

CASE

-Expresie vs. Instrucțiune-

Expresia CASE

Exemplul 3.16

```
UNDEFINE p_clasificare
DECLARE
    v_nr NATURAL;
    v_clasificare CHAR(1) := UPPER('&p_clasificare');
    mesaj VARCHAR2(100);
BEGIN
    SELECT COUNT(*) INTO v_nr
    FROM   clasific_clienti
    WHERE  clasificare = v_clasificare
    AND    id_categorie = 1;

    mesaj := CASE
        WHEN v_nr = 0 THEN
            'Nu exista clienti de tipul ' ||
            v_clasificare
        WHEN v_nr = 1 THEN
            'Exista 1 client de tipul ' ||
            v_clasificare
        ELSE
            'Exista ' || v_nr || ' clienti de tipul ' ||
            v_clasificare
        END;

    DBMS_OUTPUT.PUT_LINE(mesaj);
END;
```

În exemplul de mai sus, este utilizată expresia CASE pentru a-i atribui variabilei “mesaj” o valoare. Astfel, în expresia CASE din exemplu, se verifică, pe rând, fiecare condiție și se oprește în momentul în care condiția întâlnită este adevărată. Expresia CASE returnează o valoare. Conform documentației ORACLE, expresia CASE poate fi comparată cu IF THEN ELSE din PL/SQL sau din alte limbaje de programare.

Instrucțiunea CASE

Exemplul 3.14

```
DECLARE
  v_nr NATURAL;
  v_clas CHAR(1) := UPPER('&p_clasificare');
BEGIN
  SELECT COUNT(*) INTO v_nr
  FROM   clasific_clienti
  WHERE  clasificare = v_clas
  AND    id_categorie = 1;

  CASE
    WHEN v_nr = 0 THEN
      DBMS_OUTPUT.PUT_LINE('Nu exista clienti de ' ||
                           'tipul ' || v_clas);
    WHEN v_nr =1 THEN
      DBMS_OUTPUT.PUT_LINE('Exista 1 client ' ||
                           'de tipul ' || v_clas);
    ELSE
      DBMS_OUTPUT.PUT_LINE('Exista ' ||v_nr ||
                           ' clienti de tipul ' || v_clas);
  END CASE;
END;
```

În exemplul 3.14, este utilizată instrucțiunea CASE pentru a afișa pe ecran. Practic, instrucțiunea CASE execută acțiunea de afișare. Conform documentației ORACLE, instrucțiunea CASE poate fi comparată cu SWITCH-ul din alte limbaje de programare.

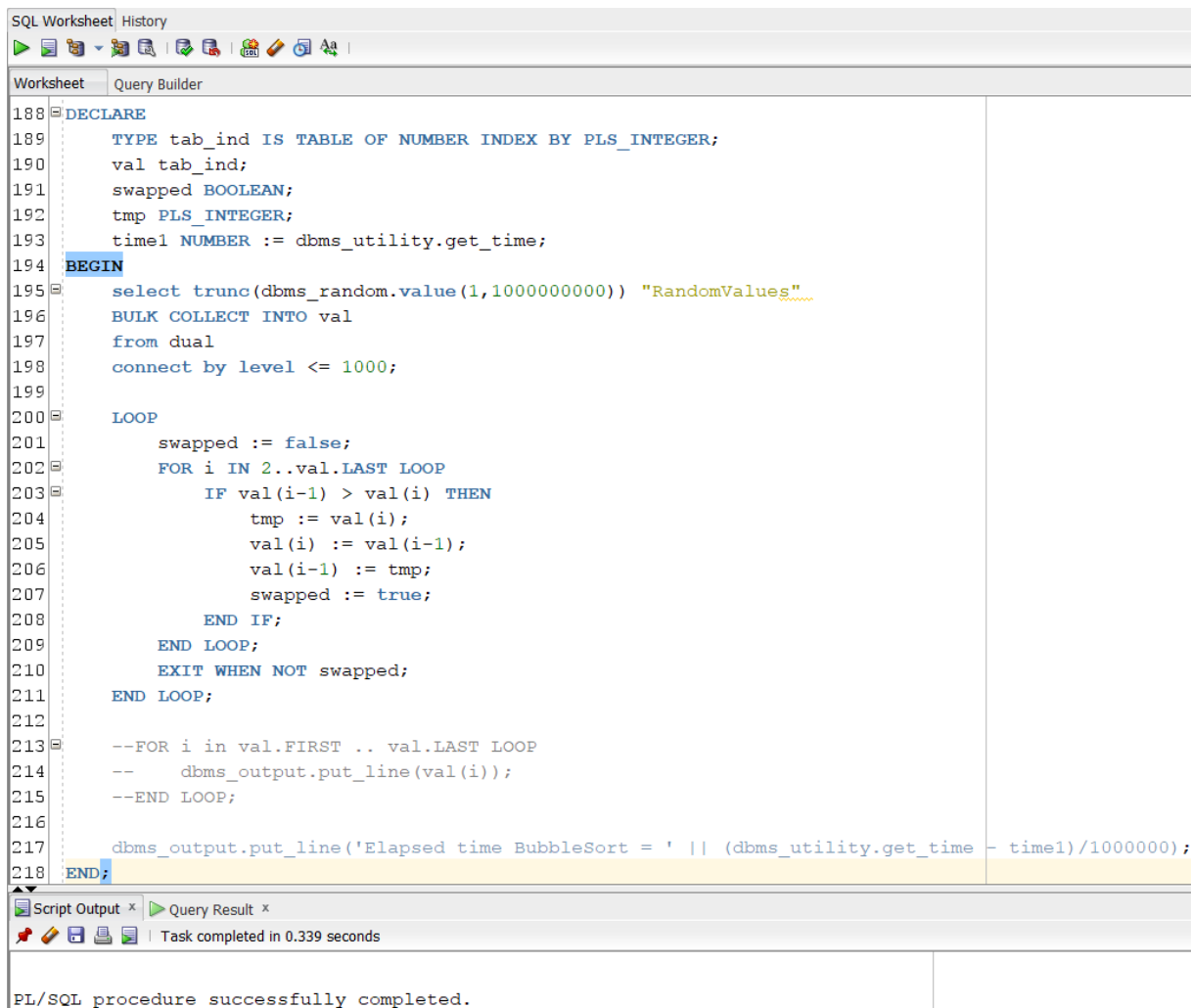
Așadar, conform comparației din documentația ORACLE, instrucțiunea CASE este mai eficientă decât expresia CASE, IF THEN ELSE fiind mai încet decât SWITCH. De asemenea, diferența dintre cele două nu este neapărat una destul de vizibilă pentru oricine (pe lângă faptul că expresia se termină doar cu END, iar instrucțiunea cu END CASE), deosebirea fiind faptul că expresia CASE returnează o valoare, iar instrucțiunea CASE execută acțiuni.

Sortarea vectorilor

--Algorithm vs Order By--

Algorithm \Rightarrow Bubble Sort

Utilizăm PL/SQL pentru a implementa algoritmul de sortare Bubble Sort. Folosim pachetul DBMS_RANDOM pentru a genera 1 000 de numere random în intervalul 1-1 000 000 000 și pachetul DBMS_UTILITY pentru a genera timpul cu ajutorul funcției GET_TIME.



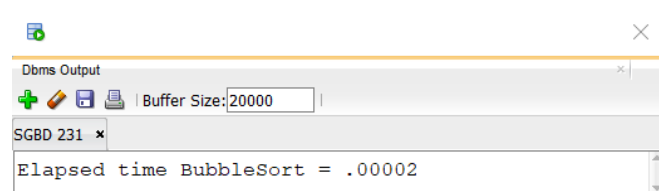
```
188 DECLARE
189     TYPE tab_ind IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
190     val tab_ind;
191     swapped BOOLEAN;
192     tmp PLS_INTEGER;
193     time1 NUMBER := dbms_utility.get_time;
194 BEGIN
195     select trunc(dbms_random.value(1,1000000000)) "RandomValues"
196     BULK COLLECT INTO val
197     from dual
198     connect by level <= 1000;
199
200     LOOP
201         swapped := false;
202         FOR i IN 2..val.LAST LOOP
203             IF val(i-1) > val(i) THEN
204                 tmp := val(i);
205                 val(i) := val(i-1);
206                 val(i-1) := tmp;
207                 swapped := true;
208             END IF;
209         END LOOP;
210         EXIT WHEN NOT swapped;
211     END LOOP;
212
213     --FOR i in val.FIRST .. val.LAST LOOP
214     --    dbms_output.put_line(val(i));
215     --END LOOP;
216
217     dbms_output.put_line('Elapsed time BubbleSort = ' || (dbms_utility.get_time - time1)/1000000);
218 END;
```

Script Output x Query Result x

Task completed in 0.339 seconds

PL/SQL procedure successfully completed.

Rezultat DBMS OUTPUT:



```
Dbms Output
+ + + Buffer Size: 20000
SGBD 231 x
Elapsed time BubbleSort = .00002
```

Order By

Utilizăm PL/SQL pentru a implementa putea evidenția diferența dintre cele două sortări. Folosim pachetul DBMS_RANDOM pentru a genera 1 000 de numere random în intervalul 1-1 000 000 000 și pachetul DBMS_UTILITY pentru a genera timpul cu ajutorul funcției GET_TIME.

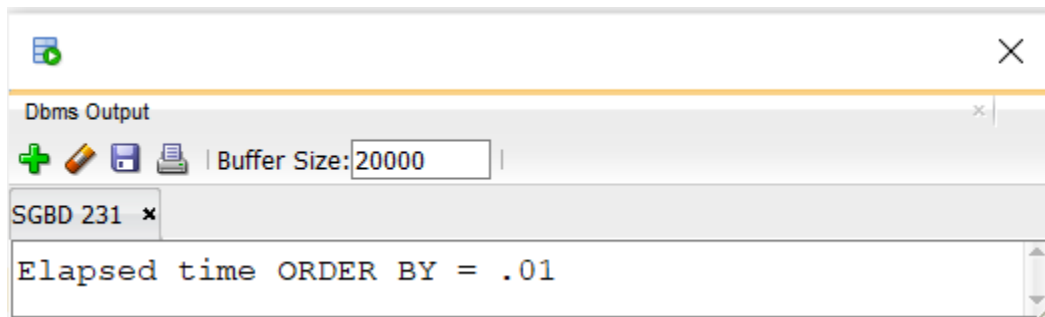
```
221 --Order By
222 DECLARE
223     TYPE tab_ind IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
224     val tab_ind;
225     swapped BOOLEAN;
226     tmp NUMBER(3);
227     time1 NUMBER := dbms_utility.get_time;
228 BEGIN
229     select trunc(dbms_random.value(1,1000000000)) "RandomValues"
230     BULK COLLECT INTO val
231     from dual
232     connect by level <= 1000
233     order by 1;
234
235     --FOR i in val.FIRST .. val.LAST LOOP
236     --     dbms_output.put_line(val(i));
237     --END LOOP;
238
239     dbms_output.put_line('Elapsed time ORDER BY = ' || (dbms_utility.get_time - time1)/100);
240 END;
241 /
```

Script Output x Query Result x

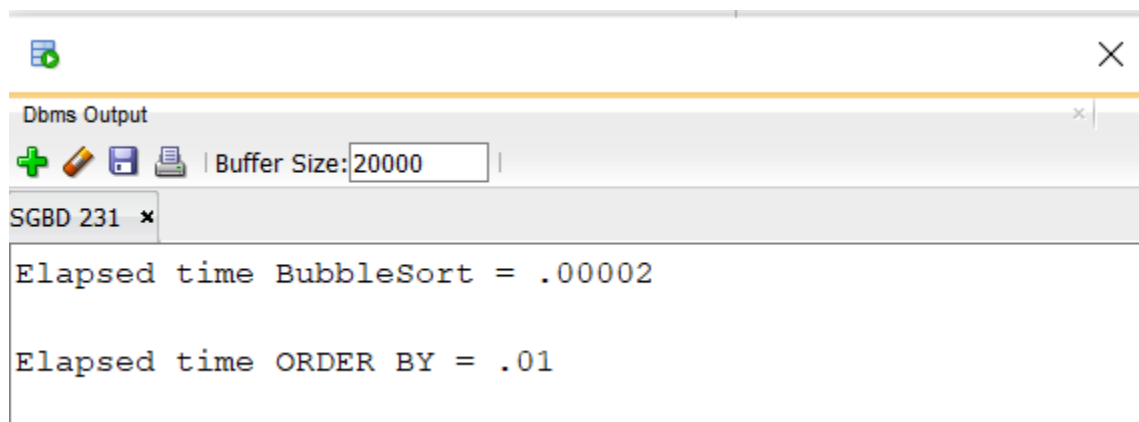
Task completed in 0.339 seconds

PL/SQL procedure successfully completed.

Rezultat DBMS OUTPUT:



Astfel, putem vedea *rezultatele ambelor sortări*:



The screenshot shows a 'Dbms Output' window with a 'Buffer Size' of 20000. It displays the results of two queries: 'Elapsed time BubbleSort = .00002' and 'Elapsed time ORDER BY = .01'. The window has a title bar with a close button and a tab labeled 'SGBD 231'.

```
Elapsed time BubbleSort = .00002

Elapsed time ORDER BY = .01
```

Observăm că timpul de execuție al Bubble Sort-ului este mai mic decât cel al Order By-ului ($0.00002 < 0.01$).

În concluzie, Bubble Sort-ul este mai rapid decât Order By, ceea ce înseamnă că Order By-ul este foarte lent, având în vedere faptul că Bubble Sort este un algoritm de sortare destul de ineficient în comparație cu alții precum QuickSort, MergeSort, ș.a.m.d.