un moment dat.
- Acest lucru se numeste *initializare*.

```
DECLARE
   suma INTEGER := 0;
BEGIN
   suma := suma + 1;
   DBMS_OUTPUT.PUT_LINE(suma);
END;
```

Initializarea variabilelor

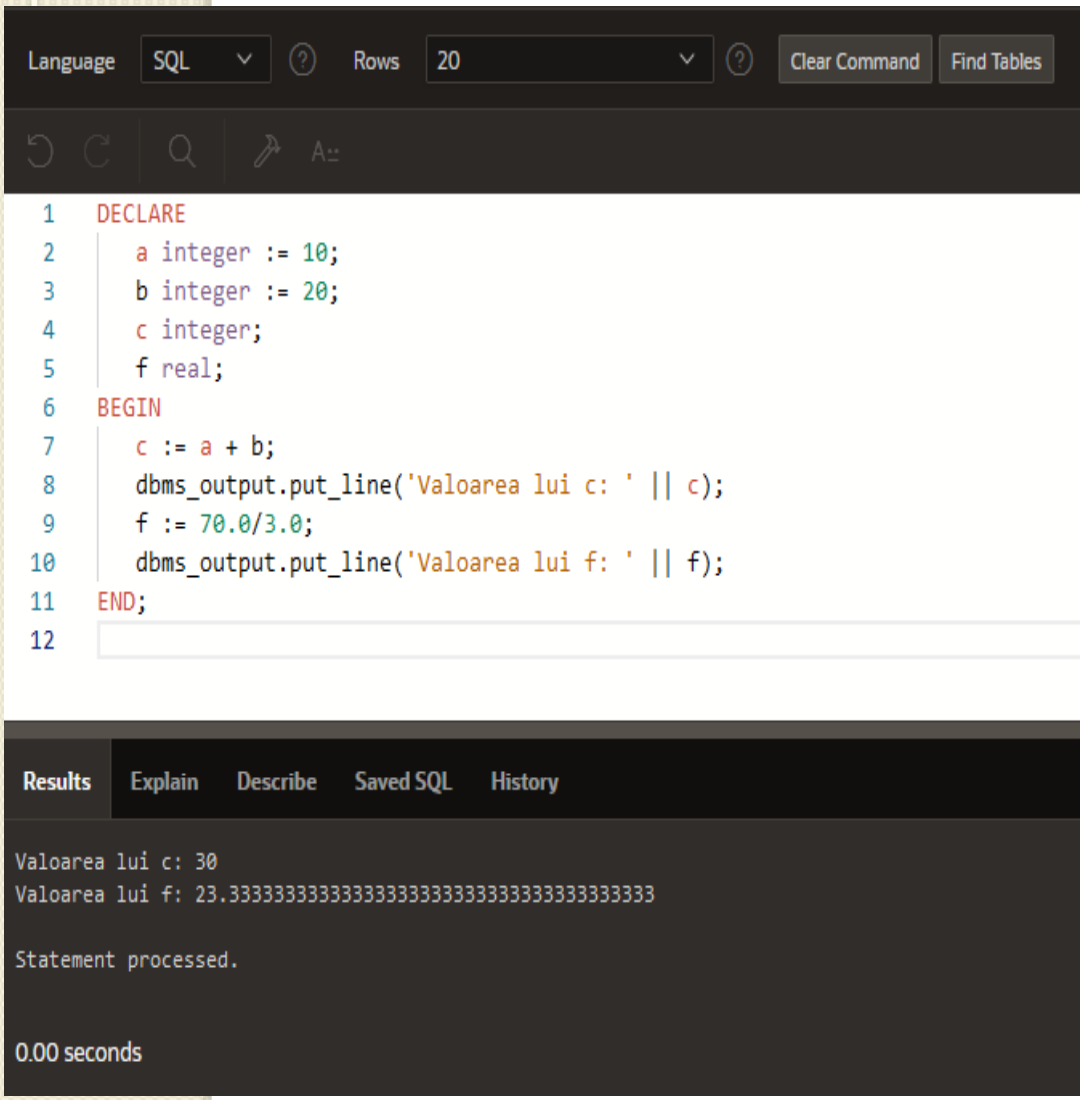


Exemple de declarare si initializare a variabilelor




```
-- Declaratii de variabile
nume VARCHAR2(30);
prenume VARCHAR2(25);
marca NUMBER(6);
activ BOOLEAN;
salariu_lunar NUMBER(6);
nr_zile_lucrate NUMBER(2);
salariu_zilnic NUMBER(6,2);
medie_zile_lucr CONSTANT NUMBER(2) := 21;
-- o constanta
BEGIN
    NULL; -- NULL indica lipsa corpului. Este permisa pt. testare.
END
```

Exemple de declarare si initializare a variabilelor




```
DECLARE
   a integer := 10;
   b integer := 20;
   c integer;
   f real;
BEGIN
   c := a + b;
dbms_output.put_line('Valoarea
lui c: ' || c);
   f := 70.0/3.0;
dbms_output.put_line('Valoarea
lui f: ' || f);
END;
```

Atribuirea de valori in sectiunea executabila

- Dupa ce o variabila a fost declarata o putem folosi in sectiunea executabila a unui bloc **PL/SQL**.
- De exemplu, in urmatorul bloc variabila ***v_nume*** este declarata in sectiunea **DECLARE**.
- Putem accesa aceasta variabila in sectiunea executabila a aceluiasi bloc.

Ce credeti ca va afisa urmatorul bloc?

```
DECLARE
  v_nume VARCHAR2(20);

BEGIN
  DBMS_OUTPUT.PUT_LINE('Numele este
  : '||v_nume);
  v_nume := 'Ion';
  DBMS_OUTPUT.PUT_LINE('Numele este
  : '||v_nume);
END;
```

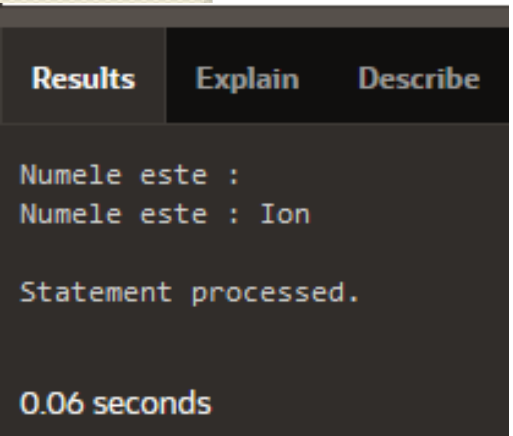
```
1  DECLARE
2      v_nume VARCHAR2(20);
3  BEGIN
4      DBMS_OUTPUT.PUT_LINE('Numele este : '||v_nume);
5      v_nume := 'Ion';
6      DBMS_OUTPUT.PUT_LINE('Numele este : '||v_nume);
7  END;
```

- In acest exemplu, valoarea ***Ion*** este atribuita unei variabile in sectiunea executabila.

- Valoarea variabilei este concatenata cu sirul de caractere ***Numele este:***

- Rezultatul afisat este:

Numele este :
Numele este : Ion
Statement processed.



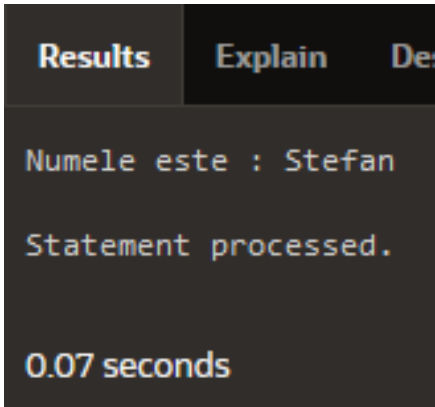
- In urmatorul bloc variabila ***v_nume*** este *declarata si initializata in sectiunea declarativa*.
- ***v_nume*** stocheaza valoarea ***Ion*** dupa initializare.

Valoarea este folosita in sectiunea executabila a blocului.

```
DECLARE
        v_nume VARCHAR2(20):= 'Ion';
BEGIN
        v_nume := 'Stefan';
        DBMS_OUTPUT.PUT_LINE('Numele este :
   '||v_nume);
END;
```

Se va afisa: Numele este : Stefan

```
1   DECLARE
2           v_nume VARCHAR2(20):= 'Ion';
3   BEGIN
4           v_nume := 'Stefan';
5           DBMS_OUTPUT.PUT_LINE('Numele este : '||v_nume);
6   END;
```



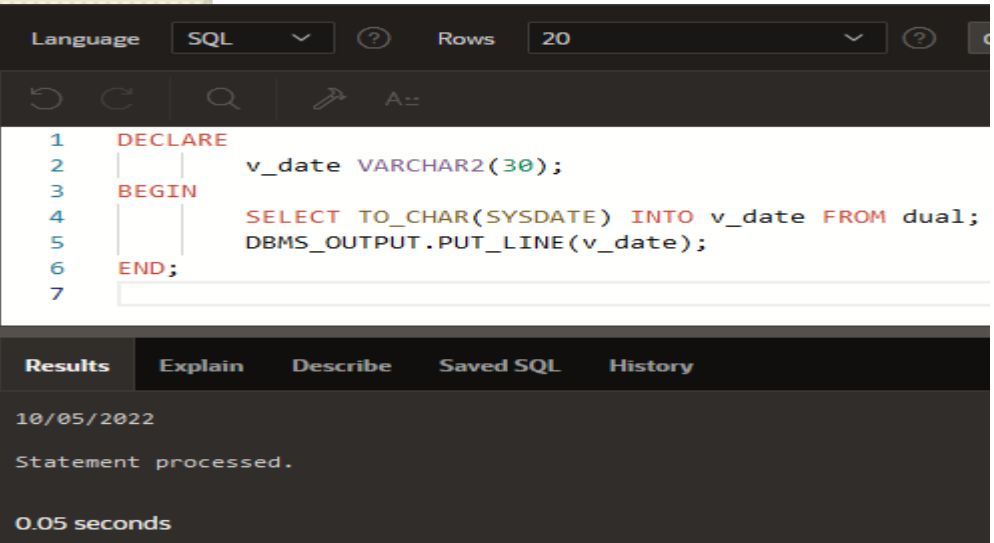
Transmiterea variabilelor ca parametri in subprogramele PL/SQL

- Parametrii sunt valori transmise programului de catre utilizator sau de catre alt program pentru personalizarea programului.
- *In **PL/SQL** subprogramele pot prelua parametri*.
- Se pot transmite variabilele ca parametri ai procedurilor si functiilor.

- In urmatorul exemplu parametrul ***v_date*** este transmis procedurii PUT_LINE care face parte din pachetul DBMS_OUTPUT.

```
DECLARE
        v_date VARCHAR2(30);
BEGIN
        SELECT TO_CHAR(SYSDATE) INTO v_date
    FROM dual;
        DBMS_OUTPUT.PUT_LINE(v_date);
END;
```

Afiseaza data calendaristica curenta



Cuprins

3. UNITATILE LEXICALE PL/SQL

- Unitati lexicale intr-un bloc **PL/SQL**:
  1. Blocurile
  2. Siruri de caractere ce includ:
     - litere
     - cifre
     - tab-uri si alte simboluri
- Unitatile lexicale pot fi clasificate in:
  - **3.1. Identificatori**
  - **3.2. Cuvinte rezervate**
  - **3.3. Delimitatori**
  - **3.4. Literali**
  - **3.5. Comentarii**

3.1. Identificatorii

- Un identificator este un nume dat unui obiect **PL/SQL**, incluzand pe oricare dintre urmatoarele:

Procedure	**Function**	**Variable**
Exception	**Constant**	**Package**
Record	**PL/SQL table**	**Cursor**

3.1. Identificatorii

- Proprietatile unui identificator:

1. contine cel mult 30 de caractere
2. trebuie sa inceapa cu o litera
3. poate contine caracterele $, _ (underscore), #
4. nu poate contine spatii
5. identificatorii nu sunt case sensitive

3.1. Identificatorii

Exemple de identificatori corecti

First_Name	LastName	address_1
ID#	Total_$	primary_department_contact

Exemple de identificatori incorecti

First Name	Contains a space
Last-Name	Contains invalid "-"
1st_address_line	Begins with a number
Total_%	Contains invalid "%"
primary_building_department_contact	More than 30 characters

3.2. Cuvinte rezervate

- *Cuvintele rezervate sunt acele cuvinte care au o semnificatie speciala pentru baza de date Oracle.*
- Cuvintele rezervate nu pot fi folosite ca identificatori intr-un program **PL/SQL**.
- O parte dintre cuvintele rezervate sunt:

ALL	**CREATE**	**FROM**	**MODIFY**	**SELECT**
ALTER	**DATE**	**GROUP**	**NOT**	**SYNONYM**
AND	**DEFAULT**	**HAVING**	**NULL**	**SYSDATE**
ANY	**DELETE**	**IN**	**NUMBER**	**TABLE**
AS	**DESC**	**INDEX**	**OR**	**THEN**
ASC	**DISTINCT**	**INSERT**	**ORDER**	**UPDATE**
BETWEEN	**DROP**	**INTEGER**	**RENAME**	**VALUES**
CHAR	**ELSE**	**INTO**	**ROW**	**VARCHAR2**
COLUMN	**EXISTS**	**IS**	**ROWID**	**VIEW**
COMMENT	**FOR**	**LIKE**	**ROWNUM**	**WHERE**

3.3. Delimitatori

- *Delimitatorii sunt simboluri care au semnificatie speciala pentru baza de date* **Oracle**:
- Delimitatori simpli:

Symbol	Meaning
+	Addition operator
–	Subtraction/negation operator
*	Multiplication operator
/	Division operator
=	Equality operator
'	Character string delimiter
;	Statement terminator

Delimitatorii compusi:

Symbol	Meaning
<>	Inequality operator
!=	Inequality operator
||	Concatenation operator
--	Single-line comment indicator
/*	Beginning comment delimiter
*/	Ending comment delimiter
:=	Assignment operator

3.4. Literali:

- *Un literal poate fi un numar, un sir de caractere, o data calendaristica sau o valoare booleana explicita care nu poate fi reprezentata printr-un identificator*.

Literalii se clasifica:

3.4.1. literali de tip sir de caractere

3.4.2. literali de tip numeric

3.4.3. literali de tip Boolean

3.4.1. Literalii siruri de caractere

- Literalii siruri de caractere includ toate caracterele printabile din multimea de caractere **PL/SQL**:
  - litere
  - numere
  - spatii
  - simboluri speciale

- *Literalii siruri de caractere sunt de tipul CHAR si trebuie scrisi intre apostrofuri*

3.4.1. Literalii siruri de caractere

- Literalii siruri de caractere pot fi formati din 0 sau mai multe caractere din multimea de caractere **PL/SQL**
- Literalii siruri de caractere sunt case sensitive

Exemple:

```
v_prenume := 'Ion';
v_grupa := '134A';
v_data_astazi := '13-OCT-2015';
```

3.4.2. Literalii de tip numeric

- Literalii numerici sunt valori numerice intregi sau reale
- Literalii numerici se pot reprezenta ca o valoare simpla (de exemplu -32.5) sau prin notatia stiintifica (de exemplu 2E5 ce semnifica $2*10^5$ -> 200000)

Exemple:

```
v_elevation := 428;
v_order_subtotal := 1025.69;
v_growth_rate := .56;
v_distance_sun_to_centauri := 4.3E13;
```

3.4.3. Literalii de tip Boolean

- Literalii de tip Boolean sunt valori ce sunt atribuite variabilelor booleene
- Literalii de tip Boolean nu se pun intre apostrofuri sau ghilimele
- TRUE, FALSE si NULL sunt literali de tip Boolean sau cuvinte cheie

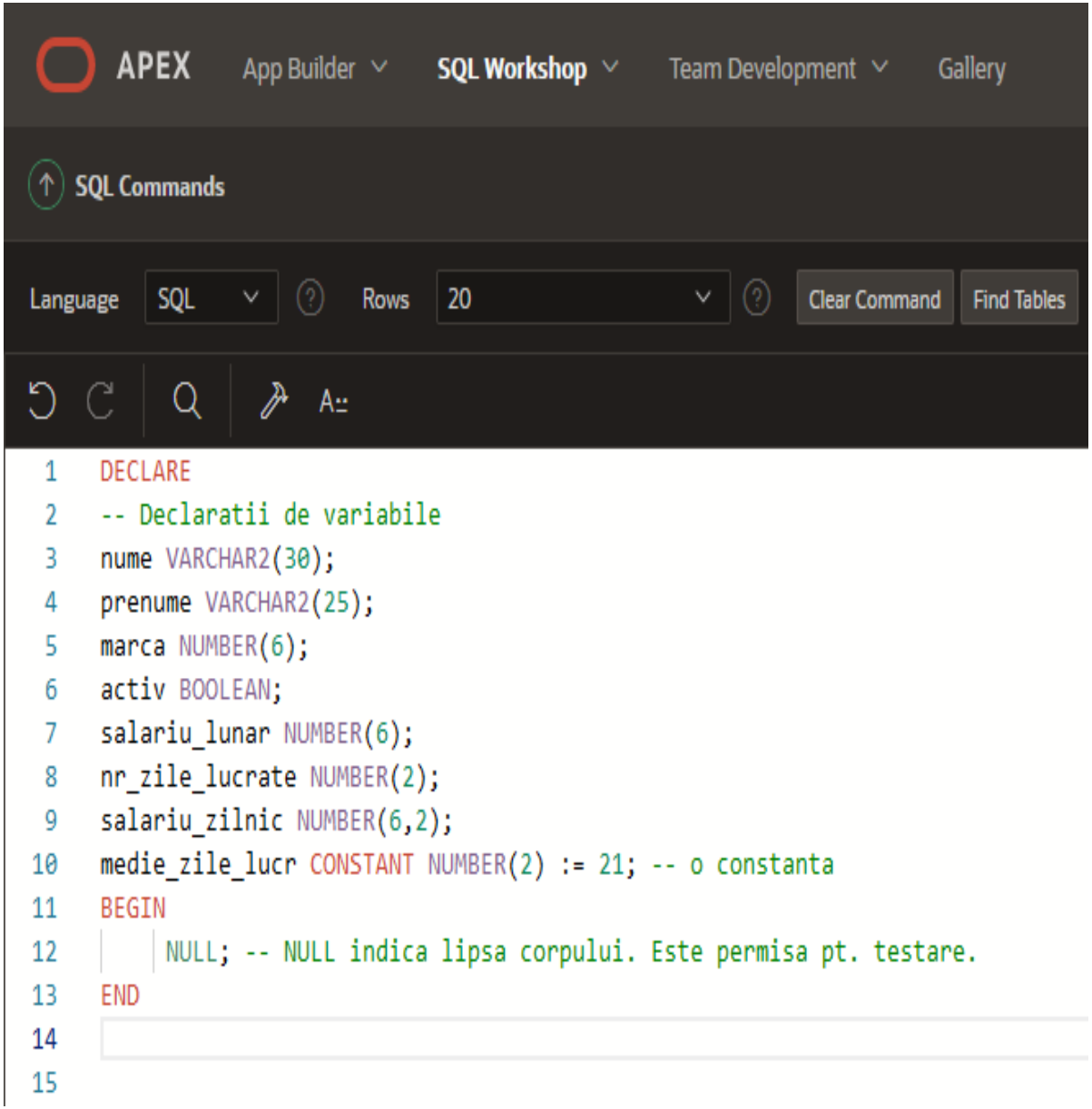
Exemple:

```
v_new_customer := FALSE;
v_paid_in_full := TRUE;
v_authorization_approved := FALSE;
v_high_school_diploma := NULL;
v_island :=FALSE;
```

3.5. Comentarii

- *Comentariile ofera explicatii cu privire la ceea ce realizeaza un anumit cod de program*.
- Comentariile plasate acolo unde trebuie sunt foarte importante pentru intelegerea si intretinerea viitoare a programului.
- Folosirea comentariilor este o buna practica in programare.
- Comentariile sunt ignorate de **PL/SQL**. Sunt instructiuni pe care **PL/SQL** nu le executa.

3.5. Comentarii




```
1   DECLARE
2   -- Declaratii de variabile
3   nume VARCHAR2(30);
4   prenume VARCHAR2(25);
5   marca NUMBER(6);
6   activ BOOLEAN;
7   salariu_lunar NUMBER(6);
8   nr_zile_lucrate NUMBER(2);
9   salariu_zilnic NUMBER(6,2);
10  medie_zile_lucr CONSTANT NUMBER(2) := 21; -- o constanta
11  BEGIN
12      NULL; -- NULL indica lipsa corpului. Este permisa pt. testare.
13  END
14
15
```

Cuprins

4. Tipuri de date PL/SQL

- *Un tip de date specifica un format de stocare, restrictii si un domeniu de valori.*

- **PL/SQL** suporta 5 categorii de tipuri de date:

  1. Scalar – stocheaza o singura valoare

  2. Compus – contine elemente care pot fi atat de tip scalar (record) cat si de tip compus (record si tabela)

4. Tipuri de date PL/SQL

3. LOB (Large Object) – stocheaza valori ce sunt denumite locatori care specifica locatia unor obiecte mari (cum ar fi imaginile grafice) care sunt stocate out of line.

4. Referinta – stocheaza valori, se numesc pointeri si indica catre o locatie de memorie

5. Obiect – Este un obiect schema care are nume, atribute si metode. Un tip de date obiect este asemanator ca mecanism cu clasele din C++ si Java

4.1. Tipurile de date scalare

- Stocheaza o singura valoare
- Nu au componente interne
- Pot fi clasificate in 4 categorii:
  1. Character
  2. Number
  3. Date
  4. Boolean

- **Tipuri de date scalare:**

Character (or String)

CHAR [(*maximum_length*)]	Base type for fixed-length character data up to 32,767 bytes. If you do not specify a *maximum_length*, the default length is set to 1.
VARCHAR2(*maximum_length*)	Base type for variable-length character data up to 32,767 bytes. There is no default size for VARCHAR2 variables and constants.
LONG	Character data of variable length (a bigger version of the VARCHAR2 data type).
LONG RAW	Raw binary data of variable length (not interpreted by PL/SQL).

- **Tipuri de date scalare: Number**

NUMBER [(*precision, scale*)]	Number having precision *p* and scale *s*. The precision *p* can range from 1 to 38. The scale *s* can range from –84 to 127.
BINARY_INTEGER	Base type for signed integers between -2,147,483,647 and 2,147,483,647.
PLS_INTEGER	Base type for signed integers between -2,147,483,647 and 2,147,483,647. PLS_INTEGER and BINARY_INTEGER values require less storage and are faster than NUMBER values.
BINARY_FLOAT BINARY_DOUBLE	New data types introduced in Oracle Database 10*g*. They represent a floating-point number in the IEEE 754 format. BINARY_FLOAT requires 5 bytes to store the value and BINARY_DOUBLE requires 9 bytes.

• Tipuri de date scalare: Date

DATE	Base type for dates and times. DATE values include the time of day in seconds since midnight. The range for dates is between 4712 B.C. and A.D. 9999.
TIMESTAMP	The TIMESTAMP data type, which extends the DATE data type, stores the year, month, day, hour, minute, second, and fraction of seconds.
TIMESTAMP WITH TIME ZONE	The TIMESTAMP WITH TIME ZONE data type, which extends the TIMESTAMP data type, includes a time-zone displacement—that is, the difference (in hours and minutes) between local time and Coordinated Universal Time (UTC), formerly known as Greenwich Mean Time.
TIMESTAMP WITH LOCAL TIME ZONE	This data type differs from TIMESTAMP WITH TIME ZONE in that when you insert a value into a database column, the value is normalized to the database time zone, and the time-zone displacement is not stored in the column. When you retrieve the value, the Oracle server returns the value in your local session time zone.
INTERVAL YEAR TO MONTH	You use the INTERVAL YEAR TO MONTH data type to store and manipulate intervals of years and months.
INTERVAL DAY TO SECOND	You use the INTERVAL DAY TO SECOND data type to store and manipulate intervals of days, hours, minutes, and seconds

- **Tipuri de date scalare: Boolean**

BOOLEAN	Base type that stores one of the three possible values used for logical calculations: TRUE, FALSE, or NULL.

Tipuri de date compuse:

- Un tip de date scalar nu are componente interne.
- *Un tip compus are componente interne care pot fi folosite individual.*
- Tipurile de date compuse includ urmatoarele:
  1. **TABLE**
  2. **RECORD**
  3. **NESTED TABLE**
  4. **VARRAY**

3. Tipul de date LOB

- *Obiectele de mari dimensiuni (Lobs) au rolul de a stoca un volum mare de informatii*

- O coloana dintr-o baza de date se poate incadra in categoria LOB

- Tipurile de date LOB permit un acces eficient si aleator la date, si pot fi atributele unui tip obiect

3. Tipul de date LOB

- Exista cateva categorii de tipuri de date LOB:
  1. Character large object (CLOB)
  2. Binary large object (BLOB)
  3. Binary file (BFILE)
  4. National language character large object (NCLOB)

- Tipurile de date LOB ne permit sa stocam blocuri de date nestructurate de o dimensiune pana la 4 gigabytes

- Book (CLOB)

- Photo(BLOB)

- Movie (BFILE)

- NCLOB

Cuprins

5. Utilizarea tipurilor de date scalare

Declararea variabilelor de tip character:

- Tipurile de date CHARACTER includ: CHAR, VARCHAR2 si LONG

```
DECLARE
   v_emp_job VARCHAR2(9);
   v_order_no VARCHAR2(6);
   v_product_id VARCHAR2(10);
   v_rpt_body_part LONG;
```

Declararea variabilelor numerice

- Tipurile de date numerice includ: NUMBER, PLS_INTEGER, BINARY_INTEGER si BINARY_FLOAT.
- Daca se foloseste constrangerea CONSTANT, valoarea variabilei nu se poate schimba.
- Constantele trebuie initializate.

INTEGER este un alias pentru NUMBER(38,0).

```
DECLARE
   v_dept_total_sal NUMBER(9,2) := 0;
   v_count_loop INTEGER := 0;
   c_tax_rate CONSTANT NUMBER(3,2) := 8.25;
```

Declararea variabilelor de tip date (data calendaristica)

- Tipurile **DATE** includ:
  1. DATE
  2. TIMESTAMP
  3. TIMESTAMP WITH TIMEZONE

```
DECLARE
   v_orderdate DATE := SYSDATE + 7;
   v_natl_holiday DATE;
   v_web_sign_on_date TIMESTAMP;
```

Declararea variabilelor booleene

- Tipul de date **BOOLEAN** stocheaza 3 valori folosite pentru expresii logice:
  1. TRUE
  2. FALSE
  3. NULL

```
DECLARE
   v_valid BOOLEAN NOT NULL := TRUE;
   v_is_found BOOLEAN := FALSE;
   v_underage BOOLEAN;
```

Declararea variabilelor booleene(continuare)

- Unei variabile de tip boolean i se poate atribui doar una dintre valorile: **TRUE**, **FALSE**, **NULL**

- Expresiile conditionale folosesc operatorii logici **AND**, **OR** si **NOT** pentru a verifica valorile variabilelor

- Pentru a returna valori de tip Boolean, se pot folosi expresii aritmetice, de tip char sau data calendaristica.

Reguli pentru declararea si initializarea variabilelor PL/SQL

- Folositi nume semnificative si respectati conventiile de denumire
- Declarati un singur identificator pe linie pentru o vizualizare mai buna, pentru o intelegere si intretinere a codului mai usoara
- Folositi restrictia **NOT NULL** atunci cand doriti ca variabila sa contina o valoare
- Evitati folosirea denumirilor de coloane ca identificatori

Exemplu de utilizare incorecta a denumirilor de variabile:

```
DECLARE
        empno NUMBER(4,0);
BEGIN
        SELECT empno
        INTO empno
        FROM emp
        WHERE deptno = 10;
END;
```

Numele variabilei este identic cu numele coloanei

```
1   DECLARE
2           empno NUMBER(4,0);
3   BEGIN
4           SELECT empno
5           INTO empno
6           FROM emp
7           WHERE deptno = 10;
8   END;
```

```
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 5
ORA-06512: at "SYS.DBMS_SQL", line 1721


3.          ename VARCHAR2(50);
4. BEGIN
5.              SELECT empno, ename
6.              INTO empno, ename
7.              FROM emp
```

Variabilele de ancorare cu atributul %TYPE

- Uneori, decat sa apelati la un cod care s-ar scrie mai dificil, este indicat sa folositi atributul **%TYPE** pentru a declara o variabila in acelasi fel cu o alta variabila declarata anterior sau cu o coloana a bazei de date.
- Atributul **%TYPE** *este folosit mai ales atunci cand valoarea stocata in variabila este derivata dintr-o tabela a bazei de date*.
- Cand folosim atributul **%TYPE** pentru a declara o variabila, il vom prefixa cu denumirea tabelei din baza de date si a coloanei.

Exemplu de tabela si de bloc **PL/SQL** care o foloseste:

```
CREATE TABLE myemps (
        emp_name VARCHAR2(6),
        emp_salary NUMBER(6,2));
DECLARE
        v_emp_salary NUMBER(6,2);
BEGIN
        SELECT emp_salary
        INTO v_emp_salary
        FROM myemps
        WHERE emp_name = 'Smith';
END;
```

- Acest bloc **PL/SQL** stocheaza salariul corect in variabila ***v_emp_salary***.
- Dar ce se va intampla daca coloana tabelei va fi modificata ulterior?
- Atributul **%TYPE**:
  - Este folosit pentru a da automat unei variabile acelasi tip de date si aceeasi dimensiune ca si:
    - In definirea unei coloane dintr-o tabela
    - O variabila declarata anterior
  - Este prefixat cu oricare dintre urmatoarele:
    - Denumirea unei tabele dintr-o baza de date si a unei coloane
    - Numele unei alte variabile declarate anterior

Declararea variabilelor cu atributul %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;
...
```

Avantajele atributului %TYPE

- Se pot evita erorile cauzate de nepotrivirile de tip de date sau de precizie
- Nu este necesara schimbarea declaratiei variabilei daca se schimba definirea coloanei
- Atunci cand se foloseste atributul **%TYPE**, **PL/SQL** determina tipul de date si dimensiunea variabilei la compilarea blocului. Acest lucru asigura compatibilitatea variabilei cu coloana pe care o va complete.

Exemplu

```
CREATE TABLE myemps (
        emp_name VARCHAR2(6),
        emp_salary NUMBER(6,2));
DECLARE
  v_emp_salary myemps.emp_salary%TYPE;
BEGIN
        SELECT emp_salary
        INTO v_emp_salary
        FROM myemps
        WHERE emp_name = 'Smith';
END;
```

- Blocul **PL/SQL** continua sa ruleze corect chiar daca tipul de date al coloanei este modificat ulterior

Întrebări?