Curs 3

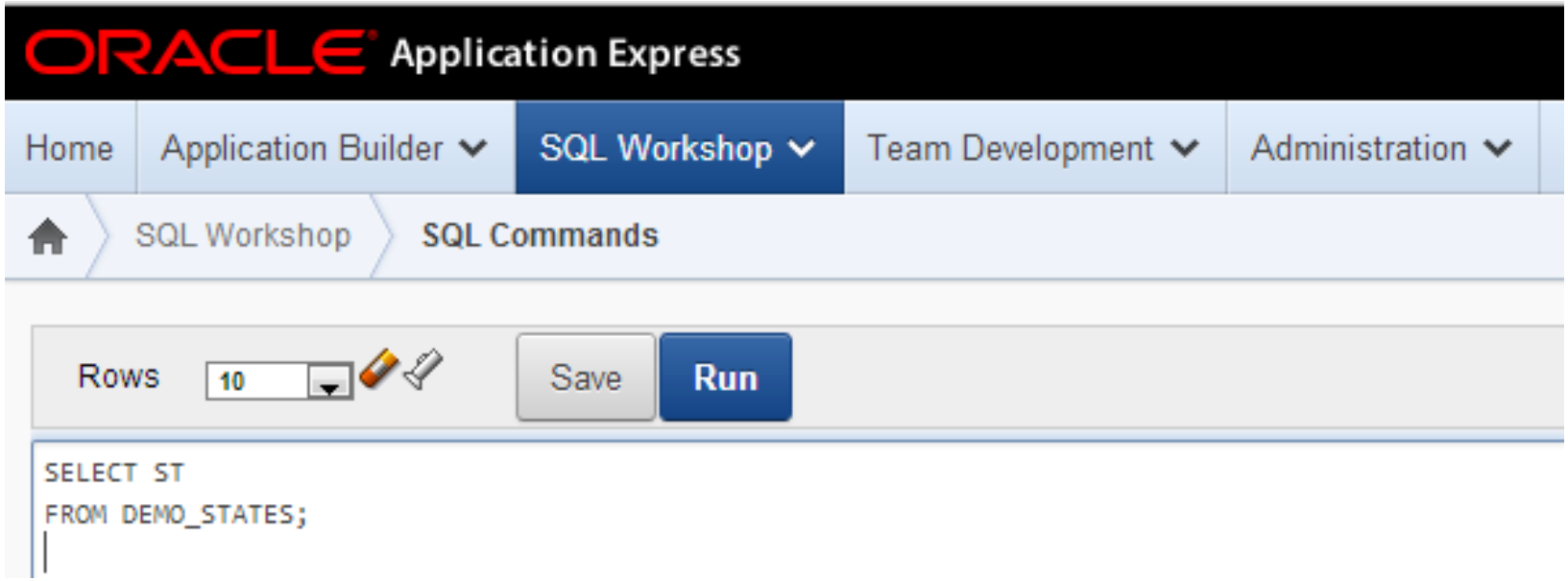
Funcţii SQL, operatori şi vizibilitatea variabilelor în PL/SQL

Cuprins

1. Functiile SQL in PL/SQL

Sunt deja cunoscute instructiunile SQL.
De exemplu:

```
SELECT ST
FROM DEMO_STATES;
```

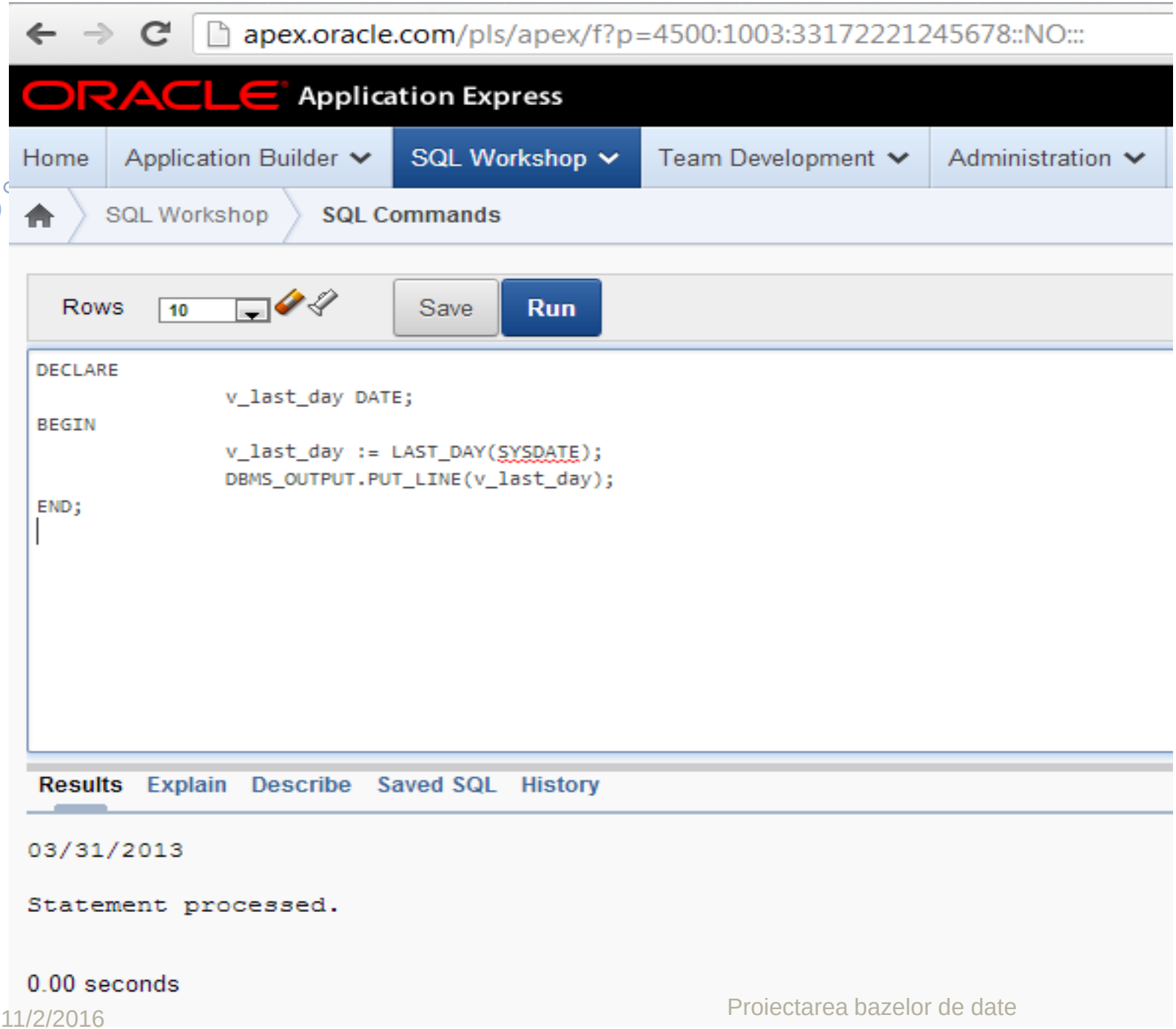


1. Functiile SQL in PL/SQL

Exista functii SQL care pot fi folosite si in instructiunile procedurale PL/SQL.

De exemplu:

```
DECLARE
        v_last_day DATE;
BEGIN
        v_last_day := LAST_DAY(SYSDATE);
        DBMS_OUTPUT.PUT_LINE(v_last_day);
END;
```


apex.oracle.com/pls/apex/f?p=4500:1003:33172221245678::NO:::
ORACLE Application Express
Home
Application Builder
SQL Workshop
Team Development
Administration
SQL Workshop
SQL Commands
Rows
10
Save
Run
DECLARE
v_last_day DATE;
BEGIN
v_last_day := LAST_DAY(SYSDATE);
DBMS_OUTPUT.PUT_LINE(v_last_day);
END;
Results
Explain
Describe
Saved SQL
History
03/31/2013
Statement processed.
0.00 seconds

1. Functiile SQL in PL/SQL

Sunt disponibile in instructiunile procedurale:

1. Functiile single-row pentru caractere
2. Functiile numerice single-row
3. Functiile pentru date calendaristice
4. Functiile pentru conversiile de tipuri de date
5. Functii diverse

Nu sunt disponibile in instructiunile procedurale:

1. DECODE
2. Functiile de grup (functiile multiple-row)

1. Functiile SQL in PL/SQL

1. Functii pentru caractere:

Functiile pentru caractere valide in PL/SQL sunt:

ASCII	LENGTH	RPAD
CHR	LOWER	RTRIM
CONCAT	LPAD	SUBSTR
INITCAP	LTRIM	TRIM
INSTR	REPLACE	UPPER

1. Functiile SQL in PL/SQL

Exemple de functii pentru caractere:

Referitor la lungimea unui sir

```
v_desc_size INTEGER(5);
v_prod_description VARCHAR2(70):='You
   can use this product with your radios for
   higher frequency';

v_desc_size:= LENGTH(v_prod_description);
```

1. Functiile SQL in PL/SQL

Scrierea numelui capitalei unei tari cu majuscule:

```
v_capitol_name:= UPPER(v_capitol_name);
```

Concatenarea prenumelui cu numele:

```
v_emp_name:=v_first_name||'
  '||v_last_name;
```

1. Functiile SQL in PL/SQL

2. Functii numerice

Functiile numerice valide in PL/SQL includ:

ABS	EXP	ROUND
ACOS	LN	SIGN
ASIN	LOG	SIN
ATAN	MOD	TAN
COS	POWER	TRUNC

1. Functiile SQL in PL/SQL

Exemple de functii numerice

Preluarea semnului unui numar

```
DECLARE
  v_my_num BINARY_INTEGER :=-56664;

BEGIN
      DBMS_OUTPUT.PUT_LINE(SIGN(v_my
  _num));
END;
```

1. Functiile SQL in PL/SQL

Rotunjirea unei valori numerice

```
DECLARE
        v_median_age NUMBER(6,2);
BEGIN
        SELECT median_age
        INTO v_median_age
        FROM countries
        WHERE country_id=27;
  DBMS_OUTPUT.PUT_LINE(ROUND(v_median
  _age,0));
END;
```

1. Functiile SQL in PL/SQL

3. Functii pentru date calendaristice

Functiile pentru date calendaristice valide in PL/SQL includ:

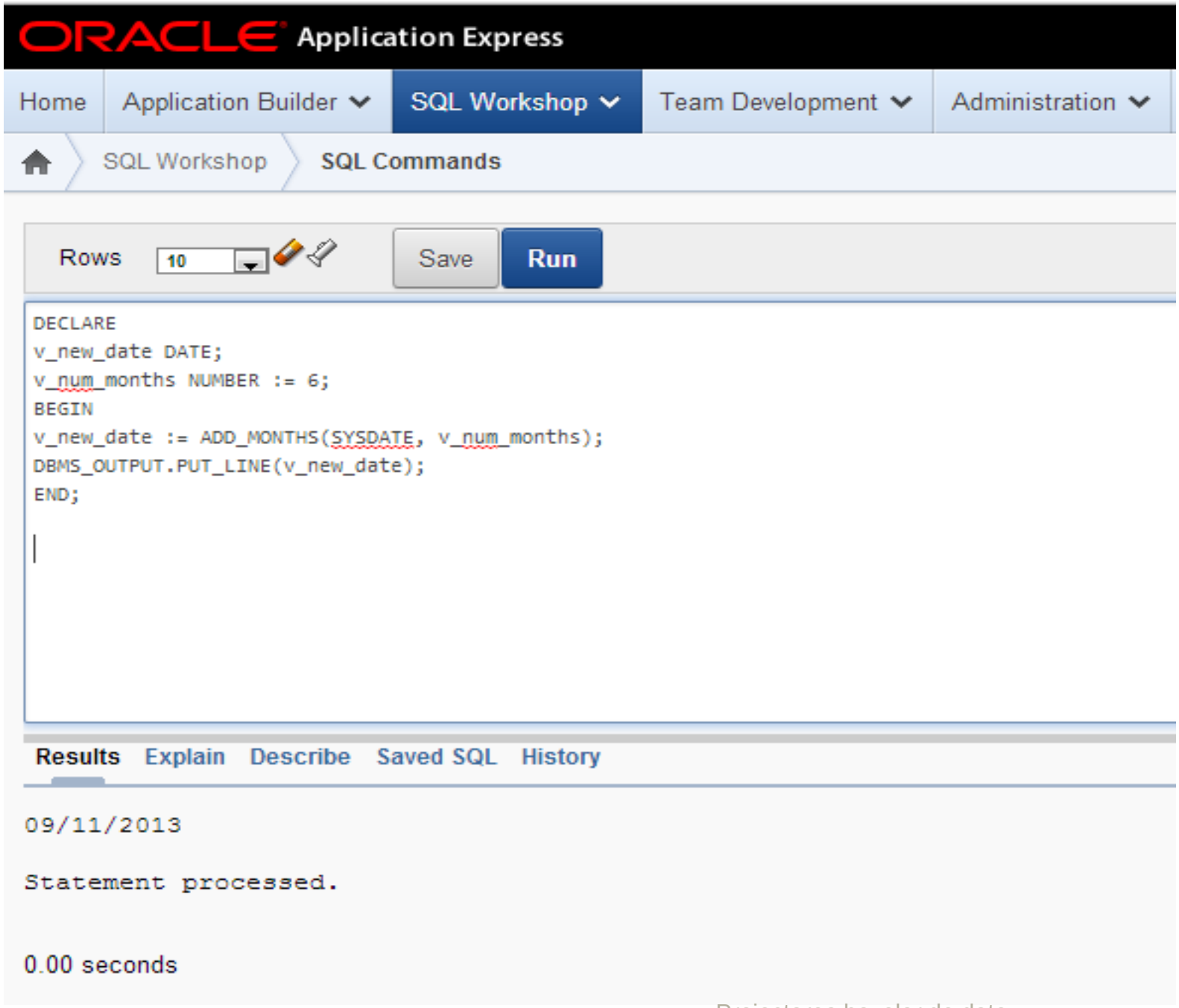
ADD_MONTHS	**MONTHS_BETWEEN**
CURRENT_DATE	**ROUND**
CURRENT_TIMESTAMP	**SYSDATE**
LAST_DAY	**TRUNC**

1. Functiile SQL in PL/SQL

Exemple de functii pentru date calendaristice:

Adunarea unui anumit numar de luni la o data calendaristica

```
DECLARE
v_new_date DATE;
v_num_months NUMBER := 6;
BEGIN
v_new_date := ADD_MONTHS(SYSDATE,
  v_num_months);
DBMS_OUTPUT.PUT_LINE(v_new_date);
END;
```


ORACLE Application Express
Home
Application Builder
SQL Workshop
Team Development
Administration
SQL Workshop
SQL Commands
Rows
10
Save
Run
DECLARE
v_new_date DATE;
v_num_months NUMBER := 6;
BEGIN
v_new_date := ADD_MONTHS(SYSDATE, v_num_months);
DBMS_OUTPUT.PUT_LINE(v_new_date);
END;
Results
Explain
Describe
Saved SQL
History
09/11/2013
Statement processed.
0.00 seconds

1. Functiile SQL in PL/SQL

- Calcularea numarului de luni dintre doua date calendaristice

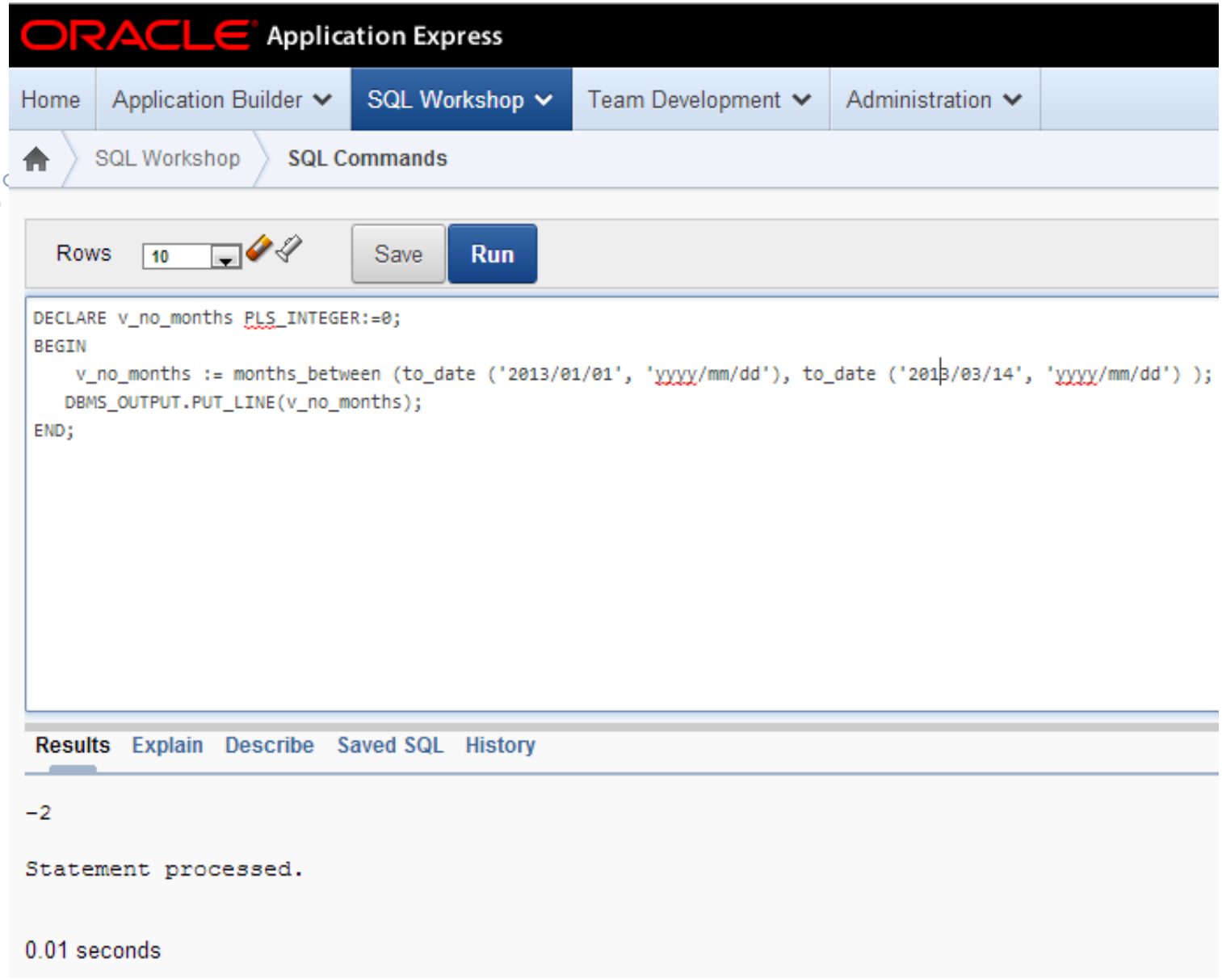
```
DECLARE v_no_months PLS_INTEGER:=0;
BEGIN
   v_no_months := months_between (to_date
  ('2013/01/01', 'yyyy/mm/dd'), to_date
  ('2013/03/14', 'yyyy/mm/dd') );
  DBMS_OUTPUT.PUT_LINE(v_no_months);
END;
```

Rezultat:

-2

Statement processed.

0.01 seconds


ORACLE Application Express
Home
Application Builder
SQL Workshop
Team Development
Administration
SQL Workshop
SQL Commands
Rows
10
Save
Run
DECLARE v_no_months PLS_INTEGER:=0;
BEGIN
v_no_months := months_between (to_date ('2013/01/01', 'yyyy/mm/dd'), to_date ('2013/03/14', 'yyyy/mm/dd'));
DBMS_OUTPUT.PUT_LINE(v_no_months);
END;
Results
Explain
Describe
Saved SQL
History
-2
Statement processed.
0.01 seconds

Cuprins

1. Functiile SQL in PL/SQL
2. Conversii de tipuri de date
3. Operatori in PL/SQL
4. Blocuri imbricate si vizibilitatea variabilelor
5. Domeniul de aplicare al variabilelor
6. Variabile locale si globale
7. Domeniul de aplicare a exceptiilor in blocurile imbricate

2. Conversii de tipuri de date

- In orice limbaj de programare conversia de la un tip de date la altul este o cerinta obisnuita.

- PL/SQL poate manipula astfel de conversii cu tipuri de date scalare.

Conversiile de tipuri de date pot fi de doua tipuri:

1. Conversii implicite
2. Conversii explicite

2. Conversii de tipuri de date

1. Conversii implicite

- In conversiile implicite, PL/SQL incearca convertirea tipurilor de date dinamic, daca sunt in diverse forme, intr-o instructiune.

- Conversiile implicite pot avea loc intre multe tipuri de date in PL/SQL dupa cum sunt ilustrate in urmatoarea schema:

2. Conversii de tipuri de date

	DATE	LONG	NUMBER	PLS_INTEGER	VARCHAR2
DATE	N/A	X			X
LONG		N/A			X
NUMBER		X	N/A	X	X
PLS_INTEGER		X	X	N/A	X
VARCHAR2	X	X	X	X	N/A

2. Conversii de tipuri de date

- **Exemple de conversie implicita**

```
DECLARE
        v_salary NUMBER(6):=6000;
        v_sal_increase VARCHAR2(5):='1000';
        v_total_salary v_salary%TYPE;
BEGIN
        v_total_salary:= v_salary + v_sal_increase;
        DBMS_OUTPUT.PUT_LINE(v_total_salary);
END;
```

- In acest exemplu, variabila **v_sal_increase** este de tipul VARCHAR2.
- Atunci cand este calculat salariul total, PL/SQL mai intai converteste **v_sal_increase** in numar, iar apoi efectueaza calculele.
- Rezultatul expresiei este de tip numeric.

Dezavantajele conversiilor implicite

- La prima vedere, conversiile implicite par a fi utile.
- Totusi exista cateva dezavantaje:

1. Conversiile implicite pot fi mai lente

2. *Cand se folosesc conversiile implicite pierdem controlul asupra programului deoarece nu stim exact cum manipuleaza* Oracle *datele*. Daca Oracle schimba regulile de conversie atunci va fi afectat codul programului.

Dezavantajele conversiilor implicite

3. *Regulile de conversie implicita depind de mediul de programare.*

- De exemplu, formatul datelor calendaristice depinde de setarile de limbaj si de tipul instalarii.
- Codurile care folosesc conversii implicite pot sa nu ruleze pe alte servere sau in alte limbaje.

4. *Codurile care folosesc conversiile implicite sunt mai greu de citit si de inteles.*

5. Este responsabilitatea programatorului sa se asigure ca valorile pot fi convertite.

- De exemplu, PL/SQL poate converti valoarea CHAR ’13-OCT-15’ la o valoare de tip data calendaristica, dar nu poate converti valoarea CHAR ‘Yesterday’ la data calendaristica.
- In mod asemanator, PL/SQL nu poate converti o valoare de tip VARCHAR2 care contine caractere alfabetice la o valoare numerica.

Valid?	**Statement**
Yes	v_new_date DATE := '13-OCT-2015';
No	v_new_date DATE := 'Yesterday';
Yes	v_my_number NUMBER := '123';
No	v_my_number NUMBER := 'abc';

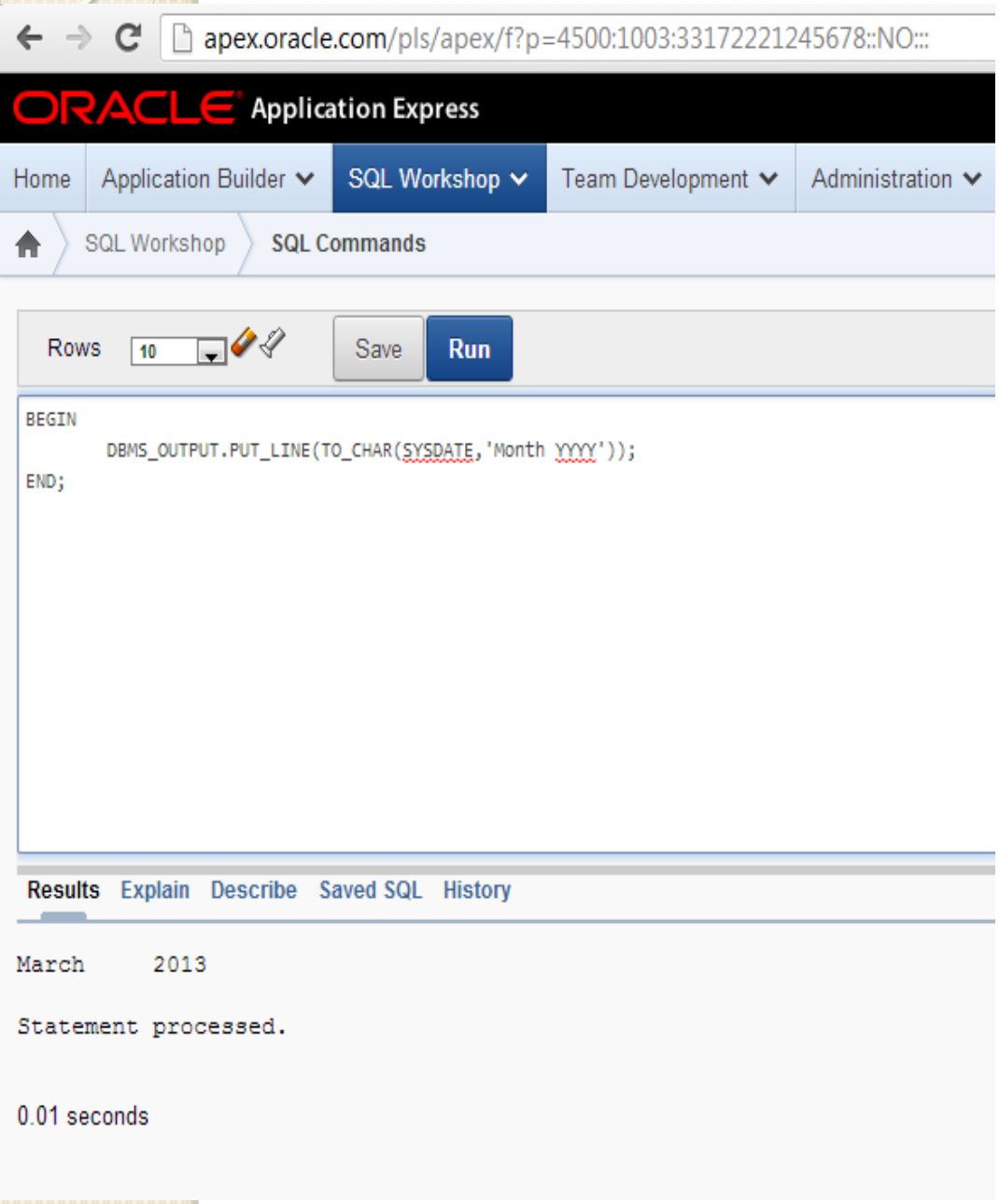
2. Conversii de tipuri de date

2. Conversii explicite

Conversiile explicite convertesc valori de la un tip de date la altul cu ajutorul *functiilor built-in*.

Exemple de functii de conversie:

TO_NUMBER()	**ROWIDTONCHAR()**
TO_CHAR()	**HEXTORAW()**
TO_CLOB()	**RAWTOHEX()**
CHARTOROWID()	**RAWTONHEX()**
ROWIDTOCHAR()	**TO_DATE()**



TO_CHAR

```
BEGIN
  DBMS_OUTPUT.PUT
  _LINE(TO_CHAR(SY
  SDATE,'Month
  YYYY'));
END;
```

Rezultat:

March 2013

TO_DATE

```
BEGIN
   DBMS_OUTPUT.PUT_LINE(to_date('20130315',
   'yyyymmdd'));
END;
```

Rezultat:

03/15/2013

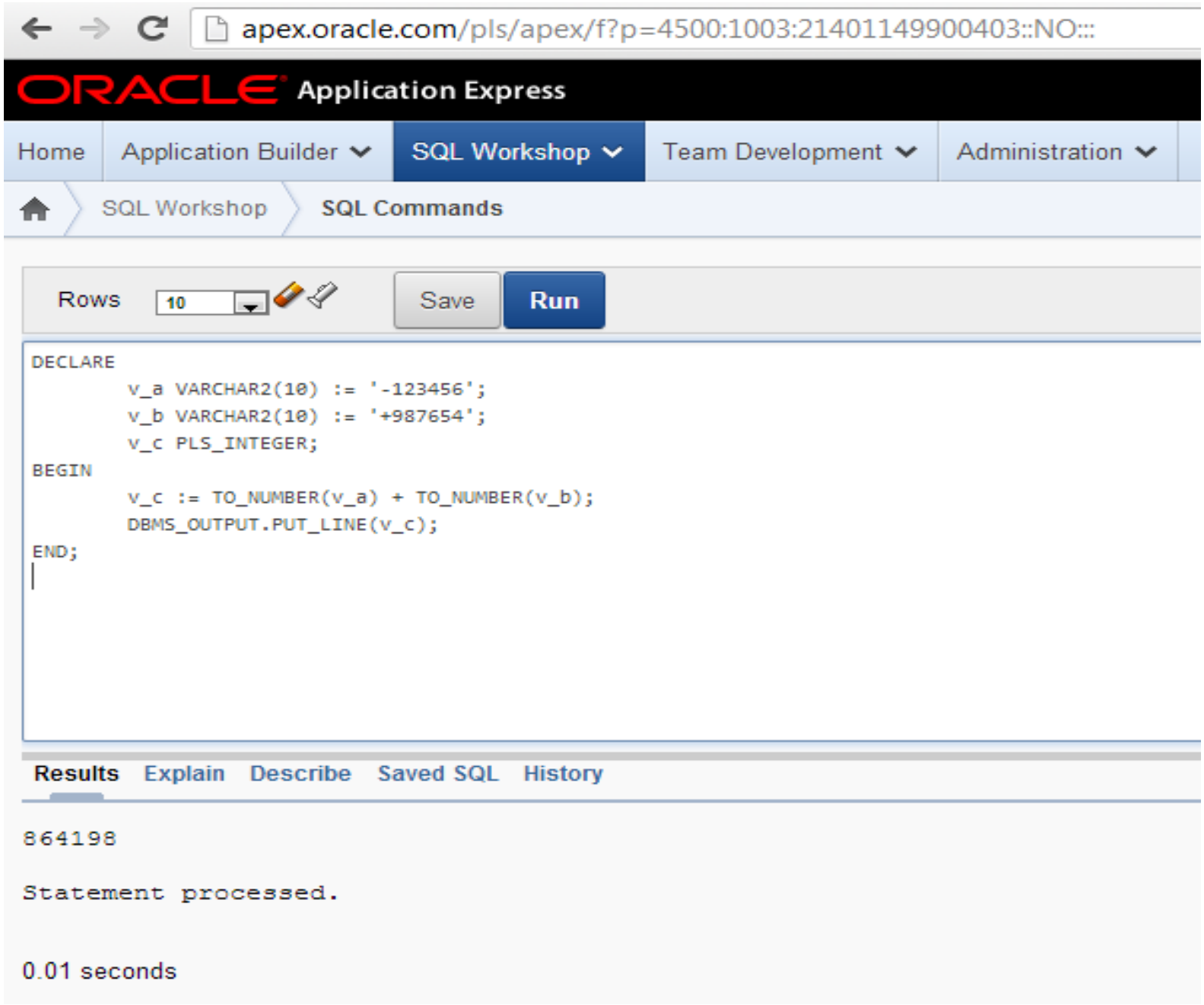
2. Conversii de tipuri de date

TO_NUMBER

```
DECLARE
   v_a VARCHAR2(10) := '-123456';
   v_b VARCHAR2(10) := '+987654';
   v_c PLS_INTEGER;
BEGIN
   v_c := TO_NUMBER(v_a) + TO_NUMBER(v_b);
   DBMS_OUTPUT.PUT_LINE(v_c);
END;
```

Rezultat:

864198


apex.oracle.com/pls/apex/f?p=4500:1003:21401149900403::NO:::
ORACLE Application Express
Home
Application Builder
SQL Workshop
Team Development
Administration
SQL Workshop
SQL Commands
Rows
10
Save
Run
DECLARE
v_a VARCHAR2(10) := '-123456';
v_b VARCHAR2(10) := '+987654';
v_c PLS_INTEGER;
BEGIN
v_c := TO_NUMBER(v_a) + TO_NUMBER(v_b);
DBMS_OUTPUT.PUT_LINE(v_c);
END;
Results
Explain
Describe
Saved SQL
History
864198
Statement processed.
0.01 seconds

Cuprins

1. Functiile SQL in PL/SQL
2. Conversii de tipuri de date
3. Operatori in PL/SQL
4. Blocuri imbricate si vizibilitatea variabilelor
5. Domeniul de aplicare al variabilelor
6. Variabile locale si globale
7. Domeniul de aplicare a exceptiilor in blocurile imbricate

3. Operatori in PL/SQL

Operatori in PL/SQL:

1. **Logici**
2. **Aritmetici**
3. **De concatenare**
4. **Parantezele care controleaza ordinea operatiilor**
5. **Operatorul exponential (**)**

Operatorii dintr-o expresie se executa intr-o anumita ordine, in functie de prioritatea lor.

Tabelul urmator prezinta ordinea implicita a operatorilor de la prioritatea cea mai mare la cea mai mica.

Operator	Operatie
`**`	Ridicare la putere
`*, /`	Inmultire, impartire
`+, -, ||`	Adunare, scadere, concatenare
`=,<,>,<=,>=,<>,!=,~=,^=, IS NULL,LIKE,BETWEEN,IN`	Comparatie
`NOT`	Negatie logica
`AND`	Conjunctie
`OR`	Disjunctie

3. Operatori in PL/SQL

Exemple

Incrementarea contorului dintr-o bucla

v_loop_count := v_loop_count + 1;

Setarea unei valori a unui flag boolean

v_good_salary := v_sal BETWEEN 50000 AND 150000;

Verificarea daca ID-ul unui angajat contine o valoare

v_valid := (v_empno IS NOT NULL);

Cuprins

1. Functiile SQL in PL/SQL
2. Conversii de tipuri de date
3. Operatori in PL/SQL
4. Blocuri imbricate si vizibilitatea variabilelor
5. Domeniul de aplicare al variabilelor
6. Variabile locale si globale
7. Domeniul de aplicare a exceptiilor in blocurile imbricate

4. Blocuri imbricate si vizibilitatea variabilelor

- Un bloc mare, complex este greu de inteles.
- Îl putem imparti in blocuri mai mici care sunt imbricate unele in altele, facand codul mai usor de inteles si de corectat.
- *Atunci cand imbricam blocuri, variabilele declarate pot sa nu mai fie valabile, aceasta depinzand de vizibilitatea lor si locul unde sunt declarate*.
- Puteti face ca variabilele invizibile sa devina valabile prin utilizarea etichetelor.

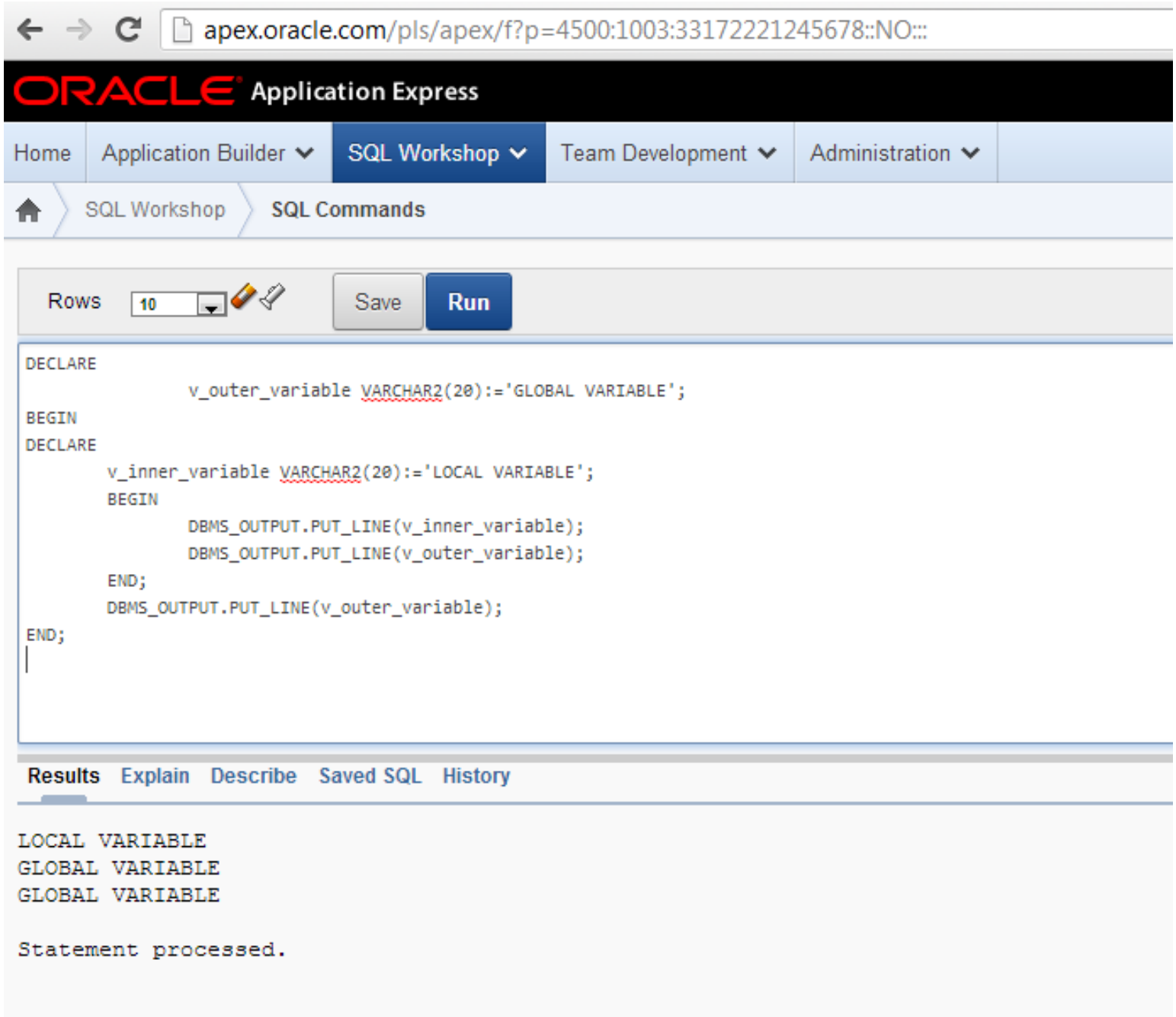
4. Blocuri imbricate si vizibilitatea variabilelor

Blocuri imbricate

- *PL/SQL este un limbaj care are la baza blocurile.*
- Unitatile de baza (proceduri, functii si blocuri anonime) sunt blocurile, care pot contine oricate subblocuri imbricate.
- Fiecare bloc logic corespunde unei probleme de rezolvat.
- Urmatorul exemplu are un bloc exterior (parinte) si un bloc imbricat (copil).
- Variabila ***v_outer_variable*** este declarata in blocul exterior si variabila ***v_inner_variable*** este declarata in blocul interior.

4. Blocuri imbricate si vizibilitatea variabilelor

```
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;
```


apex.oracle.com/pls/apex/f?p=4500:1003:33172221245678::NO:::
ORACLE Application Express
Home
Application Builder
SQL Workshop
Team Development
Administration
SQL Workshop
SQL Commands
Rows
10
Save
Run
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;
Results
Explain
Describe
Saved SQL
History
LOCAL VARIABLE
GLOBAL VARIABLE
GLOBAL VARIABLE
Statement processed.

Cuprins

1. Functiile SQL in PL/SQL
2. Conversii de tipuri de date
3. Operatori in PL/SQL
4. Blocuri imbricate si vizibilitatea variabilelor
5. Domeniul de aplicare al variabilelor
6. Variabile locale si globale
7. Domeniul de aplicare a exceptiilor in blocurile imbricate

5. Domeniul de aplicare al variabilelor

- *Domeniul de aplicabilitate al unei variabile este blocul sau blocurile in care variabila este accesibila, poate fi numita si utilizata.*

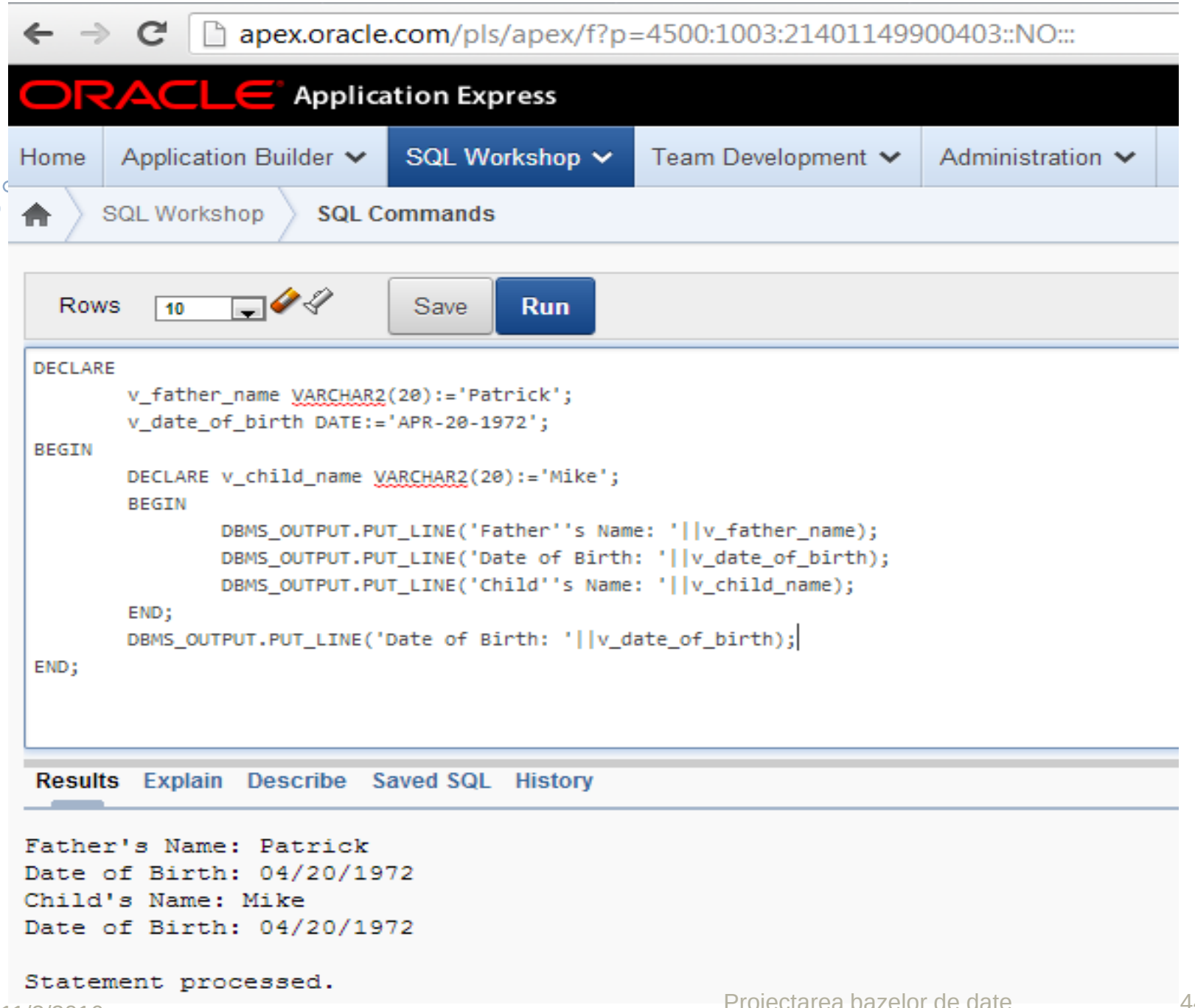
- In *PL/SQL domeniul de vizibilitate a unei variabile este blocul in care este declarata si toate blocurile imbricate in blocul declarativ*.

5. Domeniul de aplicare al variabilelor

Care este domeniul de vizibilitate al fiecarei variabile?

```
DECLARE
   v_father_name VARCHAR2(20):='Patrick';
   v_date_of_birth DATE:='APR-20-1972';

BEGIN
   DECLARE v_child_name VARCHAR2(20):='Mike';
   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;
```


apex.oracle.com/pls/apex/f?p=4500:1003:21401149900403::NO:::
ORACLE Application Express
Home
Application Builder
SQL Workshop
Team Development
Administration
SQL Workshop
SQL Commands
Rows
10
Save
Run
DECLARE
v_father_name VARCHAR2(20):='Patrick';
v_date_of_birth DATE:='APR-20-1972';
BEGIN
DECLARE v_child_name VARCHAR2(20):='Mike';
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;
Results
Explain
Describe
Saved SQL
History
Father's Name: Patrick
Date of Birth: 04/20/1972
Child's Name: Mike
Date of Birth: 04/20/1972
Statement processed.

Cuprins

1. Functiile SQL in PL/SQL
2. Conversii de tipuri de date
3. Operatori in PL/SQL
4. Blocuri imbricate si vizibilitatea variabilelor
5. Domeniul de aplicare al variabilelor
6. Variabile locale si globale
7. Domeniul de aplicare a exceptiilor in blocurile imbricate

6. Variabile locale si globale

Variabilele declarate intr-un bloc PL/SQL sunt considerate locale in acel bloc si globale pentru toate subblocurile lui.

v_outer_variable este locala pentru blocul exterior, dar globala pentru blocul interior.

Mod de functionare:

1. Cand accesam aceasta variabila in blocul interior, PL/SQL cauta mai intai o variabila locala in blocul interior cu acel nume.
2. Daca nu este nici o variabila cu acel nume, PL/SQL cauta variabila in blocul exterior.

6. Variabile locale si globale

```
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;
```

6. Variabile locale si globale

- Variabila ***v_inner_variable*** este locala blocului interior si nu este globala deoarece blocul interior nu are alte blocuri imbricate.
- Aceasta variabila poate fi accesata doar in blocul interior.
- Daca PL/SQL nu gaseste variabila declarata local atunci cauta in sus in partea declarativa a blocului parinte.
- PL/SQL nu cauta in jos blocurile copii.

6. Variabile locale si globale

```
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;
```

6. Variabile locale si globale

- Variabilele ***v_father_name*** si ***v_date_of_birth*** sunt declarate in blocul exterior.

Sunt locale blocului exterior si globale celui interior.

Domeniul lor de aplicare include ambele blocuri.

```
DECLARE
   v_date_of_birth DATE:='APR-20-1972';
   v_father_name VARCHAR2(20):='Patrick';
BEGIN
   DECLARE
        v_child_name VARCHAR2(20):='Mike';
```

6. Variabile locale si globale

- Variabila ***v_child_name*** este declarata in blocul interior (cel imbricat).
- Aceasta variabila este accesibila doar in blocul imbricat si nu este accesibila in blocul exterior.

a) Denumirea variabilelor

- *Nu putem declara doua variabile cu acelasi nume in acelasi bloc.*
- Oricum, putem declara variabile cu acelasi nume in doua blocuri diferite (blocuri imbricate).
- Cele doua elemente reprezentate prin acelasi nume sunt distincte si orice modificare a unuia nu afecteaza pe celalalt.

6. Variabile locale si globale

- **b) Vizibilitatea variabilelor**

Ce se intampla daca acelasi nume este folosit pentru doua variabile, cate una in fiecare bloc?

In exemplul urmator, variabila ***v_date_of_birth*** este declarata de doua ori:

```
DECLARE
   v_father_name VARCHAR2(20):='Patrick';
   v_date_of_birth DATE:='APR-20-1972';
BEGIN
   DECLARE
        v_child_name VARCHAR2(20):='Mike';
        v_date_of_birth DATE:='Dec-12-2002';
   BEGIN
   DBMS_OUTPUT.PUT_LINE('Date of Birth:'
   ||v_date_of_birth);
```

Care ***v_date_of_birth*** este referita de instructiunea DBMS_OUTPUT.PUT_LINE?

Vizibilitatea unei variabile este portiunea de program unde variabila poate fi accesata fara a folosi unui calificativ.

Care este vizibilitatea fiecarei variabile?

```
DECLARE
   v_father_name VARCHAR2(20):='Patrick';
   v_date_of_birth DATE:='Apr-20-1972';
BEGIN
   DECLARE
         v_child_name  VARCHAR2(20):='Mike';
         v_date_of_birth DATE:='Dec-12-2002';
   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;
```

6. Variabile locale si globale

- Variabila ***v_date_of_birth*** declarata in blocul exterior are domeniul de aplicabilitate si in blocul interior.
- Aceasta variabila este vizibila in blocul exterior.
- Oricum ea nu este vizibila in blocul interior deoarece acesta are o variabila cu acelasi nume.
- Variabila ***v_father_name*** este vizibila atat in blocul interior cat si in cel exterior.
- Variabila ***v_child_name*** este vizibila doar in blocul interior.

6. Variabile locale si globale

```
DECLARE
        v_father_name VARCHAR2(20):='Patrick';
        v_date_of_birth DATE:='Apr-20-1972';

BEGIN
        DECLARE
        v_child_name VARCHAR2(20):='Mike';
        v_date_of_birth DATE:='Dec-12-2002';
```

6. Variabile locale si globale

- **c) Calificarea unui identificator**

Un calificator este o eticheta data unui bloc.

Putem folosi acest calificativ pentru a accesa variabilele care au domeniu dar nu sunt vizibile.

In urmatorul exemplu blocul exterior are eticheta **<<*outer*>>.**

```
<<outer>>
DECLARE
   v_father_name VARCHAR2(20):='Patrick';
   v_date_of_birth DATE:='Apr-20-1972';
BEGIN
  DECLARE
    v_child_name VARCHAR2(20):='Mike';
    v_date_of_birth DATE:='Dec-12-2002';
```

Etichetele nu se pun doar blocului exterior. Se pot pune oricarui bloc.

Folosind eticheta outer pentru a califica identificatorul ***v_date_of_birth***, putem afisa acum data de nastere a tatalui in blocul interior.

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

6. Variabile locale si globale

Se va afisa:

Father’s Name: Patrick
Date of Birth: 04/20/1972
Child’s Name: Mike
Date of Birth: 12/12/2002

Statement processed.

Cuprins

1. Functiile SQL in PL/SQL
2. Conversii de tipuri de date
3. Operatori in PL/SQL
4. Blocuri imbricate si vizibilitatea variabilelor
5. Domeniul de aplicare al variabilelor
6. Variabile locale si globale
7. Domeniul de aplicare a exceptiilor in blocurile imbricate

7. Domeniul de aplicare a exceptiilor in blocurile imbricate

O exceptie este o *sectiune de programare care captureaza erorile pentru a opri brusc programul*.

Se poate administra o exceptie prin:

1. Manipularea ei („prinderea ei in cursa”) in blocul in care apare
2. Propagarea ei in mediul apelant

7. Domeniul de aplicare a exceptiilor in blocurile imbricate

1. Prinderea in cursa a exceptiilor cu un handler

- Este bine sa includem o sectiune de exceptii intr-un program PL/SQL.
- Daca exceptia apare in partea executabila a unui bloc prelucrarea este tratata de catre handler-ul de exceptie corespunzator sectiunii de exceptii din acelasi bloc.
- Daca PL/SQL trateaza cu succes exceptia, atunci exceptia nu se propaga in blocul exterior.
- Blocul PL/SQL se incheie cu succes.

Manipularea exceptiilor intr-un bloc interior

In urmatorul exemplu survine o eroare in timpul executiei blocului interior.

Sectiunea EXCEPTION a blocului interior rezolva exceptia cu succes.

Blocul exterior continua executia in mod obisnuit.

```
BEGIN -- outer block
……..……
BEGIN -- inner block
…….  -- exception_name occurs here
…….
EXCEPTION
WHEN exception_name THEN -- handled here
……..
END; -- inner block terminates successfully
…… -- outer block continues execution
END;
```

2. Propagarea exceptiilor catre un bloc exterior

- In cazul in care apare o exceptie in sectiunea executabila a blocului interior si nu este nici un handler de exceptie corespunzator, blocul PL/SQL se incheie cu insucces si exceptia este propagata in blocul imediat exterior.
- In acest exemplu apare o eroare in timpul executiei blocului interior.
- Sectiunea EXCEPTION a blocului interior nu rezolva exceptia.
- Blocul interior se incheie fara succes si PL/SQL transmite exceptia blocului exterior.
- Sectiunea EXCEPTION a blocului exterior manipuleaza cu succes exceptia.

7. Domeniul de aplicare a exceptiilor in blocurile imbricate

```
BEGIN -- outer block
……
BEGIN -- inner block
…….  – exception_name occurs here
…….
END; -- inner block terminates unsuccessfully
…….. -- Remaining code in outer block's executable
……  -- section is skipped
EXCEPTION
WHEN exception_name THEN – outer block handles
   the exception
……
END;
```

Propagarea exceptiilor intr-un subbloc

- Daca PL/SQL intalneste o exceptie si blocul curent nu are un handler pentru aceasta exceptie, exceptia se propaga in blocul imediat exterior pana cand gaseste un handler.
- Atunci cand exceptia se propaga in blocul exterior, actiunile executabile ramase in acel bloc sunt ignorate.
- Un avantaj al acestui lucru este ca puteti sa adaugati instructiuni care necesita propriile manipulari de erori, in blocurile proprii.
- Daca nici unul dintre aceste blocuri nu rezolva exceptia atunci apare o exceptie netratata in mediul gazda (de exemplu Application Express).

Întrebări?