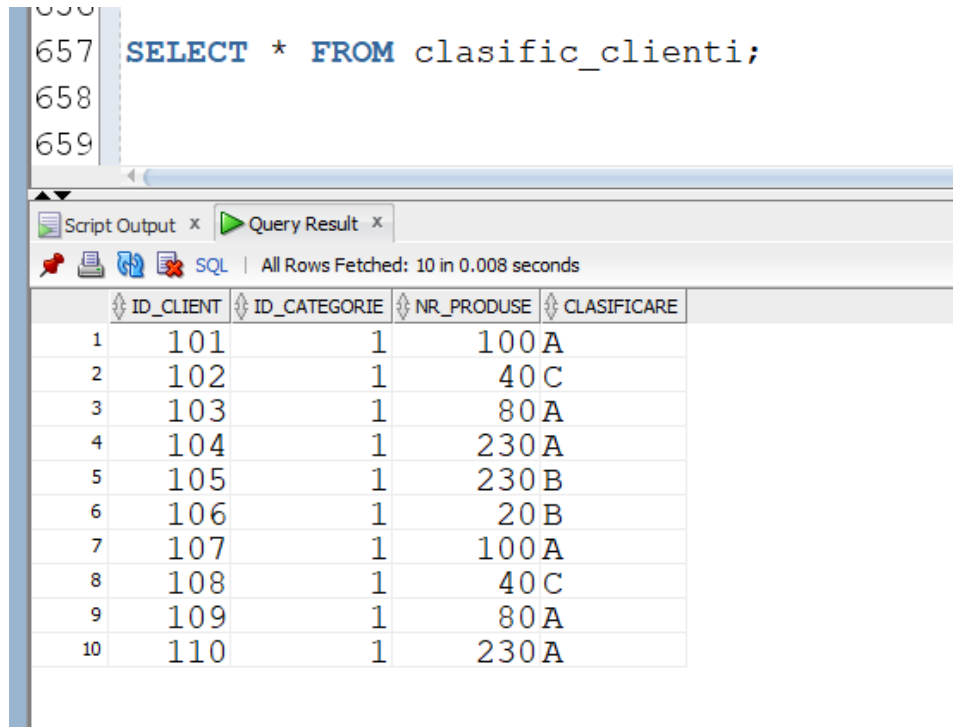


## Tema cursurile 1-3

**Tema 1** – De accesat baza de date prin instructiuni sql executate in Java.

Pentru acest exercitiu am folosit tabela definita in rezolvarea de la tema3 de mai jos, dar cu urmatoarele inregistrari initiale:



```
657 SELECT * FROM clasific_clienti;
658
659
```

	ID_CLIENT	ID_CATEGORIE	NR_PRODUSE	CLASIFICARE
1	101	1	100	A
2	102	1	40	C
3	103	1	80	A
4	104	1	230	A
5	105	1	230	B
6	106	1	20	B
7	107	1	100	A
8	108	1	40	C
9	109	1	80	A
10	110	1	230	A

Am implementat un program in Java care selecteaza toate datele din tabelul clasific\_clienti si face cate un update pentru fiecare client pentru id-ul categoriei si clasificare in functie de numarul de produse avute astfel:

- pentru numarul de produse mai mic sau egal cu 40 id-ul categoriei creste cu 1, iar clasificarea va deveni A
- pentru numarul de produse mai mic sau egal cu 60 id-ul categoriei creste cu 2 (dar mai mare ca 40), iar clasificarea va deveni B
- pentru numarul de produse mai mic sau egal cu 80 id-ul categoriei creste cu 3 (dar mai mare ca 60), iar clasificarea va deveni C
- pentru numarul de produse mai mare ca 80 id-ul categoriei creste cu 4, iar clasificarea va deveni D

Am folosit o conexiune locala si am creat cate un obiect statement pentru a executa query-ul de select si update.

```

package ro.fmi.tema1;
import java.sql.*;

class Main {
    public static void main(String args[]) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");

            Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "c##ana", "parola");

            if(con == null) {
                System.out.println("Nu s-a facut conexiunea");
            }

            Statement stmt_select = con.createStatement();
            Statement stmt_update = con.createStatement();

            ResultSet rs = stmt_select.executeQuery("select * from clasific_clienti");

            while (rs.next()){
                int id_client = rs.getInt(1);
                int id_categorie = rs.getInt(2);
                int nr_produce = rs.getInt(3);
                String clasificare = rs.getString(4);
                String afisare = "Clientul cu id-ul " + id_client + " are id-ul categoriei " + id_categorie + " si clasificarea " +
                clasificare + " si va primi id-ul categoriei ";
                if(nr_produce <= 40){
                    stmt_update.execute("update clasific_clienti set clasificare='A', id_categorie=" + (id_categorie + 1) + "
                    where id_client = " + id_client );
                    afisare += (id_categorie + 1) + " si clasificarea A";
                } else if(nr_produce <= 60){
                    stmt_update.execute("update clasific_clienti set clasificare='B', id_categorie=" + (id_categorie + 2) + "
                    where id_client = " + id_client );
                    afisare += (id_categorie + 2) + " si clasificarea B";
                } else if(nr_produce <= 80){
                    stmt_update.execute("update clasific_clienti set clasificare='C', id_categorie=" + (id_categorie + 3) + "
                    where id_client = " + id_client );
                    afisare += (id_categorie + 3) + " si clasificarea C";
                } else {
                    stmt_update.execute("update clasific_clienti set clasificare='D', id_categorie=" + (id_categorie + 4) + "
                    where id_client = " + id_client );
                    afisare += (id_categorie + 4) + " si clasificarea D";
                }
                System.out.println(afisare);
            }
            stmt_select.close();
            stmt_update.close();
            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

```

```
}
}
```

Print screen cu codul rulat:

```

18 Statement stmt_update = conn.createStatement();
19
20 ResultSet rs = stmt_select.executeQuery( "select * from clasific_clienti");
21
22 while (rs.next()){
23     int id_client = rs.getInt( columnindex: 1);
24     int id_categorie = rs.getInt( columnindex: 2);
25     int nr_produce = rs.getInt( columnindex: 3);
26     String clasificare = rs.getString( columnindex: 4);
27     String afisare = "Clientul cu id-ul " + id_client + " are id-ul categoriei " + id_categorie + " si clasificarea " + clasificare + " si va primi id-ul categoriei ";
28     if(nr_produce <= 40){
29         stmt_update.executeUpdate( "update clasific_clienti set clasificare='A', id_categorie=" + (id_categorie + 1) + " where id_client = " + id_client );
30         afisare += (id_categorie + 1) + " si clasificarea A";
31     } else if(nr_produce <= 80){
32         stmt_update.executeUpdate( "update clasific_clienti set clasificare='B', id_categorie=" + (id_categorie + 2) + " where id_client = " + id_client );
33         afisare += (id_categorie + 2) + " si clasificarea B";
34     } else if(nr_produce <= 100){
35         stmt_update.executeUpdate( "update clasific_clienti set clasificare='C', id_categorie=" + (id_categorie + 3) + " where id_client = " + id_client );
36         afisare += (id_categorie + 3) + " si clasificarea C";
37     } else {
38         stmt_update.executeUpdate( "update clasific_clienti set clasificare='D', id_categorie=" + (id_categorie + 4) + " where id_client = " + id_client );
39         afisare += (id_categorie + 4) + " si clasificarea D";
40     }
41     System.out.println(afisare);
42 }
43 stmt_select.close();
44 stmt_update.close();
45 conn.close();

```

Run Console Output:

```

Clientul cu id-ul 101 are id-ul categoriei 1 si clasificarea A si va primi id-ul categoriei 5 si clasificarea D
Clientul cu id-ul 102 are id-ul categoriei 1 si clasificarea C si va primi id-ul categoriei 2 si clasificarea A
Clientul cu id-ul 103 are id-ul categoriei 1 si clasificarea A si va primi id-ul categoriei 4 si clasificarea C
Clientul cu id-ul 104 are id-ul categoriei 1 si clasificarea A si va primi id-ul categoriei 5 si clasificarea D
Clientul cu id-ul 105 are id-ul categoriei 1 si clasificarea B si va primi id-ul categoriei 5 si clasificarea D
Clientul cu id-ul 106 are id-ul categoriei 1 si clasificarea B si va primi id-ul categoriei 2 si clasificarea A
Clientul cu id-ul 107 are id-ul categoriei 1 si clasificarea A si va primi id-ul categoriei 5 si clasificarea A
Clientul cu id-ul 108 are id-ul categoriei 1 si clasificarea C si va primi id-ul categoriei 2 si clasificarea A
Clientul cu id-ul 109 are id-ul categoriei 1 si clasificarea A si va primi id-ul categoriei 4 si clasificarea C
Clientul cu id-ul 110 are id-ul categoriei 1 si clasificarea A si va primi id-ul categoriei 5 si clasificarea D

```

Dupa cum se poate vedea si in consola, datele dupa o rulare a programului sunt:

Bibliografie:

<https://www.javatpoint.com/example-to-connect-to-the-oracle-database>

657 **SELECT \* FROM clasific\_clienti;**

Script Output x Query Result x

SQL | All Rows Fetched: 11 in 0.002 seconds

	ID_CLIENT	ID_CATEGORIE	NR_PRODUSE	CLASIFICARE
1	101	5	100	D
2	102	2	40	A
3	103	4	80	C
4	104	5	230	D
5	105	5	230	D
6	106	2	20	A
7	107	5	100	D
8	108	2	40	A
9	109	4	80	C
10	110	5	230	D
11	111	5	230	D

<https://www.oracle.com/ro/database/technologies/appdev/jdbc-downloads.html>  
<https://coderanch.com/t/298352/databases/Multiple-statements-single-connection>

**Tema 2** – Comparare sortare scrisa in PL/SQL cu sortare facuta de order by din SQL. (bubble sort)

Codul pentru bubble sort implementat in PL/SQL este:

```
DECLARE
    TYPE va_int IS VARRAY(20) OF PLS_INTEGER;
    v_int va_int;
    v_aux PLS_INTEGER;
    v_gata PLS_INTEGER := 0;
    v_indice PLS_INTEGER := 0;
BEGIN
    v_int := va_int(1,12,9,6,3,31,2,7,9,-2,17,4,0,-32,5,3,1,8,10);

    DBMS_OUTPUT.PUT('Vectorul initial -> [');
    FOR elem IN v_int.FIRST..v_int.LAST LOOP
        IF elem < v_int.LAST THEN
            DBMS_OUTPUT.PUT(v_int(elem) || ', ');
        ELSE
            DBMS_OUTPUT.PUT_LINE(v_int(elem) || ']');
        END IF;
    END LOOP;

    -- bubble sort
    WHILE v_gata = 0 LOOP
        v_gata := 1;
        v_indice := v_indice + 1;
        FOR indice IN v_int.FIRST..v_int.LAST LOOP
            IF indice < v_int.COUNT THEN
                IF v_int(indice) > v_int(indice+1) THEN
                    v_aux := v_int(indice);
                    v_int(indice) := v_int(indice+1);
                    v_int(indice+1) := v_aux;
                    v_gata := 0;
                END IF;
            END IF;
        END LOOP;
    END LOOP;

    DBMS_OUTPUT.PUT('Vectorul sortat -> [');
```

```

FOR indice IN v_int.FIRST..v_int.LAST LOOP
    IF indice < v_int.LAST THEN
        DBMS_OUTPUT.PUT(v_int(indice)||', ');
    ELSE
        DBMS_OUTPUT.PUT_LINE(v_int(indice)||']');
    END IF;
END LOOP;
END;
/

```

Pentru a goli memoria cache si a avea un timp de rulare cat mai apropiat de realitate am folosit urmatoarele comenzi:

```

alter system flush buffer_cache;
alter system flush shared_pool;

```

Pentru un vector de 19 de elemente sortarea in PL/SQL a durat 0.048 secunde.

The screenshot shows the Oracle SQL Developer interface. The top pane displays a PL/SQL script with line numbers 666 to 686. The script defines a VARRAY, initializes it with 19 integers, and then sorts them using a bubble sort algorithm. The bottom pane shows the 'Script Output' window, which displays the output of the script, including the initial and sorted vectors, and a confirmation message.

```

666 TYPE va_int IS VARRAY(20) OF PLS_INTEGER;
667 v_int va_int;
668 v_aux PLS_INTEGER;
669 v_gata PLS_INTEGER := 0;
670 v_indice PLS_INTEGER := 0;
671 BEGIN
672 v_int := va_int(1,12,9,6,3,31,2,7,9,-2,17,4,0,-32,5,3,1,8,10);
673
674 DBMS_OUTPUT.PUT('Vectorul initial -> ');
675 FOR elem IN v_int.FIRST..v_int.LAST LOOP
676 IF elem < v_int.LAST THEN
677 DBMS_OUTPUT.PUT(v_int(elem)||', ');
678 ELSE
679 DBMS_OUTPUT.PUT_LINE(v_int(elem)||']');
680 END IF;
681 END LOOP;
682
683 -- bubble sort
684 WHILE v_gata = 0 LOOP
685 v_gata := 1;
686 v_indice := v_indice + 1;

```

Script Output x Query Result x

Task completed in 0.048 seconds

Vectorul initial -> [1, 12, 9, 6, 3, 31, 2, 7, 9, -2, 17, 4, 0, -32, 5, 3, 1, 8, 10]  
 Vectorul sortat -> [-32, -2, 0, 1, 1, 2, 3, 3, 4, 5, 6, 7, 8, 9, 9, 10, 12, 17, 31]

PL/SQL procedure successfully completed.

Pentru a compara cu SQL voi crea un tabel cu o coloana in care voi introduce elementele vectorului.

```
710  
711 CREATE TABLE vector(  
712     nr NUMBER  
713 );  
714
```

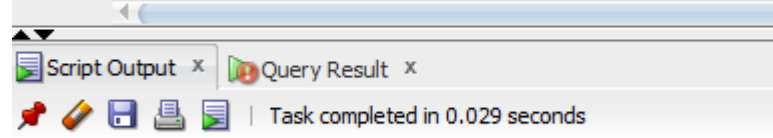
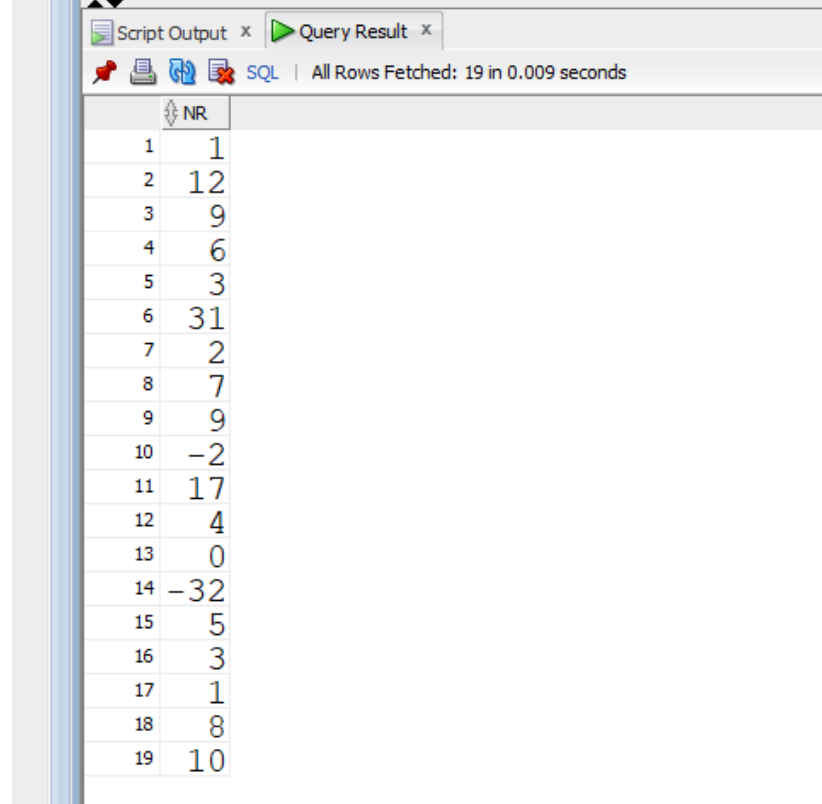


Table VECTOR created.

```
734  
735  
736 SELECT * FROM vector;  
737  
738  
739 SELECT * FROM vector;
```



	NR
1	1
2	12
3	9
4	6
5	3
6	31
7	2
8	7
9	9
10	-2
11	17
12	4
13	0
14	-32
15	5
16	3
17	1
18	8
19	10

Cand sortez dupa prima coloana tabelul nou creat o sa obtin acelasi rezultat ca mai sus in doar 0.004 secunde.

```
735
736 SELECT *
737 FROM vector
738 ORDER BY 1;
739
```

Script Output x Query Result x	
SQL   All Rows Fetched: 19 in 0.004 seconds	
NR	
1	-32
2	-2
3	0
4	1
5	1
6	2
7	3
8	3
9	4
10	5
11	6
12	7
13	8
14	9
15	9
16	10
17	12
18	17
19	31

Deci sortarea cu ORDER\_BY este de aproximativ 12 ori mai rapid ca o implementare de bubble sort in PL/SQL.

Bibliografie:

<https://stackoverflow.com/questions/4093377/pl-sql-bubble-sort>

<https://www.kayshav.com/channels/bubbleSort.php>

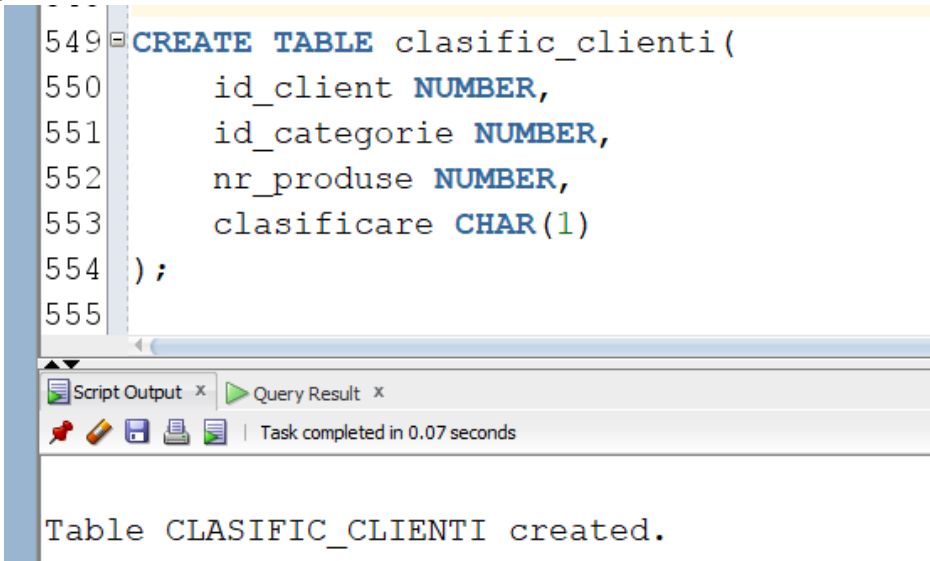
[https://docs.databricks.com/sql/language-manual/functions/array\\_sort.html](https://docs.databricks.com/sql/language-manual/functions/array_sort.html)



**Tema 3** – De implementat exemplul 3.8 din curs pe o tabela existenta precum employees (sau de creat tabela clasific\_clienti) si de tratat exceptiile too\_many\_rows si no\_data\_found.

Am ales sa creez tabela clasific\_clienti cu urmatoarea definitie:

```
CREATE TABLE clasific_clienti(  
    id_client NUMBER,  
    id_categorie NUMBER,  
    nr_produce NUMBER,  
    clasificare CHAR(1)  
);
```

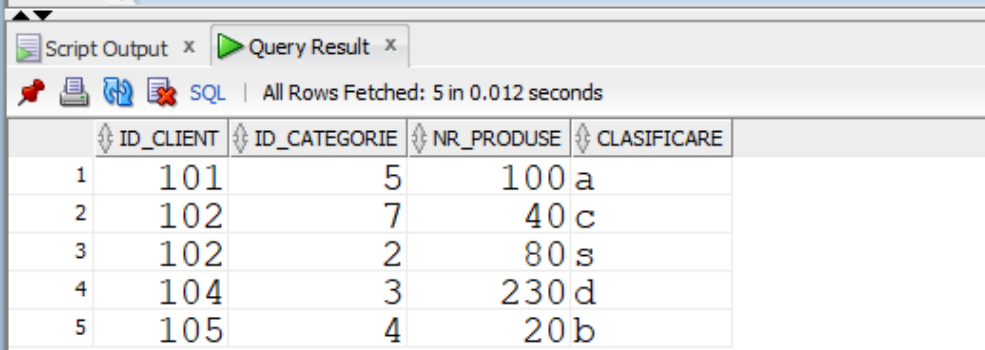


Apoi am introdus niste date de test:

```
INSERT INTO clasific_clienti VALUES (101, 5, 100, 'a');  
INSERT INTO clasific_clienti VALUES (102, 7, 40, 'c');  
INSERT INTO clasific_clienti VALUES (102, 2, 80, 's');  
INSERT INTO clasific_clienti VALUES (104, 3, 230, 'd');  
INSERT INTO clasific_clienti VALUES (105, 4, 20, 'b');
```

Acum comanda SELECT \* FROM clasific\_clienti o sa se afiseze:

```
589  
590 SELECT * FROM clasific_clienti;  
591
```



Codul modificat cat sa trateze cazul in care nu exista niciun client cu id-ul dat si cazul in care exista mai multi client cu acelasi id este:

```
DECLARE
    v_categorie NUMBER;
    v_produce NUMBER;
    v_clasificare CHAR(1);
BEGIN
    DELETE FROM clasific_clienti WHERE id_client=2
    RETURNING id_categorie, nr_produce, clasificare
    INTO v_categorie, v_produce, v_clasificare;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu acest id!');
    ELSE
        INSERT INTO clasific_clienti
        VALUES (2, v_categorie, v_produce, null);

        UPDATE clasific_clienti
        SET clasificare = v_clasificare
        WHERE id_client = 2;

        COMMIT;
    END IF;

    EXCEPTION
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Doi sau mai multi clienti au acelasi id, id-ul trebuie sa fie
            unic!');
        END;
```

Se poate observa ca eroarea NO\_DATA\_FOUND nu a fost scrisa in blocul EXCEPTION pentru ca operatia de delete nu va arunca aceasta eroare in cazul in care nu gaseste niciun client cu id-ul respectiv, dar putem verifica ca nu a fost gasit si sa tratam acest caz folosind expresia SQL%NOTFOUND.

Pentru a verifica ca sunt tratate cele doua erori:

1. Am introdus in tabel inca un client cu id-ul 105.  
INSERT INTO clasific\_clienti VALUES (105, 2, 30, 's');

```

588 /
589
590 SELECT * FROM clasific_clienti;
591

```

	ID_CLIENT	ID_CATEGORIE	NR_PRODUSE	CLASIFICARE
1	101	5	100	a
2	102	7	40	c
3	102	2	80	s
4	104	3	230	d
5	105	4	20	b
6	105	2	30	s

Apoi la rulara codului pentru id-ul 105 se va afisa mesajul corespunzator.

```

563     v_categorie NUMBER;
564     v_produce   NUMBER;
565     v_clasificare CHAR(1);
566 BEGIN
567     DELETE FROM clasific_clienti WHERE id_client=105
568     RETURNING id_categorie, nr_produce, clasificare
569     INTO v_categorie, v_produce, v_clasificare;
570
571     IF SQL%NOTFOUND THEN
572         DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu acest id!');
573     ELSE
574         INSERT INTO clasific_clienti
575         VALUES (105, v_categorie, v_produce, null);
576
577         UPDATE clasific_clienti
578         SET clasificare = v_clasificare
579         WHERE id_client = 105;
580
581         COMMIT;
582     END IF;
583

```

Doi sau mai multi clienti au acelasi id, id-ul trebuie sa fie unic!

PL/SQL procedure successfully completed.

2. Am rulat codul modificand id\_client cu unul care nu exista.

Daca rulez codul pentru id-ul 2, spre exemplu, care nu exista, se va afisa mesajul corespunzator.

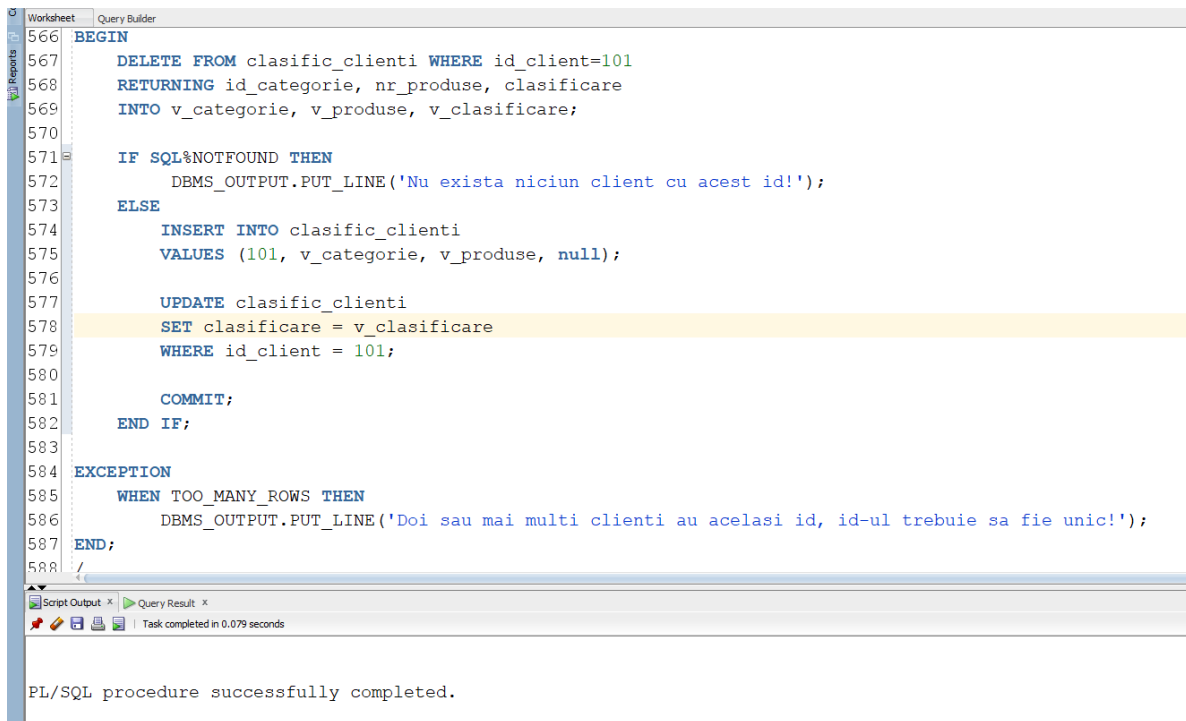
```
563     v_categorie NUMBER;
564     v_produce NUMBER;
565     v_clasificare CHAR(1);
566 BEGIN
567     DELETE FROM clasific_clienti WHERE id_client=2
568     RETURNING id_categorie, nr_produce, clasificare
569     INTO v_categorie, v_produce, v_clasificare;
570
571     IF SQL%NOTFOUND THEN
572         DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu acest id!');
573     ELSE
574         INSERT INTO clasific_clienti
575         VALUES (2, v_categorie, v_produce, null);
576
577         UPDATE clasific_clienti
578         SET clasificare = v_clasificare
579         WHERE id_client = 2;
580
581         COMMIT;
582     END IF;
583
```

Script Output x Query Result x  
Task completed in 0.082 seconds

Nu exista niciun client cu acest id!

PL/SQL procedure successfully completed.

In final, daca vrem sa verificam ca merge codul daca id-ul exista in tabel si este unic putem sa alegem de exemplu id-ul 101.



```
566 BEGIN
567     DELETE FROM clasific_clienti WHERE id_client=101
568     RETURNING id_categorie, nr_produce, clasificare
569     INTO v_categorie, v_produce, v_clasificare;
570
571     IF SQL%NOTFOUND THEN
572         DBMS_OUTPUT.PUT_LINE('Nu exista niciun client cu acest id!');
573     ELSE
574         INSERT INTO clasific_clienti
575         VALUES (101, v_categorie, v_produce, null);
576
577         UPDATE clasific_clienti
578         SET clasificare = v_clasificare
579         WHERE id_client = 101;
580
581         COMMIT;
582     END IF;
583
584 EXCEPTION
585     WHEN TOO_MANY_ROWS THEN
586         DBMS_OUTPUT.PUT_LINE('Doi sau mai multi clienti au acelasi id, id-ul trebuie sa fie unic!');
587 END;
588 /
```

Script Output x Query Result x

Task completed in 0.079 seconds

PL/SQL procedure successfully completed.

### Bibliografie:

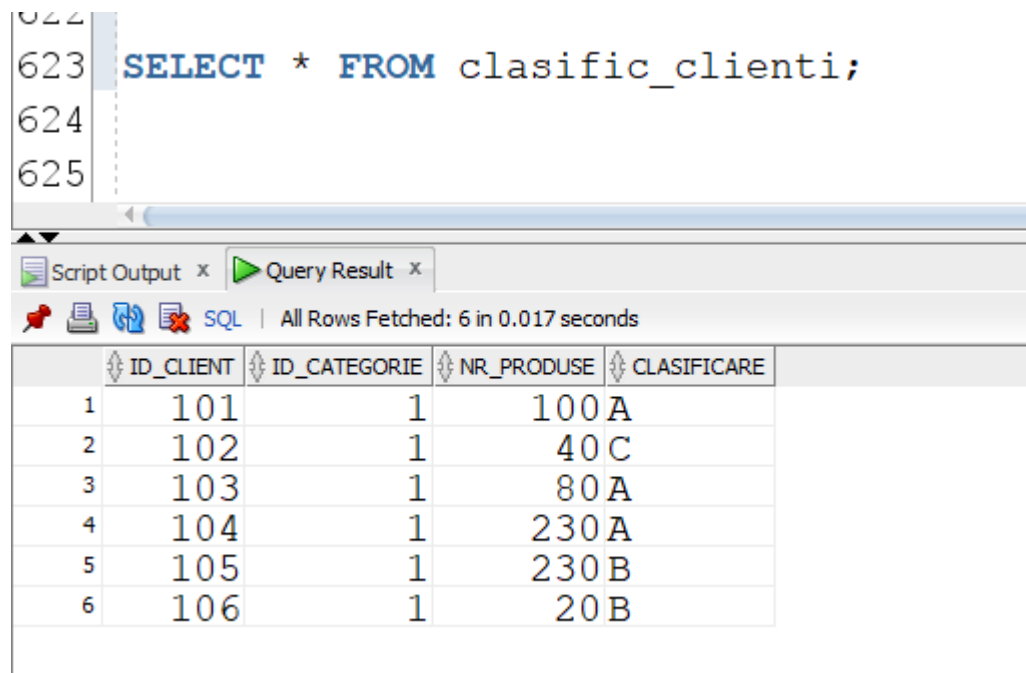
<https://community.oracle.com/tech/developers/discussion/854804/does-a-delete-throw-a-when-no-data-found-exception-when-no-records-arefound>

<https://coderanch.com/t/434012/application-servers/delete-existing-row>

**Tema 4** – La exemplul 3.15 din curs este mai efficient codul scris in PL/SQL sau codul echivalentul in SQL? De cronometrat.

Ne vom folosi de tabela clasific\_clienti creata la punctul anterior doar ca vom sterge toate randurile si inseram urmatoarele:

```
INSERT INTO clasific_clienti VALUES (101, 1, 100, 'A');
INSERT INTO clasific_clienti VALUES (102, 1, 40, 'C');
INSERT INTO clasific_clienti VALUES (103, 1, 80, 'A');
INSERT INTO clasific_clienti VALUES (104, 1, 230, 'A');
INSERT INTO clasific_clienti VALUES (105, 1, 230, 'B');
INSERT INTO clasific_clienti VALUES (106, 1, 20, 'B');
```



The screenshot shows a SQL IDE interface. The top pane displays the query: `SELECT * FROM clasific_clienti;`. The bottom pane, titled 'Query Result', shows the results of the query. It indicates 'All Rows Fetched: 6 in 0.017 seconds'. The results are displayed in a table with the following columns: ID\_CLIENT, ID\_CATEGORIE, NR\_PRODUSE, and CLASIFICARE.

	ID_CLIENT	ID_CATEGORIE	NR_PRODUSE	CLASIFICARE
1	101	1	100	A
2	102	1	40	C
3	103	1	80	A
4	104	1	230	A
5	105	1	230	B
6	106	1	20	B

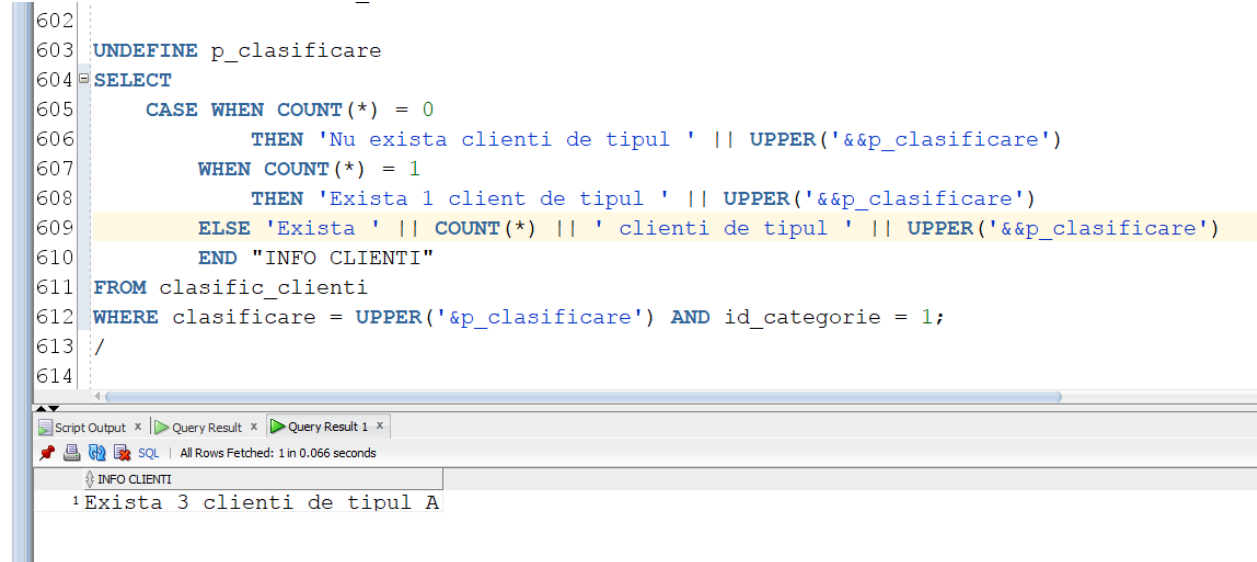
Codul in PL/SQL este:

```
UNDEFINE p_clasificare
SELECT
  CASE WHEN COUNT(*) = 0
    THEN 'Nu exista clienti de tipul ' || UPPER('&p_clasificare')
  WHEN COUNT(*) = 1
    THEN 'Exista 1 client de tipul ' || UPPER('&p_clasificare')
  ELSE 'Exista ' || COUNT(*) || ' clienti de tipul ' || UPPER('&p_clasificare')
  END "INFO CLIENTI"
FROM clasific_clienti
```

WHERE clasificare = UPPER('&p\_clasificare') AND id\_categorie = 1;

La rularea codului pentru p\_clasificare A rezultatul este urmatorul:

```
602
603 UNDEFINE p_clasificare
604 SELECT
605     CASE WHEN COUNT(*) = 0
606         THEN 'Nu exista clienti de tipul ' || UPPER('&p_clasificare')
607         WHEN COUNT(*) = 1
608         THEN 'Exista 1 client de tipul ' || UPPER('&p_clasificare')
609         ELSE 'Exista ' || COUNT(*) || ' clienti de tipul ' || UPPER('&p_clasificare')
610     END "INFO CLIENTI"
611 FROM clasific_clienti
612 WHERE clasificare = UPPER('&p_clasificare') AND id_categorie = 1;
613 /
614
```



Rezultatul a fost returnat in 0.066 secunde, iar inaintea de rulare s-au folosit urmatoarele comenzi pentru golirea cache-ului:

```
alter system flush buffer_cache;
alter system flush shared_pool;
```

Codul echivalent in SQL este:

```
SELECT
    CASE WHEN COUNT(*) = 0
        THEN 'Nu exista clienti de tipul A'
        WHEN COUNT(*) = 1
        THEN 'Exista 1 client de tipul A'
        ELSE 'Exista ' || COUNT(*) || ' clienti de tipul A'
    END "INFO CLIENTI"
FROM clasific_clienti
WHERE clasificare = 'A' AND id_categorie = 1;
```

```

613 /
614
615 SELECT
616     CASE WHEN COUNT(*) = 0
617         THEN 'Nu exista clienti de tipul A'
618     WHEN COUNT(*) = 1
619         THEN 'Exista 1 client de tipul A'
620     ELSE 'Exista ' || COUNT(*) || ' clienti de tipul A'
621     END "INFO CLIENTI"
622 FROM clasific_clienti
623 WHERE clasificare = 'A' AND id_categorie = 1;
624 /

```

Script Output x Query Result x Query Result 1 x	
SQL   All Rows Fetched: 1 in 0.065 seconds	
INFO CLIENTI	
1	Exista 3 clienti de tipul A

Rezultatul este returnat in 0.065 secunde.


Observam ca pentru un set de date foarte mic ca acesta (6 inregistrari) diferenta este foarte mica. Urmeaza testarea cu un set de date de 25 de inregistrari.

```

651
652 SELECT * FROM clasific_clienti;
653

```

Script Output x Query Result x

 All Rows Fetched: 25 in 0.014 seconds

	ID_CLIENT	ID_CATEGORIE	NR_PRODUSE	CLASIFICARE
1	101	1	100	A
2	102	1	40	C
3	103	1	80	A
4	104	1	230	A
5	105	1	230	B
6	106	1	20	B
7	107	1	100	A
8	108	1	40	C
9	109	1	80	A
10	110	1	230	A
11	111	1	230	B
12	112	1	20	B
13	113	1	100	A
14	114	1	40	C
15	115	1	80	A
16	116	1	230	A
17	117	1	230	B
18	118	1	20	B
19	119	1	100	A
20	120	1	40	C
21	121	1	80	A
22	122	1	230	A
23	123	1	230	B
24	124	1	20	B
25	124	1	20	B



```

603 UNDEFINE p_clasificare
604 SELECT
605     CASE WHEN COUNT(*) = 0
606         THEN 'Nu exista clienti de tipul ' || UPPER('&p_clasificare')
607         WHEN COUNT(*) = 1
608         THEN 'Exista 1 client de tipul ' || UPPER('&p_clasificare')
609         ELSE 'Exista ' || COUNT(*) || ' clienti de tipul ' || UPPER('&p_clasificare')
610         END "INFO CLIENTI"
611 FROM clasific_clienti
612 WHERE clasificare = UPPER('&p_clasificare') AND id_categorie = 1;
613 /

```

Script Output x	Query Result x	Query Result 1 x
SQL   All Rows Fetched: 1 in 0.066 seconds		
<div>INFO CLIENTI</div> <div>1 Exista 12 clienti de tipul A</div>		

Varianta in PL/SQL dureaza tot 0.066 secunde.

```

5 SELECT
6     CASE WHEN COUNT(*) = 0
7         THEN 'Nu exista clienti de tipul A'
8         WHEN COUNT(*) = 1
9         THEN 'Exista 1 client de tipul A'
0         ELSE 'Exista ' || COUNT(*) || ' clienti de tipul A'
1         END "INFO CLIENTI"
2 FROM clasific_clienti
3 WHERE clasificare = 'A' AND id_categorie = 1;
4 /

```

Script Output x	Query Result x	Query Result 1 x
SQL   All Rows Fetched: 1 in 0.062 seconds		
<div>INFO CLIENTI</div> <div>1 Exista 12 clienti de tipul A</div>		

Varianta in SQL dureaza 0.062 secunde. Observam ca diferenta de timp creste odata cu marimea setului de date, varianta in SQL fiind mai rapida.

Bibliografie:

<https://stackoverflow.com/questions/2147456/how-to-clear-all-cached-items-in-oracle>

**Tema 5** – De gasit caractere care sunt stocate pe mai mult de un byte si sa vedem ce erori pot aprea.

Lista tuturor caracterelor stocate pe mai mult de un byte se poate gasi pe site-ul acesta:

<https://design215.com/toolbox/ascii-utf8.php>.

Erorile cauzate de aceste caractere stocate pe mai mult de un byte pot aparea, spre exemplu, cand specificam numarul de caractere al unei coloane si incercam sa stocam un sir de caractere cu lungimea respectiva (dar el ocupa mai multi bytes decat are lungimea).

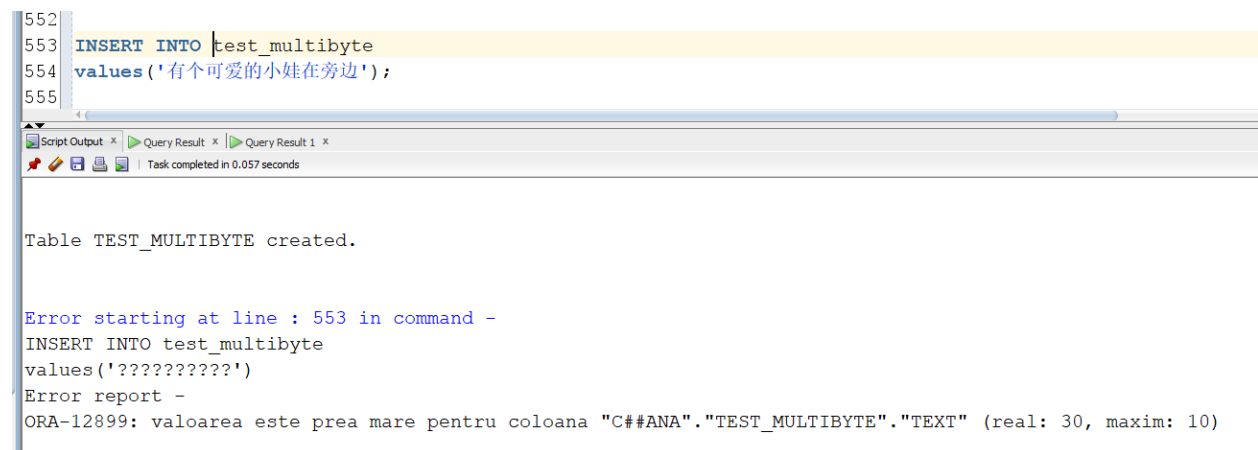
Pentru exemplificare am creat un tabel cu definitia urmatoare:

```
CREATE TABLE test_multibyte(  
    text varchar2(10)  
);
```

Apoi am incercat inserarea unui text de lungime 10, dar care contine caractere stocate pe mai multi bytes.

```
INSERT INTO test_multibyte  
values('有个可爱的小娃在旁边');
```

Se va produce urmatoarea eroare:



```
552  
553 INSERT INTO test_multibyte  
554 values('有个可爱的小娃在旁边');  
555  
Table TEST_MULTIBYTE created.  
  
Error starting at line : 553 in command -  
INSERT INTO test_multibyte  
values('????????')  
Error report -  
ORA-12899: valoarea este prea mare pentru coloana "C##ANA"."TEST_MULTIBYTE"."TEXT" (real: 30, maxim: 10)
```

Acesta eroare se produce pentru ca definitia VARCHAR2(10) este echivalenta cu VARCHAR2(10 BYTE) si nu cu VARCHAR2(10 CHAR) care ar fi putut sa stocheze sirul de mai sus.

**Bibliografie:**

<https://community.oracle.com/tech/developers/discussion/2569732/multibyte-character>

<https://stackoverflow.com/questions/13505367/multibyte-characters-in-oracle>

<https://design215.com/toolbox/ascii-utf8.php>

## Tema 6 – Cum facem daca dorim sa retinem datele din baza de date cu diacritice? (alter session set zona, tara, limba etc)

Pentru inceput, am definit un tabel in care voi retine texte cu diacritice.

```
CREATE TABLE test_diacritice(  
    id NUMBER PRIMARY KEY,  
    text_cu_diacritice VARCHAR2(2000)  
);
```

```
INSERT INTO test_diacritice VALUES (1, 'Cheia succesului e să începi înainte de a fi pregătit.');
```

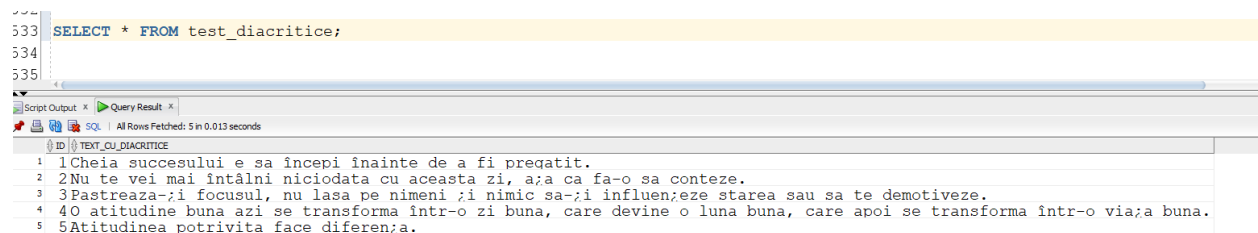
```
INSERT INTO test_diacritice VALUES (2, 'Nu te vei mai întâlni niciodată cu această zi, așa că fă-o  
să conteze.');
```

```
INSERT INTO test_diacritice VALUES (3, 'Păstrează-ți focusul, nu lăsa pe nimeni și nimic să-ți  
influențeze starea sau să te demotiveze.');
```

```
INSERT INTO test_diacritice VALUES (4, 'O atitudine bună azi se transformă într-o zi bună, care  
devine o lună bună, care apoi se transformă într-o viață bună.');
```

```
INSERT INTO test_diacritice VALUES (5, 'Atitudinea potrivită face diferența.');
```

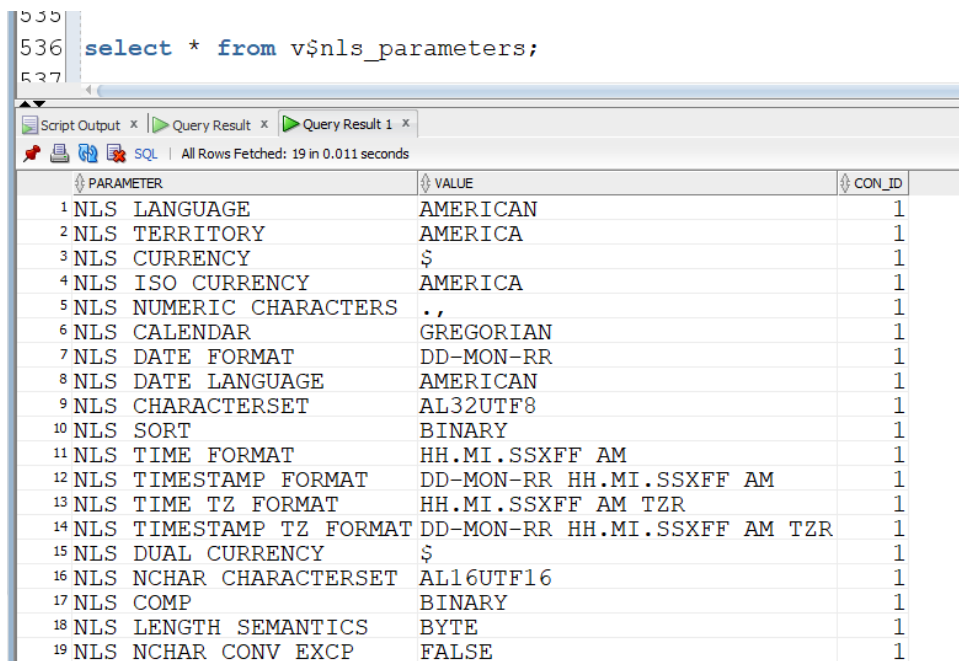
Daca facem un select putem observa ca unele diacritice sunt inlocuite cu semul intrebarii intors, iar unele sunt ignorate cum e cazul lui ă inlocui cu a.



```
533 SELECT * FROM test_diacritice;
```

ID	TEXT_CU_DIACRITICE
1	1Cheia succesului e sa începi înainte de a fi pregatit.
2	2Nu te vei mai întâlni niciodata cu aceasta zi, a;a ca fa-o sa conteze.
3	3Pastreaza-ți focusul, nu lasa pe nimeni și nimic sa-ți influen;eze starea sau sa te demotiveze.
4	4O atitudine buna azi se transforma într-o zi buna, care devine o luna buna, care apoi se transforma într-o via;a buna.
5	5Atitudinea potrivita face diferen;a.

Daca rulam comanda select \* from v\$nls\_parameters putem vedea parametrii sesiunii:



```
535  
536 select * from v$nls_parameters;
```

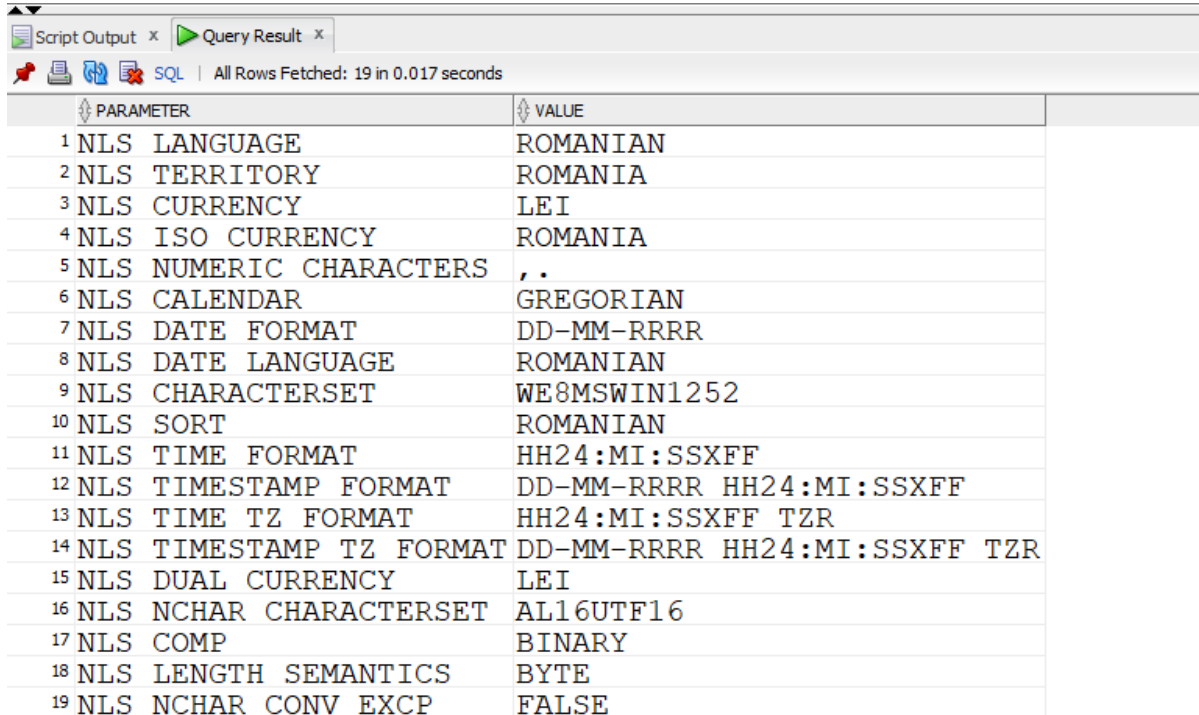
PARAMETER	VALUE	CON_ID
1 NLS_LANGUAGE	AMERICAN	1
2 NLS_TERRITORY	AMERICA	1
3 NLS_CURRENCY	\$	1
4 NLS_ISO_CURRENCY	AMERICA	1
5 NLS_NUMERIC_CHARACTERS	.,	1
6 NLS_CALENDAR	GREGORIAN	1
7 NLS_DATE_FORMAT	DD-MON-RR	1
8 NLS_DATE_LANGUAGE	AMERICAN	1
9 NLS_CHARACTERSET	AL32UTF8	1
10 NLS_SORT	BINARY	1
11 NLS_TIME_FORMAT	HH.MI.SSXFF AM	1
12 NLS_TIMESTAMP_FORMAT	DD-MON-RR HH.MI.SSXFF AM	1
13 NLS_TIME_TZ_FORMAT	HH.MI.SSXFF AM TZR	1
14 NLS_TIMESTAMP_TZ_FORMAT	DD-MON-RR HH.MI.SSXFF AM TZR	1
15 NLS_DUAL_CURRENCY	\$	1
16 NLS_NCHAR_CHARACTERSET	AL16UTF16	1
17 NLS_COMP	BINARY	1
18 NLS_LENGTH_SEMANTICS	BYTE	1
19 NLS_NCHAR_CONV_EXCP	FALSE	1

Vom schimba limba sesiunii in romana si zona in Romania cu comenziile:

```
ALTER SESSION SET NLS_LANGUAGE = 'ROMANIAN';
```

```
ALTER SESSION SET NLS_TERRITORY = 'ROMANIA';
```

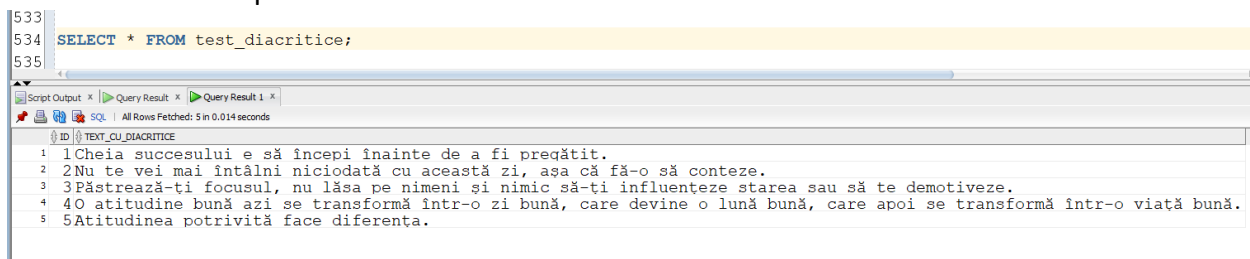
Putem observa ca s-au schimbat acesti parametrii:



The screenshot shows the SQL Developer interface with the 'Query Result' tab active. It displays a table of NLS parameters. The table has two columns: 'PARAMETER' and 'VALUE'. There are 19 rows of data, numbered 1 through 19. The parameters include NLS\_LANGUAGE, NLS\_TERRITORY, NLS\_CURRENCY, NLS\_ISO\_CURRENCY, NLS\_NUMERIC\_CHARACTERS, NLS\_CALENDAR, NLS\_DATE\_FORMAT, NLS\_DATE\_LANGUAGE, NLS\_CHARACTERSET, NLS\_SORT, NLS\_TIME\_FORMAT, NLS\_TIMESTAMP\_FORMAT, NLS\_TIME\_TZ\_FORMAT, NLS\_TIMESTAMP\_TZ\_FORMAT, NLS\_DUAL\_CURRENCY, NLS\_NCHAR\_CHARACTERSET, NLS\_COMP, NLS\_LENGTH\_SEMANTICS, and NLS\_NCHAR\_CONV\_EXCP.

PARAMETER	VALUE
1 NLS_LANGUAGE	ROMANIAN
2 NLS_TERRITORY	ROMANIA
3 NLS_CURRENCY	LEI
4 NLS_ISO_CURRENCY	ROMANIA
5 NLS_NUMERIC_CHARACTERS	, .
6 NLS_CALENDAR	GREGORIAN
7 NLS_DATE_FORMAT	DD-MM-RRRR
8 NLS_DATE_LANGUAGE	ROMANIAN
9 NLS_CHARACTERSET	WE8MSWIN1252
10 NLS_SORT	ROMANIAN
11 NLS_TIME_FORMAT	HH24:MI:SSXFF
12 NLS_TIMESTAMP_FORMAT	DD-MM-RRRR HH24:MI:SSXFF
13 NLS_TIME_TZ_FORMAT	HH24:MI:SSXFF TZR
14 NLS_TIMESTAMP_TZ_FORMAT	DD-MM-RRRR HH24:MI:SSXFF TZR
15 NLS_DUAL_CURRENCY	LEI
16 NLS_NCHAR_CHARACTERSET	AL16UTF16
17 NLS_COMP	BINARY
18 NLS_LENGTH_SEMANTICS	BYTE
19 NLS_NCHAR_CONV_EXCP	FALSE

Iar in baza de date putem acum stoca texte cu diacritice:



The screenshot shows the SQL Developer interface with the 'Query Result' tab active. It displays the result of a query: 'SELECT \* FROM test\_diacritice;'. The result is a table with one column, 'TEXT\_CU\_DIACRITICE', and five rows of text. The text contains various diacritics and special characters.

TEXT_CU_DIACRITICE
1 Cheia succesului e să începi înainte de a fi pregătit.
2 Nu te vei mai întâlni niciodată cu această zi, așa că fă-o să conteze.
3 Păstrează-ți focusul, nu lăsa pe nimeni și nimic să-ți influențeze starea sau să te demotiveze.
4 40 atitudine bună azi se transformă într-o zi bună, care devine o lună bună, care apoi se transformă într-o viață bună.
5 Atitudinea potrivită face diferența.

Bibliografie:

<https://docs.oracle.com/en/database/oracle/oracle-database/19/ntcli/configuring-locale-and-character-sets-using-nls-lang-environment-variable.html>

<https://docs.oracle.com/en/database/oracle/oracle-database/19/nlspg/setting-up-globalization-support-environment.html#GUID-6475CA50-6476-4559-AD87-35D431276B20>

<https://www.oracle.com/ro/database/technologies/faq-nls-lang.html>

<https://community.oracle.com/tech/developers/discussion/2376286/searching-diacritic-characters>

<https://community.oracle.com/tech/developers/discussion/3600439/displaying-diacritics-can-it-be-done-in-sqlplus>

<https://ora-base.com/2020/01/17/changing-the-oracle-database-character-set/>

**Tema 7** - Cand folosesc obiectele de tip lob (blob, clob, nclob, bfile)? Cand stochez direct in baza de date si cand folosesc un link? Cum le stochez daca vreau securitate maxima?

Obiectele **LOB (Large object)** dupa cum le spune si numele sunt un set de tipuri de date concepute pentru a stoca cantitati mari de date. Un LOB poate contine pana la o dimensiune maxima intre 8 terabytes la 128 terabytes, in functie de modul in care este configurata baza de date.

Tipurile de date de tip **LOB** (Large object) se impart in subcategoriile:

- Character Large Object - **CLOB**,
- Binary large object - **BLOB**,
- Binary file - **BFILE**,
- National language character large object -**NCLOB**

Aceste obiecte sunt potrivite pentru stocarea datelor nestructurate (precum o imagine stocata ca fisier binar) sau semi-structurate (precum un document XML). CLOB si NCLOB sunt ideale pentru stocarea datelor semi-structurate, iar BFILE si BLOB pentru datele nestructurate.

Tipurile de date LOB au mai multe avantaje față de tipurile LONG și LONG RAW, inclusiv:

- **Capacitatea** - LOB-urile pot stoca 4 GB de date sau mai mult, în funcție de configurația sistemului. Tipurile LONG și LONG RAW sunt limitate la 2 GB de date
- **Numărul de coloane LOB dintr-un tabel** - Un tabel poate avea mai multe coloane LOB. Coloanele LOB dintr-un tabel pot fi de orice tip LOB. În Oracle Database Release 7.3 și versiunile ulterioare, tabelele sunt limitate la o singură coloană LONG sau LONG RAW.
- **Acces aleatoriu pe bucăți**: LOB-urile acceptă acces aleatoriu la date, dar LONG-urile acceptă doar acces secvențial.

LOB-urile sunt impartite in doua tipuri: interne si externe.

LOB-urile interne sunt stocate în spațiul de tabele al bazei de date într-un mod care optimizează spațiul și permite accesul eficient. LOB-urile interne se pot recupera în cazul unei tranzacții esuate, iar orice modificare a valorii LOB internă poate fi confirmată sau anulată.

LOB-urile externe (BFILES) sunt obiecte mari de date binare stocate în fișierele sistemului de operare în afara spațiului de tabele al bazei de date. Aceste fișiere folosesc link-uri catre locatia reala in care sunt stocate. În afară de dispozitivele de stocare secundare convenționale, cum ar fi hard disk-urile, BFILE-urile pot fi, de asemenea, amplasate si pe alte dispozitive de stocare în bloc, cum ar fi CD-ROM-uri, PhotoCD-uri și DVD-uri.

In general, se prefera stocarea direct in baza de date a LOB-urilor deoarece aduce un plus de performanta, fiind optimizate pentru acces eficient. De asemenea, LOB-urile interne au avantajul de a se putea recupera in cazul unei tranzactii esuate. Atunci cand avem nevoie sa stocam date in regim read-only pe medii externe (spre exemplu CD-uri) putem folosi LOB-uri externe.

Cand vrem securitate maxima putem folosi LOB-urile SecureFiles pentru care se pot aplica functii precum criptarea, compresia si deduplicarea. Sistemul de fisiere al bazei de date (DBFS) oferă o interfață pentru fisierele care sunt stocate într-o bază de date Oracle. Cu DBFS se pot face referințe de la locatoarele SecureFile-urilor la fisiere stocate în afara bazei de date.

Bibliografie:

[https://docs.oracle.com/cd/B19306\\_01/appdev.102/b14249/adlob\\_intro.htm](https://docs.oracle.com/cd/B19306_01/appdev.102/b14249/adlob_intro.htm)

[http://ora-srv.wlv.ac.uk/oracle19c\\_doc/adlob/introduction-to-large-objects.html#GUID-DF70A5C9-3FCB-4789-AFB9-FD0405806D12](http://ora-srv.wlv.ac.uk/oracle19c_doc/adlob/introduction-to-large-objects.html#GUID-DF70A5C9-3FCB-4789-AFB9-FD0405806D12)

[https://docs.oracle.com/cd/E11882\\_01/appdev.112/e18294/adlob\\_tables.htm#ADLOB45260](https://docs.oracle.com/cd/E11882_01/appdev.112/e18294/adlob_tables.htm#ADLOB45260)

[https://profs.info.uaic.ro/~bd/wiki/index.php/PLSQL\\_1](https://profs.info.uaic.ro/~bd/wiki/index.php/PLSQL_1)