TEMĂ

În această temă voi prezenta cazul când SELECT INTO nu declanșează eroarea NO_DATA_FOUND, dar și cel când o declanșează, cazul când utilizarea unui element dintr-un tabel indexat, care nu a fost inițializat, declanșează eroarea NO_DATA_FOUND, precum și rezolvarea acestuia pentru a nu declanșa excepția. De asemenea voi preciza care variantă pentru comandă update din Cursul 2 este optimă pentru utilizare.

Captarea excepției NO DATA FOUND pentru SELECT INTO VARIABILE.

Să se salveze și să se afișeze datele: numele, prenumele, salariul și manager-ul angajatului pentru care salariul este egal cu v_emp_salary = 1337.

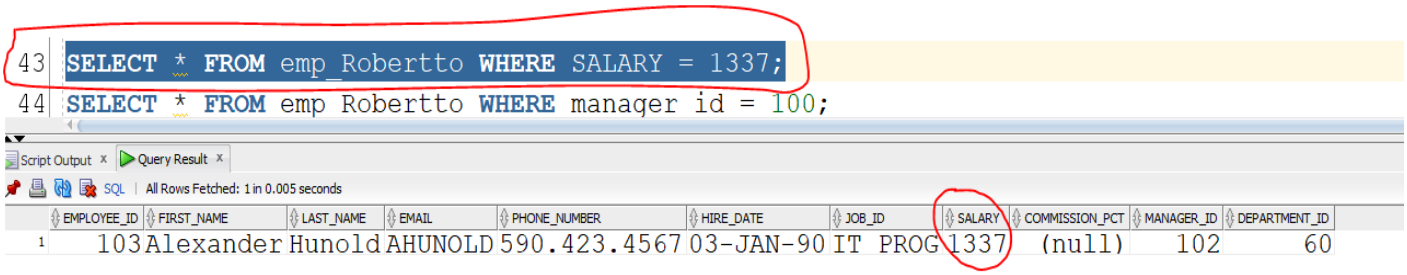
Rezolvare:

```
DECLARE
   v_emp_first_name   VARCHAR2(30);
   v_emp_last_name    VARCHAR2(30);
   v_emp_manager      NUMBER(10) := 1337;
   v_emp_salary       NUMBER(30) := 1337;
   TYPE tab_ind IS TABLE OF emp_Robertto.salary%TYPE
      INDEX BY PLS_INTEGER;
   t tab_ind;
BEGIN
   DBMS_OUTPUT.PUT_LINE('<< Incerc sa fac select into variabile >>');
   SELECT first_name, last_name, manager_id, salary
   INTO v_emp_first_name, v_emp_last_name, v_emp_manager, v_emp_salary
   FROM emp_Robertto
   WHERE salary = v_emp_salary;
   DBMS_OUTPUT.PUT_LINE('<< Am iesit cu succes din select into! >>');
   DBMS_OUTPUT.PUT_LINE('Afisez datele din variabile: ');
   DBMS_OUTPUT.PUT_LINE(v_emp_first_name || ' ' || v_emp_last_name || ' ' || v_emp_manager || '
' || v_emp_salary);
```

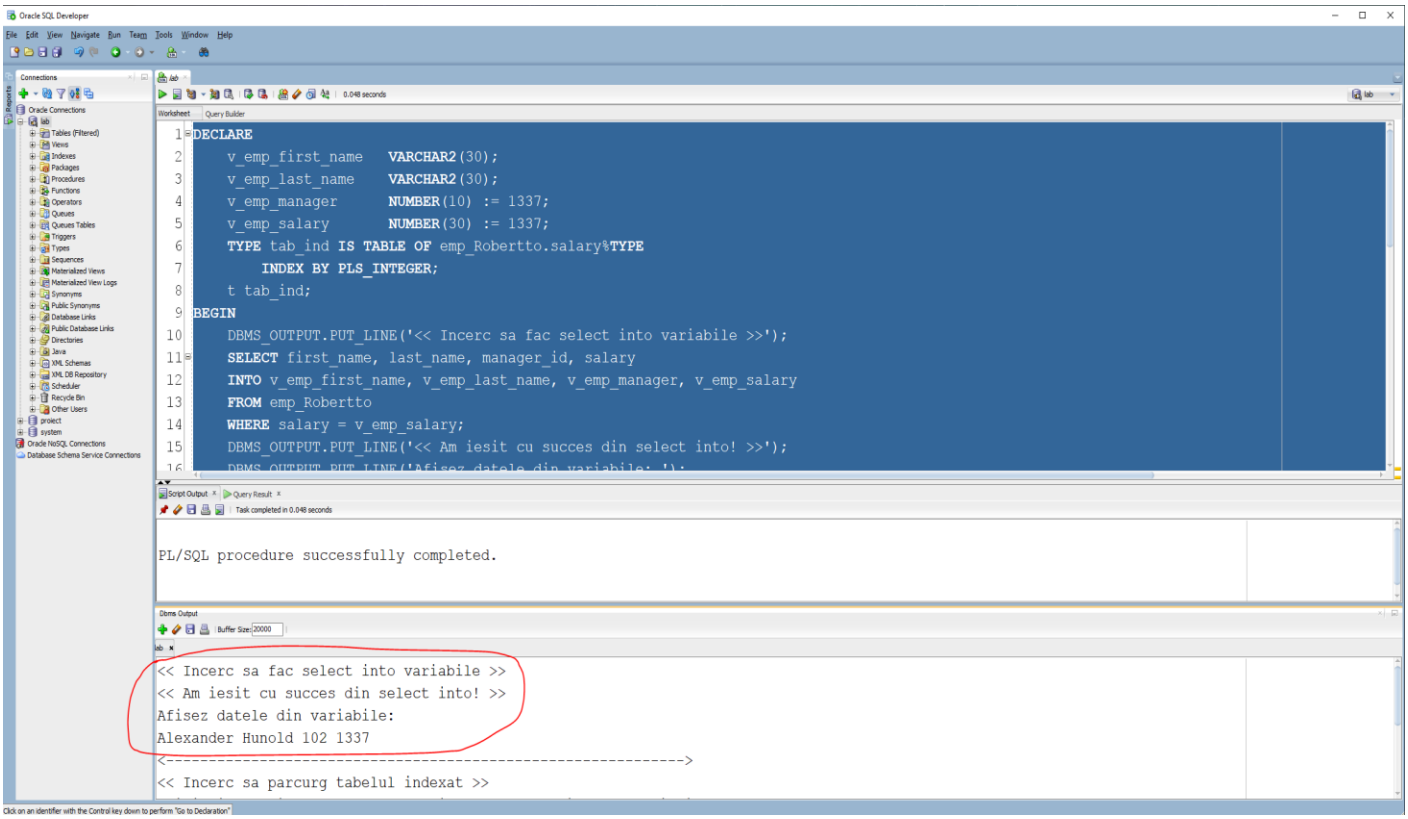
```
   DBMS_OUTPUT.PUT_LINE('<---------------------------------------------------------->');

   DBMS_OUTPUT.PUT_LINE('<< Incerc sa parcurg tabelul indexat >>');

   DBMS_OUTPUT.PUT_LINE('Initializez primele 10 elemente din tabelul meu indexat cu indicele sau la
puterea a doua.');

   DBMS_OUTPUT.PUT_LINE('Acestea sunt:');

   FOR i IN 1..10 LOOP

      t(i) := power(i, 2);

      DBMS_OUTPUT.PUT(t(i) || ' ');

   END LOOP;

   DBMS_OUTPUT.PUT_LINE('');

--    DBMS_OUTPUT.PUT_LINE(t(11));

--    t(11) := 11;

   DBMS_OUTPUT.PUT_LINE('<< Incerc sa actualizez salariul pentru toti angajatii care au managerul cu
id-ul 100, cu elementul de pe pozitia 11 din tabelul meu indexat. >>');

   UPDATE emp_Robertto

   SET salary = t(11)

   WHERE manager_id = 100;

   DBMS_OUTPUT.PUT_LINE('<< Am actualizat cu succes liniile din tabel pentru angajatii cu manager_id
= 100! >>');

EXCEPTION

   WHEN too_many_rows THEN

      DBMS_OUTPUT.PUT_LINE('EROARE: TOO_MANY_ROWS!');

   WHEN no_data_found THEN

      DBMS_OUTPUT.PUT_LINE('EROARE: NO_DATA_FOUND!');

   WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('ALTA EROARE!');

END;
```

Print-Screen:

Pentru început vom verifica dacă în tabelul nostru există un singur angajat cu salariul 1337.



Observăm că avem un singur angajat cu salariul egal cu 1337. Acum încercăm să îi salvăm datele lui în variabile și să le afișăm.

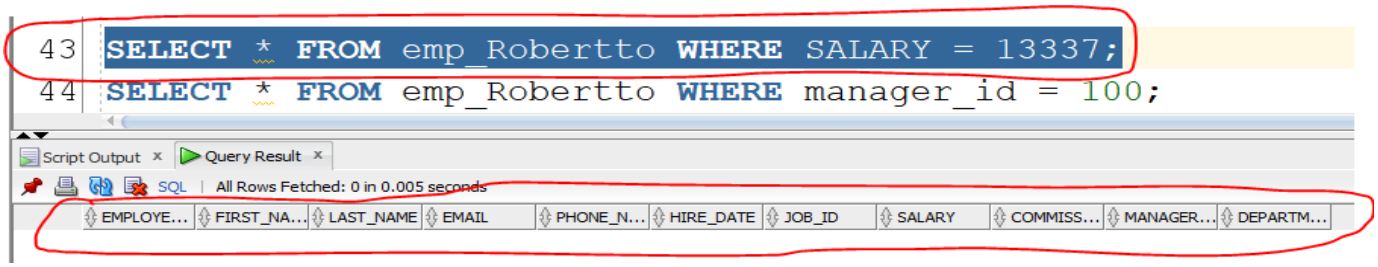


Codul nostru a rulat a compilat cu succes (a salvat datele angajatului cu salariul 1337 și le-a afișat). Deci pentru acest SELECT INTO... nu vom intra pe excepția ***NO_DATA_FOUND!***

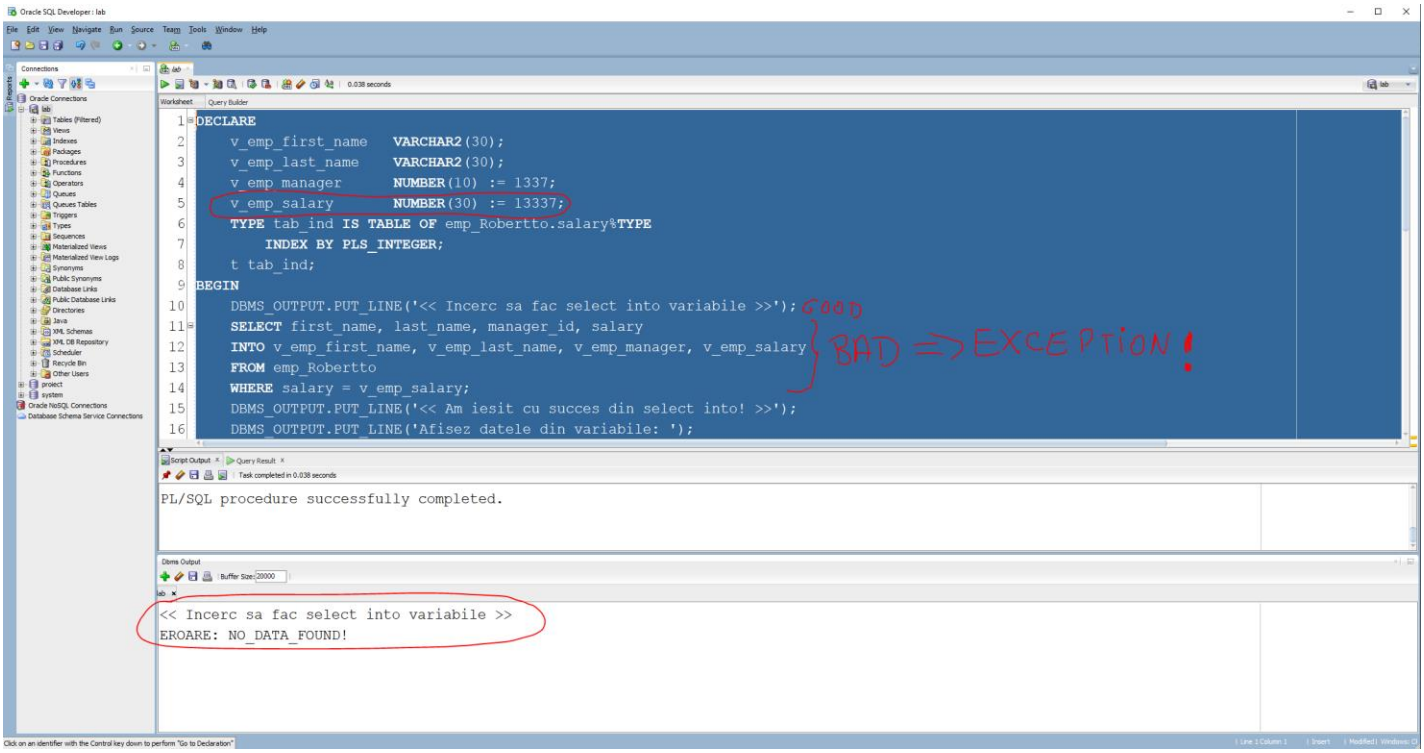
Cum ne-am dat seama ca SELECT INTO... s-a executat cu succes? Am afișat diferite mesaje după acesta (linia 15, 16 etc..). Dacă s-ar fi declanșat excepția ***NO_DATA_FOUND***, programul s-ar fi oprit la linia 11 (unde începe SELECT INTO-ul nostru).

Acum vom face în așa fel încât acest SELECT INTO... să declanșeze excepția ***NO_DATA_FOUND!*** Cum vom proceda? Vom modifica valoarea din variabila *v_emp_salary* din 1337 în 13337 din **DECLARE**.

Verificăm cu un *select * from emp_Robertto where salary = 13337* pentru a fi siguri ca nu există vreun angajat cu acest salariu.



Nu avem niciun angajat cu salariul egal cu 13337!



Am modificat valoarea din **DECLARE** pentru variabile *v_emp_salary* din 1337 în 13337.

➢ *Ce observăm?*

Când rulăm blocul PL/SQL observăm că se execută linia 10 cu succes, dar când încearcă să salveze datele angajatului cu salariul 13337 în variabilele noastre se declanșează excepția ***NO_DATA_FOUND!*** (ajunge pe liniile 11, 12, 13, respectiv 14), **după care restul liniilor nu se mai execută!**

➢ *Care este cauza?*

Nu avem niciun angajat pentru care salary = v_emp_salary = 13337, deci nu se poate îndeplini filtrarea după salary. Rezultând astfel eroarea ***NO_DATA_FOUND***, care am tratat-o cu o excepție (în cazul nostru am afișat pe ecran 'EROARE: NO_DATA_FOUND!')

Dacă s-ar fi putut realiza filtrarea cu succes după salary, s-ar fi afișat în continuare că ***'<< Am iesit cu succes din select into! >>'*** (se execută linia 15), după care valoarea variabilelor pe ecran (linia 16) etc..

Captarea excepției NO DATA FOUND pentru o colecție (tabel indexat).

Pentru a verifica NO_DATE_FOUND pe tabelul nostru indexat vom lasa valoarea pentru variabila noastră v_emp_salary = 1337. De ce 1337 și nu 13337? Pentru a nu declanșa excepția NO_DATA_FOUND pentru SELECT INTO… VARIABILE.

Să se initializeze primele 10 elemente din tabelul indexat cu indicele acestuia la puterea a 2-a, după care să se afișeze valoarea acestora. După care să se modifice salariul actual al angajaților din tabelul emp_Robertto cu al 11-lea element din tabelul meu indexat pentru care manager_id = 100.

Rezolvare:

```
DECLARE
   v_emp_first_name   VARCHAR2(30);
   v_emp_last_name    VARCHAR2(30);
   v_emp_manager      NUMBER(10) := 1337;
   v_emp_salary       NUMBER(30) := 1337;
   TYPE tab_ind IS TABLE OF emp_Robertto.salary%TYPE
      INDEX BY PLS_INTEGER;
   t tab_ind;
BEGIN
   DBMS_OUTPUT.PUT_LINE('<< Incerc sa fac select into variabile >>');
   SELECT first_name, last_name, manager_id, salary
   INTO v_emp_first_name, v_emp_last_name, v_emp_manager, v_emp_salary
   FROM emp_Robertto
   WHERE salary = v_emp_salary;
   DBMS_OUTPUT.PUT_LINE('<< Am iesit cu succes din select into! >>');
   DBMS_OUTPUT.PUT_LINE('Afisez datele din variabile: ');
   DBMS_OUTPUT.PUT_LINE(v_emp_first_name || ' ' || v_emp_last_name || ' ' || v_emp_manager || '
' || v_emp_salary);
```

```
   DBMS_OUTPUT.PUT_LINE('<---------------------------------------------------------->');

   DBMS_OUTPUT.PUT_LINE('<< Incerc sa parcurg tabelul indexat >>');

   DBMS_OUTPUT.PUT_LINE('Initializez primele 10 elemente din tabelul meu indexat cu indicele sau la
puterea a doua.');

   DBMS_OUTPUT.PUT_LINE('Acestea sunt:');

   FOR i IN 1..10 LOOP

      t(i) := power(i, 2);

      DBMS_OUTPUT.PUT(t(i) || ' ');

   END LOOP;

   DBMS_OUTPUT.PUT_LINE('');

--    DBMS_OUTPUT.PUT_LINE(t(11));

--    t(11) := 11;

   DBMS_OUTPUT.PUT_LINE('<< Incerc sa actualizez salariul pentru toti angajatii care au managerul cu
id-ul 100, cu elementul de pe pozitia 11 din tabelul meu indexat. >>');

   UPDATE emp_Robertto

   SET salary = t(11)

   WHERE manager_id = 100;

   DBMS_OUTPUT.PUT_LINE('<< Am actualizat cu succes liniile din tabel pentru angajatii cu manager_id
= 100! >>');

EXCEPTION

   WHEN too_many_rows THEN

      DBMS_OUTPUT.PUT_LINE('EROARE: TOO_MANY_ROWS!');

   WHEN no_data_found THEN

      DBMS_OUTPUT.PUT_LINE('EROARE: NO_DATA_FOUND!');

   WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('ALTA EROARE!');

END;
```

Print-Screen:

Prima dată vom verifica dacă există cel puțin un angajat cu manager_id = 100.

```
43  SELECT * FROM emp_Robertto WHERE SALARY = 13337;
44  SELECT * FROM emp_Robertto WHERE manager_id = 100;
```

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.01 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	13000	(null)	100	20
2	101	Neena	Kochhar	NKOCHHAR	6666	21-SEP-89	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	(null)	100	90
4	114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-94	PU_MAN	11000	(null)	100	30
5	120	Matthew	Weiss	MWEISS	650.123.1234	18-JUL-96	ST_MAN	8000	(null)	100	50
6	121	Adam	Fripp	AFRIPP	650.123.2234	10-APR-97	ST_MAN	8200	(null)	100	50
7	122	Payam	Kaufling	PKAUFLIN	650.123.3234	01-MAY-95	ST_MAN	7900	(null)	100	50
8	123	Shanta	Vollman	SVOLLMAN	650.123.4234	10-OCT-97	ST_MAN	6500	(null)	100	50
9	124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	5800	(null)	100	50
10	145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-96	SA_MAN	14000	0.4	100	80
11	146	Karen	Partners	KPARTNER	011.44.1344.467268	05-JAN-97	SA_MAN	13500	0.3	100	80
12	147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	10-MAR-97	SA_MAN	12000	0.3	100	80
13	148	Gerald	Cambrault	GCAMBRAU	011.44.1344.619268	15-OCT-99	SA_MAN	11000	0.3	100	80
14	149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	10500	0.2	100	80

Observăm că avem 11 angajați cu manager_id = 100.

Acum vom afișa inițializa primele 10 elemente din tabelul nostru indexat cu indicele la puterea a 2-a și le vom afișa pe aceeași linie valoarea acestora, așa cum se cere în cerință.

```
22      FOR i IN 1..10 LOOP
23          t(i) := power(i, 2);
24          DBMS_OUTPUT.PUT(t(i) || ' ');
25      END LOOP;
26      DBMS_OUTPUT.PUT_LINE('');
27  --    DBMS_OUTPUT.PUT_LINE(t(11));
28  --    t(11) := 11;
29      DBMS_OUTPUT.PUT_LINE('<< Incerc sa actualizez salariul pentru toti angajatii care au managerul cu id-ul 100, cu elementul de pe pozitia 11 din tabelul meu
30      UPDATE emp_Robertto
31      SET salary = t(11)
32      WHERE manager_id = 100;
33      DBMS_OUTPUT.PUT_LINE('<< Am actualizat cu succes liniile din tabel pentru angajatii cu manager_id = 100! >>');
34  EXCEPTION
35      WHEN too_many_rows THEN
36          DBMS_OUTPUT.PUT_LINE('EROARE: TOO_MANY_ROWS!');
37      WHEN no_data_found THEN
38          DBMS_OUTPUT.PUT_LINE('EROARE: NO_DATA_FOUND!');
```

BAM ⇒ EXCEPTION!

```
PL/SQL procedure successfully completed.
```

```
<< Incerc sa fac select into variabile >>
<< Am iesit cu succes din select into! >>
Afisez datele din variabile:
Alexander Hunold 102 1337
<------------------------------------------------------------>
<< Incerc sa parcurg tabelul indexat >>
Initializez primele 10 elemente din tabelul meu indexat cu indicele sau la puterea a doua.
Acestea sunt:
1 4 9 16 25 36 49 64 81 100
<< Incerc sa actualizez salariul pentru toti angajatii care au managerul cu id-ul 100, cu elementul de pe pozitia 11 din tabelul meu indexat. >>
EROARE: NO_DATA_FOUND!
```

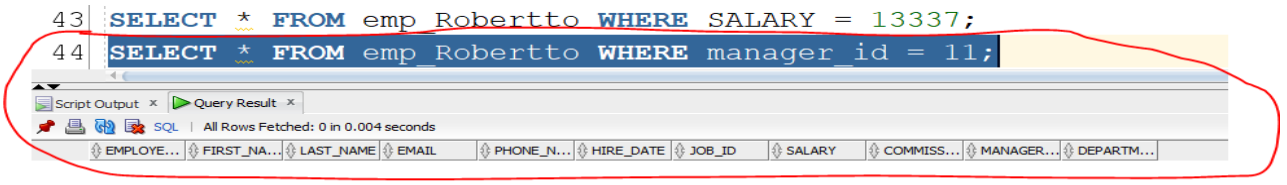
➢ *Ce observăm?*

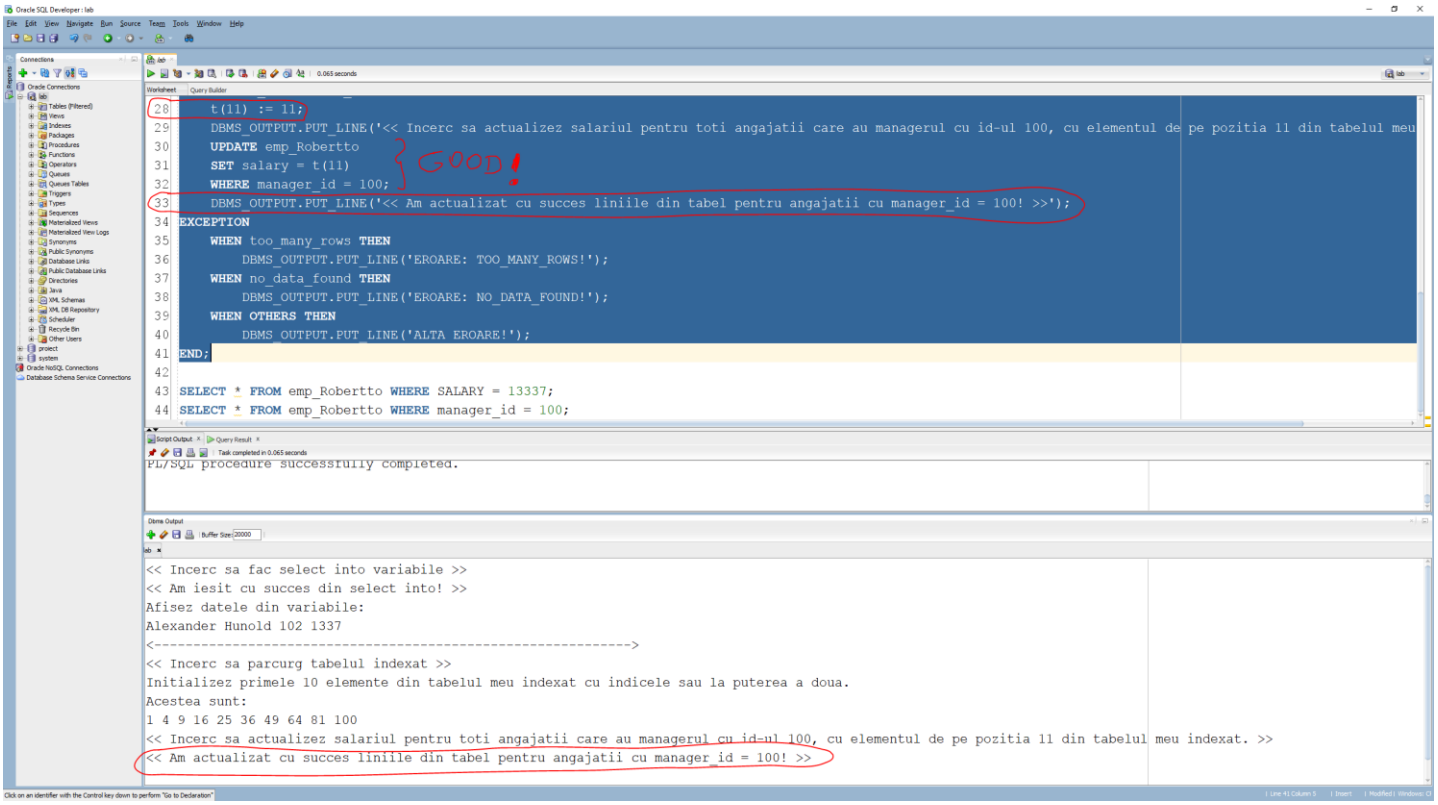
Când am rulat codul, am inițializat și afișat cu succes primele 10 elemente din tabelul nostru indexat, așa cum se cerea mai sus. După care am încercat să actualizăm toate liniile din tabelul nostru pentru care salary = t(11), elementul de pe poziția 11 din tabelul nostru indexat. **Această filtrare nu s-a putut realiza întrucât elementul de pe poziția 11 nu a fost inițializat!** Rezultând astfel eroarea ***NO_DATA_FOUND***, care am tratat-o cu o excepție (în cazul nostru am afișat pe ecran 'EROARE: NO_DATA_FOUND!'). Elementele unui tablou indexat nu sunt inițializate cu o valoare default.

Dacă elementul de pe poziția 11 ar fi fost inițializat înainte de instrucțiunea UPDATE, nu am fi avut eroarea ***NO_DATA_FOUND***.

Pentru a fi siguri de acest lucru vom testa din nou codul, dar vom inițializa înainte de instrucțiunea UPDATE, elementul de pe poziția 11, t(11), cu valoarea 11 și vom verifica printr-un select dacă există vreun angajat cu manager_id = 11.

Verificăm și dacă există vreun angajat cu manager_id = 11. Observăm că nu există un astfel de angajat.





La linia 28 am inițializat elementul de pe poziția 11 cu valoarea 11. După care instrucțiunea UPDATE s-a executat cu succes de data aceasta de ce? S-a efectuat filtrarea după manager_id. Cum ne dăm seama de acest lucru? Observăm că la linia 33 s-a afișat un mesaj pe ecran '**<< Am actualizat cu succes liniile din tabel pentru angajatii cu manager_id = 100! >>**'.

Nu vom primi eroarea ***NO_DATA_FOUND*** chiar dacă în tabelul nostru nu există măcar un angajat cu manager_id = t(11) = 11. De ce? Pentru că practic "noi actualizăm nimic", nu există vreo linie în tabelul nostru emp_Robertto pentru manager_id = 11, deci ele vor rămâne cu aceleași valori.

Care variantă din cursul 2 pe comanda update este optimă pentru a fi utilizată.

Eu consider că varianta 3 este cea mai eficientă având în vedere timpul, dar și spațiul ocupat în memorie. De ce? Pentru că celelalte variante folosesc mai multe comenzi ***UPDATE***, în timp ce această variantă folosește o singură comandă ***UPDATE***. Știm că fiecare accesare/parcurgere a unui tabel este foarte "costisitoare" din punct de vedere al timpului, dar și al memoriei (comanda ***UPDATE*** accesează toate liniile unui tabel), practic se verifică linie cu line.. Noi ne dorim acest lucru să se întâmple de cât mai puține ori (spre exemplu: în cazul în care am vrem să adăugăm o categorie pentru un anumit rând din tabel care respectă anumite proprietăți, este redundant să parcurgem tabelul pentru fiecare proprietate a fiecărei categorii, *WHERE nr_produse BETWEEN 1000 AND 2000 AND id_categorie = 2 THEN 'B'* – 1 parcurgere, *WHERE nr_produse BETWEEN 500 AND 1000 AND id_categorie = 1 THEN 'B'* – 1 parcurgere => 2 parcurgeri în total). Putem adăuga această categorie printr-o singură parcurgere. Acest lucru scoate în evidență faptul că pentru fiecare comandă ***UPDATE*** parcurgem "degeaba" ***numărul total de linii din tabel – numărul total de linii care respectă filtrarea***. Dacă tabelul nostru ar avea mai multe linii, de exemplu 1.000.000.000 linii, iar doar 5 linii ar respecta proprietățile impuse, noi practic am parcurge 999.999.995 "degeaba", acest aspect punând în evidență risipa de timp și memorie. La fiecare parsare a unei comenzi SQL, se fac anumite verificări care sunt destul de "costisitoare", verificarea corectitudinii sintactice, semantice, parcurgerea textului și asociarea unei funcții hash. Folosirea unui CASE este și ea într-adevăr "costisitoare", dar nu la fel de costisitoare precum o parcurge a unui întreg tabel. Un alt motiv pentru care Varianta 3 ar fi mai eficientă în comparație cu celelalte două este că, dacă ne dorim să adăugăm o nouă categorie, noi nu suntem obligați să efectuăm alte comenzi ***UPDATE*** (alte tranzacții), ci doar una singură, cea curentă (efectuăm o singură parcurgere). Cum putem realiza acest lucru? Adăugând un nou ***CASE***.

De asemenea știm de la workshop-ul de optimizare că:

Selectivitate = (numărul de rânduri care satisfac o condiția/condițiile impuse) împărțit la numărul total de rânduri din tabel.

Cardinalitatea = numărul total de rânduri din tabel deînmulțit cu selectivitatea.

Popescu Paullo Robertto Karloss

Grupa 231

Temă SGBD #4