TEMĂ

În această temă voi prezenta *WHERE CURRENT OF* (cursor explicit) vs *FORALL* (cursor implicit definit de Oracle) pe exemplul 5.12 din Curs, exemplul 5.12 (*WHERE CURRENT OF*) vs 5.13 (*WHERE ROWID = i.ROWID*), cursoare *imbricate* 5.16 vs cursoare *parametrizate* 5.10. De asemenea voi prezenta și costul pentru fiecare dintre cele 3 instrucțiuni SQL din exemplul 5.14

***Exemplul 5.12 (WHERE CURRENT OF vs FORALL cu BULK COLLECT INTO) din Curs.***

Pentru a vedea diferențele între cele două vom folosi funcția ***systimestamp*** cu care vom vedea timpul la care a început/sfârșit fiecare instrucțiune, *WHERE CURRENT OF NUME\_CURSOR*¸ respective *FORALL*.

***Rezolvare:***

DECLARE

CURSOR c IS

SELECT employee\_id

FROM employees

WHERE department\_id IN

(SELECT department\_id

FROM departments

WHERE department\_name = 'IT')

FOR UPDATE OF salary NOWAIT;

TYPE tab\_ind IS TABLE OF employees%ROWTYPE INDEX BY PLS\_INTEGER;

t tab\_ind;

BEGIN

dbms\_output.put\_line('Inainte de WHERE CURRENT OF: ' || systimestamp);

FOR i IN c LOOP

UPDATE employees

SET salary = (salary\*0.5)\*2 + 199\*0.95

WHERE CURRENT OF c;

END LOOP;

dbms\_output.put\_line('Dupa de WHERE CURRENT OF: ' || systimestamp);

ROLLBACK;

SELECT \* BULK COLLECT INTO t

FROM employees

WHERE department\_id IN

(SELECT department\_id

FROM departments

WHERE department\_name = 'IT');

dbms\_output.put\_line('Inainte de FORALL: ' || systimestamp);

FORALL i in 1..t.LAST

UPDATE employees

SET salary = (salary\*0.5)\*2 + 199\*0.95

WHERE t(i).employee\_id = employee\_id;

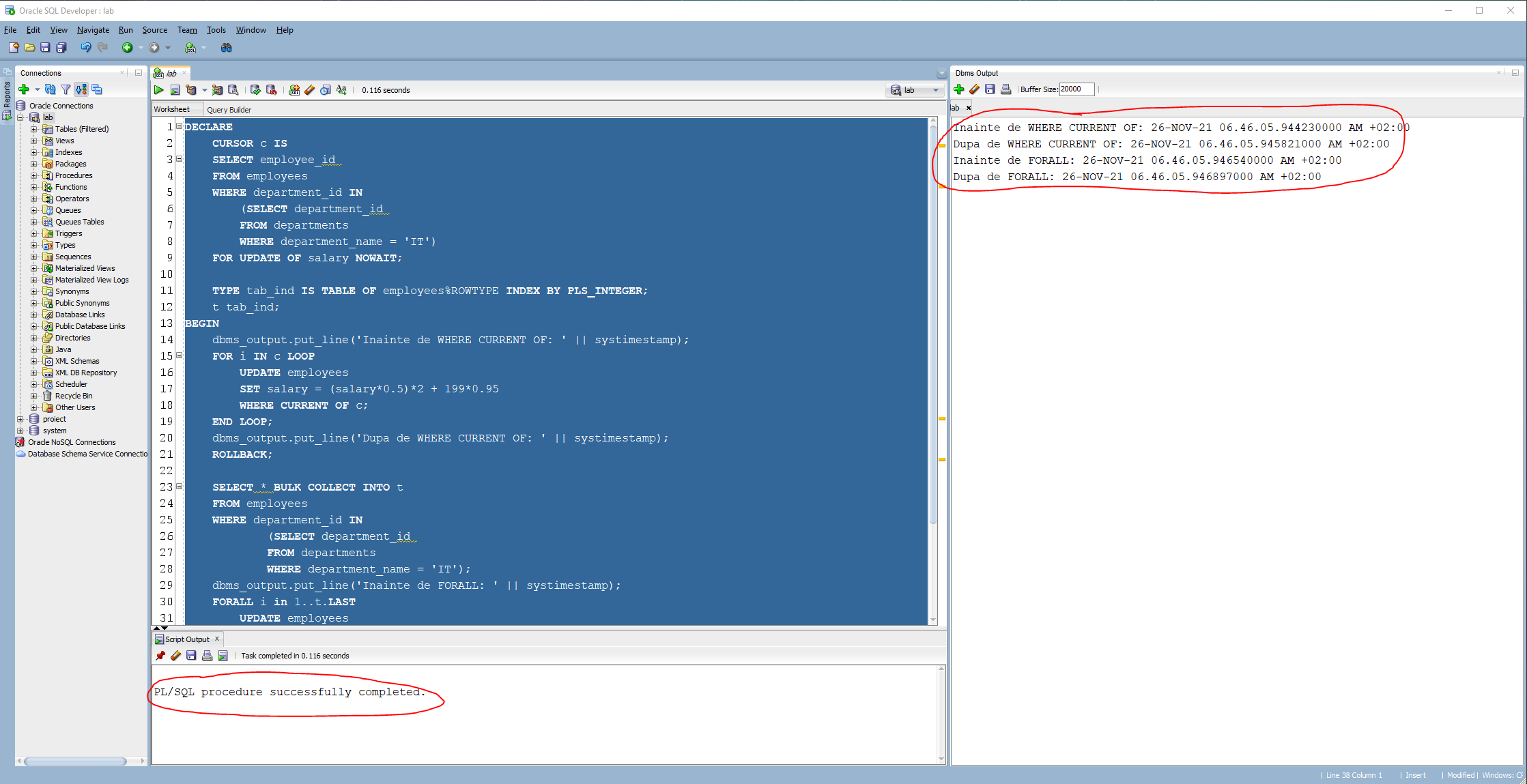
dbms\_output.put\_line('Dupa de FORALL: ' || systimestamp);

COMMIT;

END;

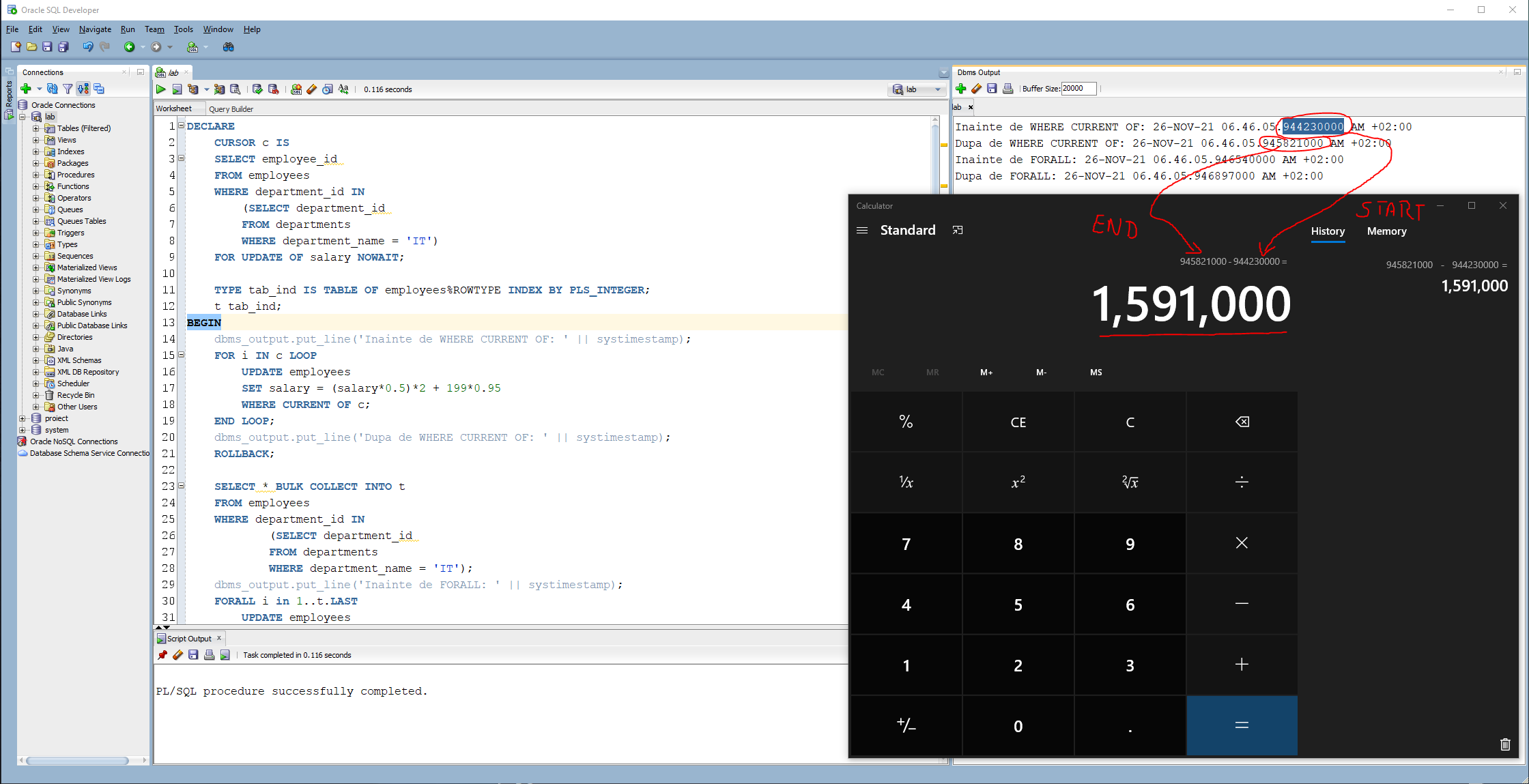
/

***Print-Screen:***

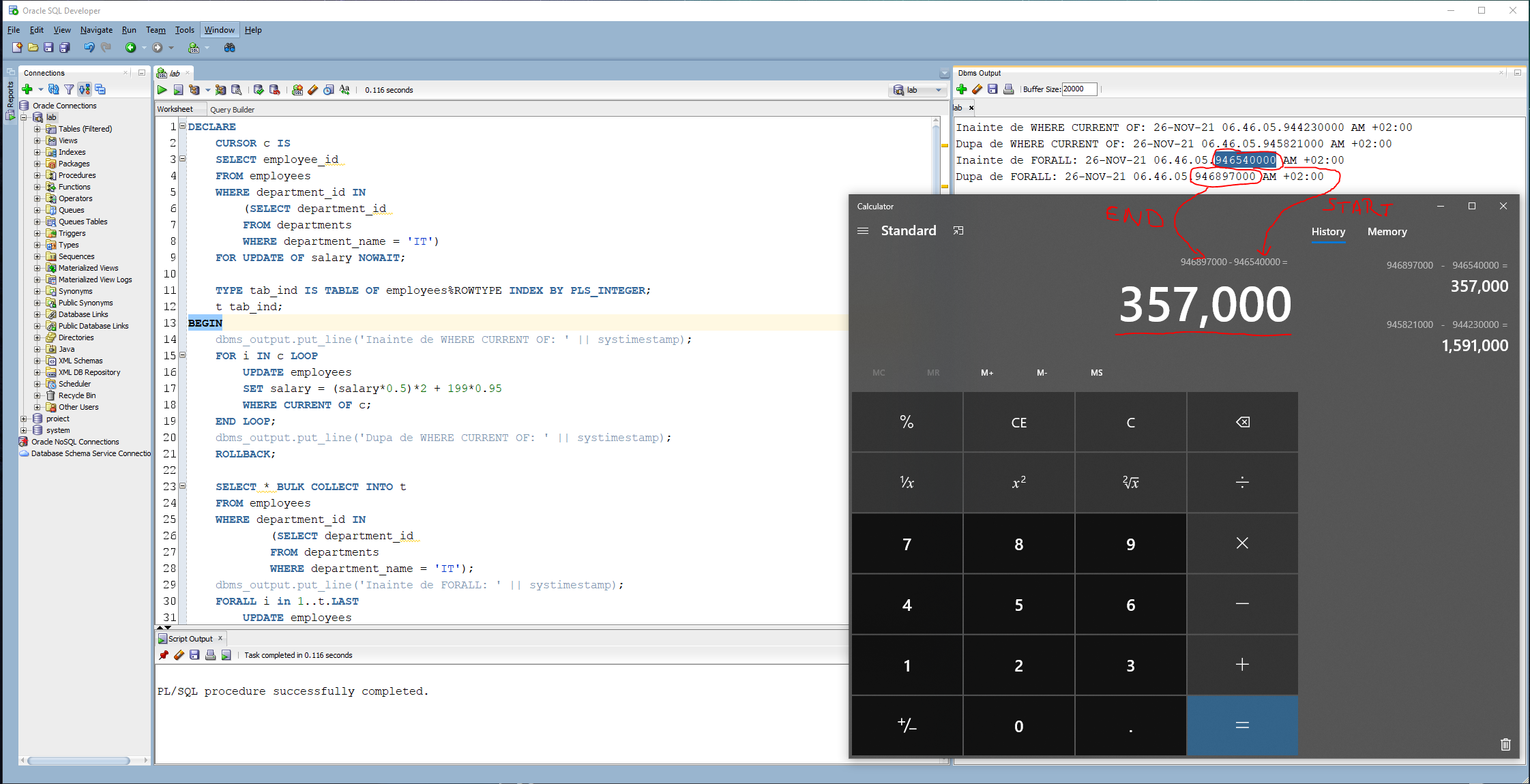


Putem observa că blocul nostru PL/SQL s-a executat cu succes. În dreapta vedem pe prima linie, respectiv a doua, ora la care a început/sfârșit monitorizarea pentru loop-ul cu *WHERE CURRENT OF NUME\_CURSOR*, respectiv pentru *FORALL*.

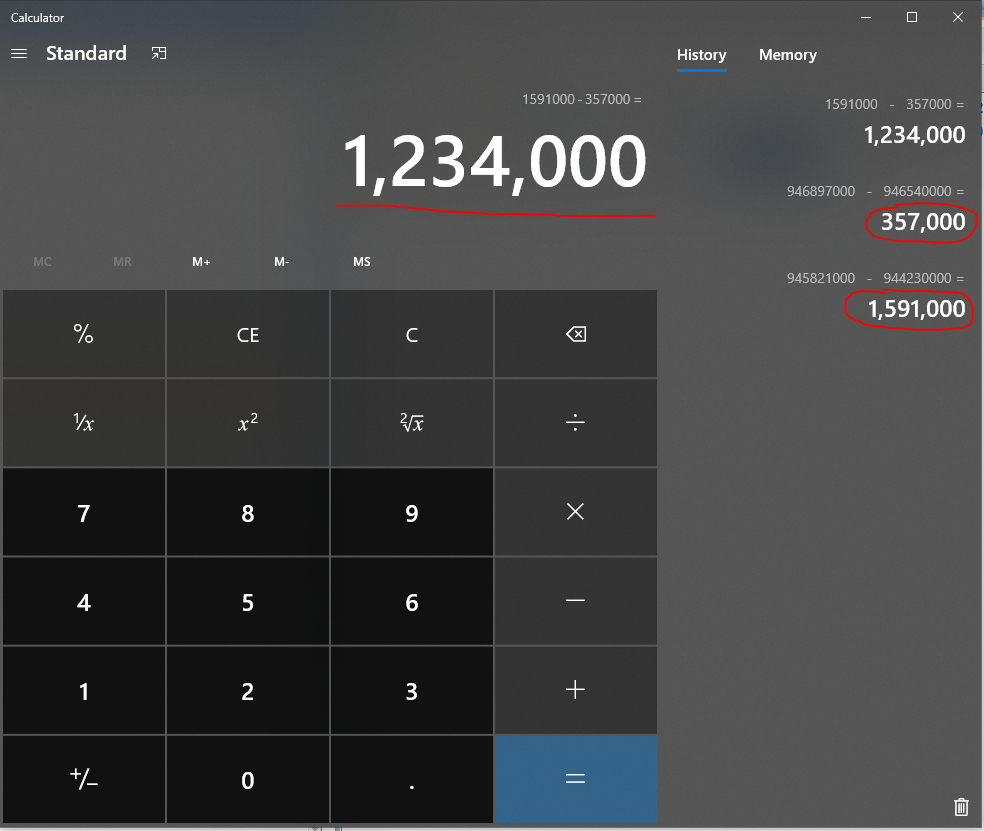
Diferențele sunt infime! Ele se pot observa la ordinul milisecundelor. Pentru a vedea cu exactitate vom face un diferența între timpul de început și timpul de sfâșit folosind calculatorul.



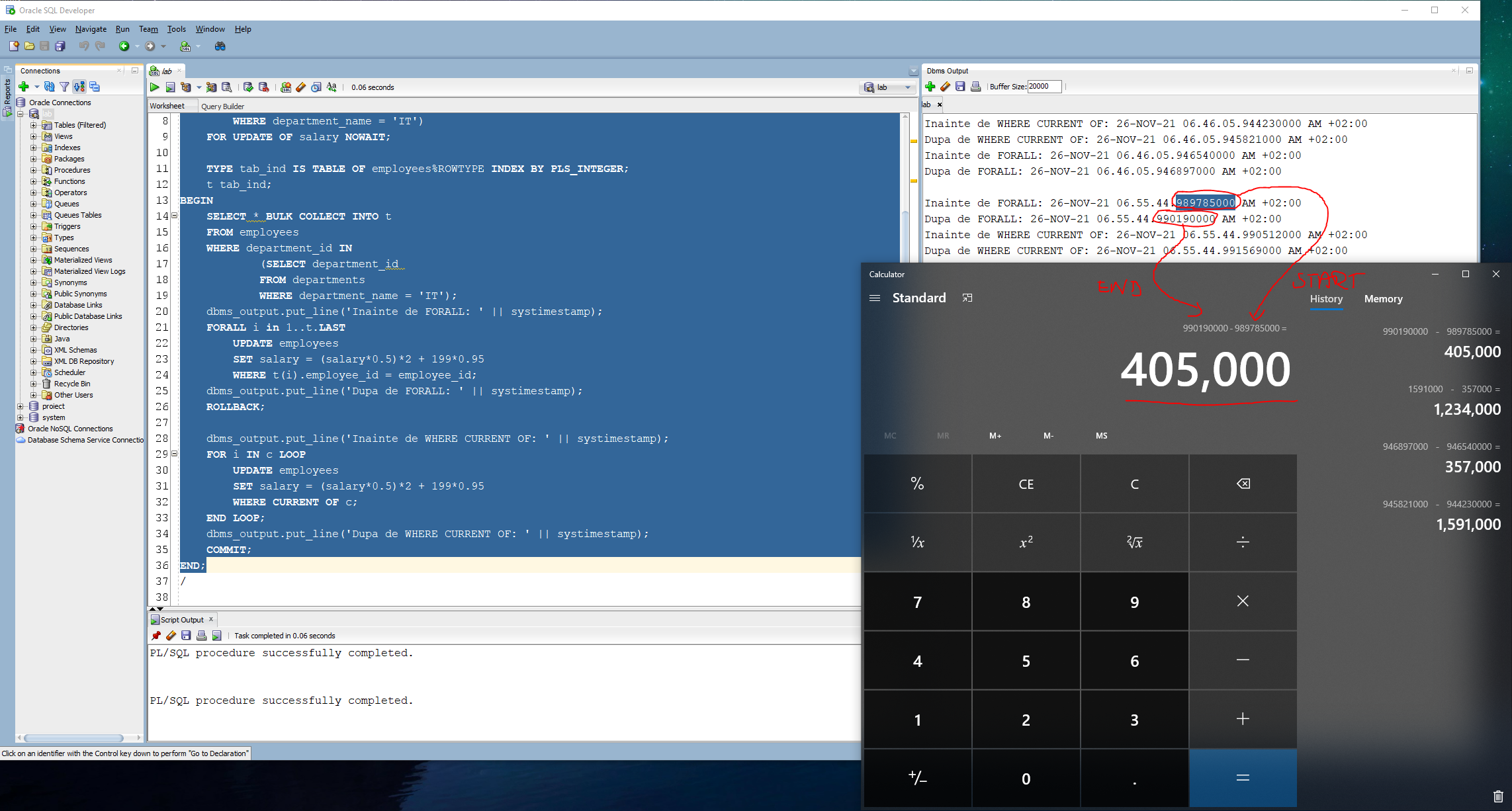
Observăm că a durat ***1.591.000*** unități (ordinul milisecundelor), loop-ul pentru cursorul explicit cu *WHERE CURRENT OF*. Acum vom face calculul pentru loop-ul *FORALL*.



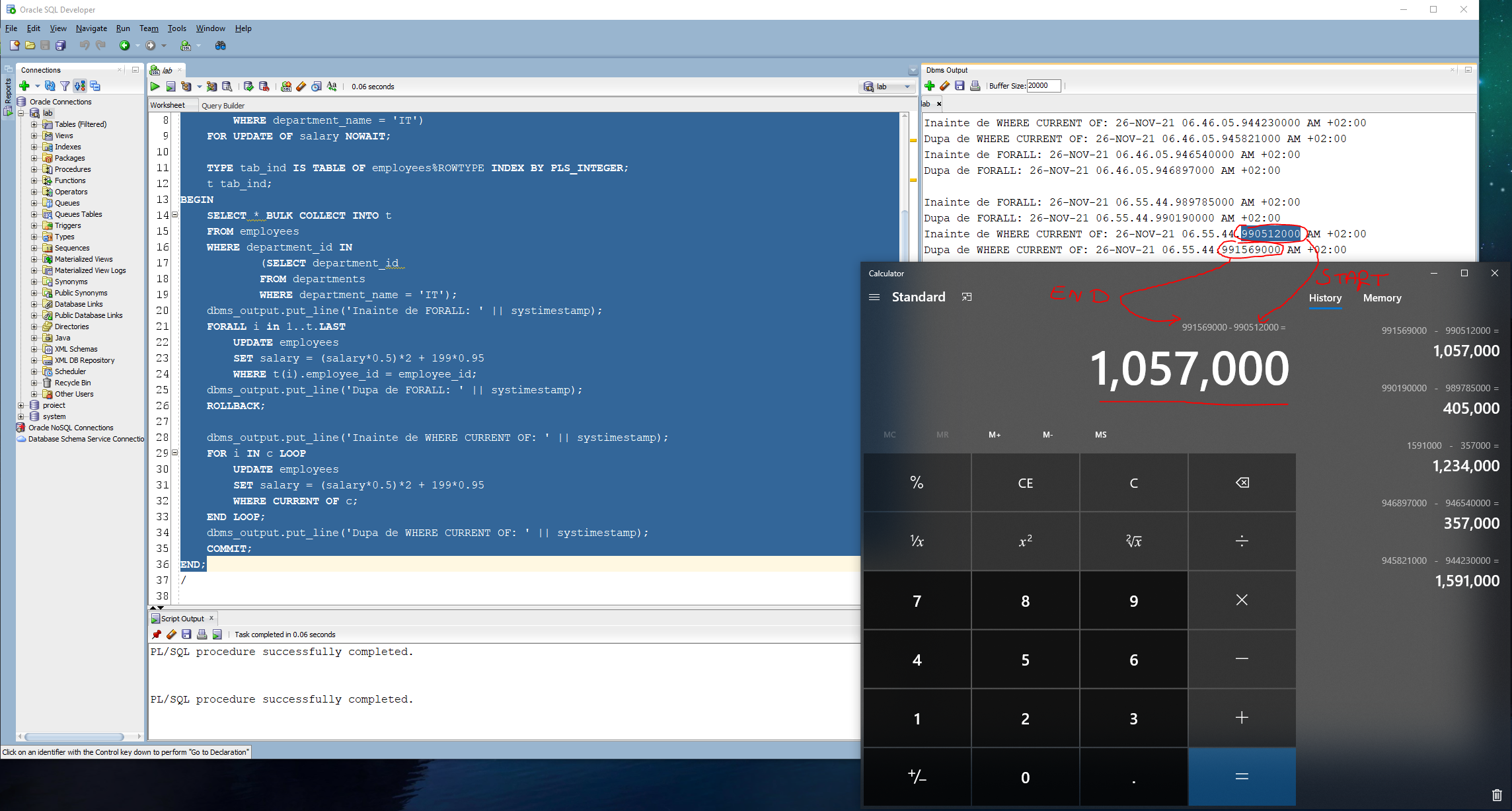
Observăm că loop-ul cu FORALL a durat ***357.000*** unități (ordinul milisecundelor). Deci s-a executat cu ***1.234.000*** de unități mai repede decât loop-ul cursorului cu *WHERE CURRENT OF*!



Dacă testăm mai întâi pentru *FORALL* și după accea pentru *WHERE CURRENT OF*, vom observa că indiferent de ordinea execuției *FORALL* se va executa mult mai repede decât *WHERE CURRENT OF*.



Avem ***405.000*** unități pentru *FORALL*.



Avem ***1.057.000*** unități pentru cursorul explicit, *WHERE CURRENT OF*. Deci din nou avem o diferență observabilă între cele două la nivelul microsecundelor de ***652.000*** unități.



De aici putem trage concluzia că, *BULK COLLECT INTO cu FORALL* (cursor implicit) este mult mai rapid decât *cursoarele explicite cu loop FOR*! Un singur dezavantaj pentru *FORALL* ar fi acela că nu putem folosi decât o instrucțiune DML, nu avem clauza ***FORALL END;***

***Exemplul 5.12 (WHERE CURRENT OF) vs Exemplul 5.13 (WHERE ROWID = i.ROWID) din Curs.***

Update-ul din exemplul 5.12 este echivalent cu cel din exemplul 5.13 (ambele funcționează). Eu consider că, cel din exemplul 5.12 (WHERE CURRENT OF) este optim în comparație cu cel din exemplul 5.13 (ROWID = i.ROWID) deoarece în cursor facem select doar pe o singură coloană, spre deosebire de exemplul 5.13, unde selectăm și id-ul produsului, dar și row id-ul, acest lucru aducând un consum mai mare de memorie la procesare (ocupă mai multă memorie, ceea ce nu ne dorim, știm acest lucru de la Workshop). De asemenea putem spune că, folosind ROWID (exemplul 5.13) nu avem nicio “garanție” că, până vom încheia noi tranzacția curentă, rowid-ul nu se va modifica. Acest lucru se poate întâmpla spre exemplu atunci când facem operații ALTER TABLE... SHRINK SPACE etc. (operații în masă sau DML-uri foarte mari).

***Exemplul 5.14, costul pentru fiecare dintre cele 3 instrucțiuni SQL.***

***Rezolvare:***

-- VARIANTA 1

SELECT \*

FROM employees e

WHERE EXISTS (SELECT 1

FROM departments d, job\_history j

WHERE e.employee\_id = j.employee\_id

AND d.department\_id = j.department\_id

AND TO\_CHAR(end\_date,'q') = 1);

-- VARIANTA 2

SELECT \*

FROM employees e

WHERE employee\_id IN (SELECT employee\_id

FROM departments d, job\_history j

WHERE e.employee\_id = j.employee\_id

AND TO\_CHAR(end\_date,'q') = 1);

-- VARIANTA 3

SELECT DISTINCT e.\*

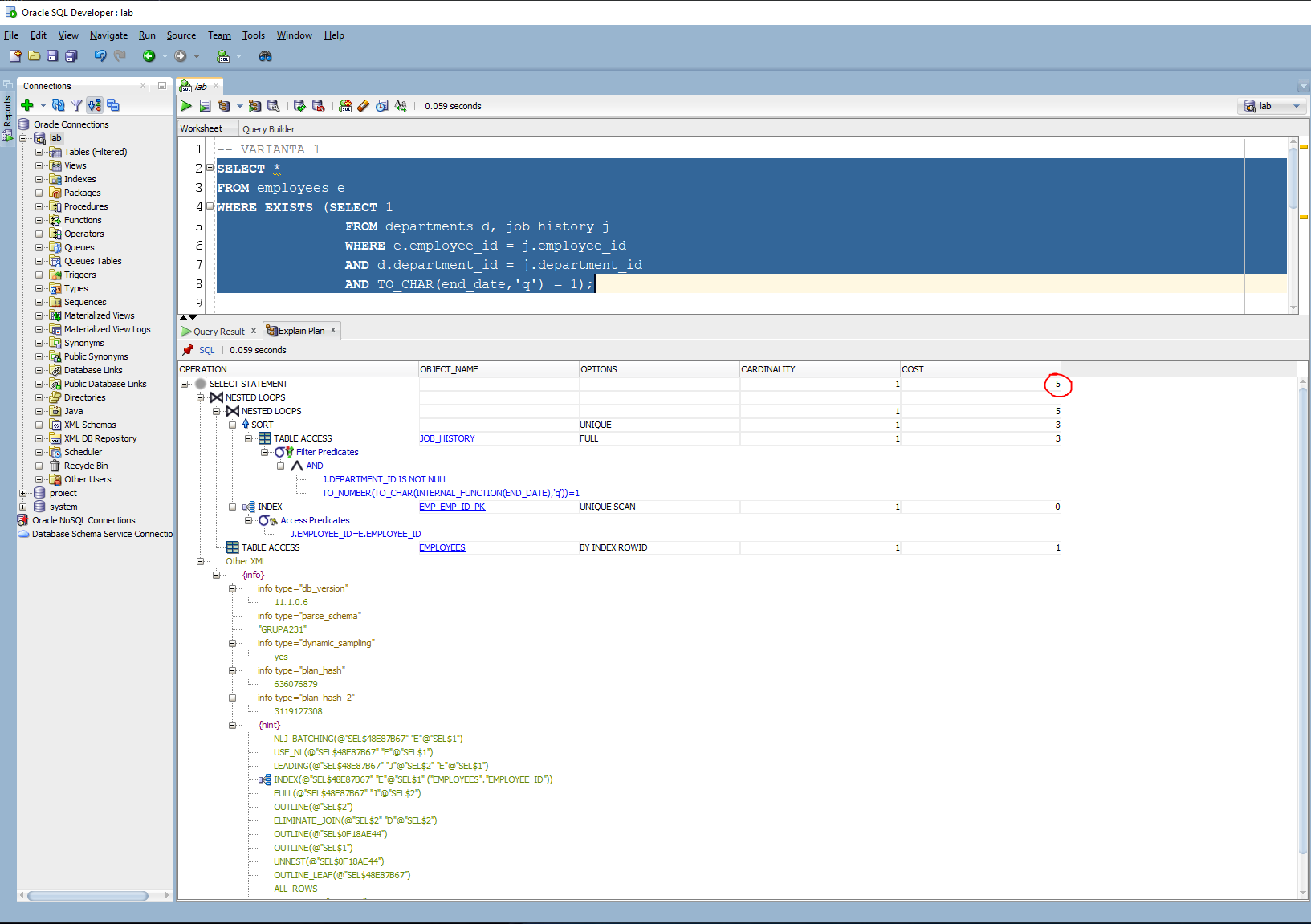
FROM employees e, departments d, job\_history j

WHERE e.employee\_id = j.employee\_id

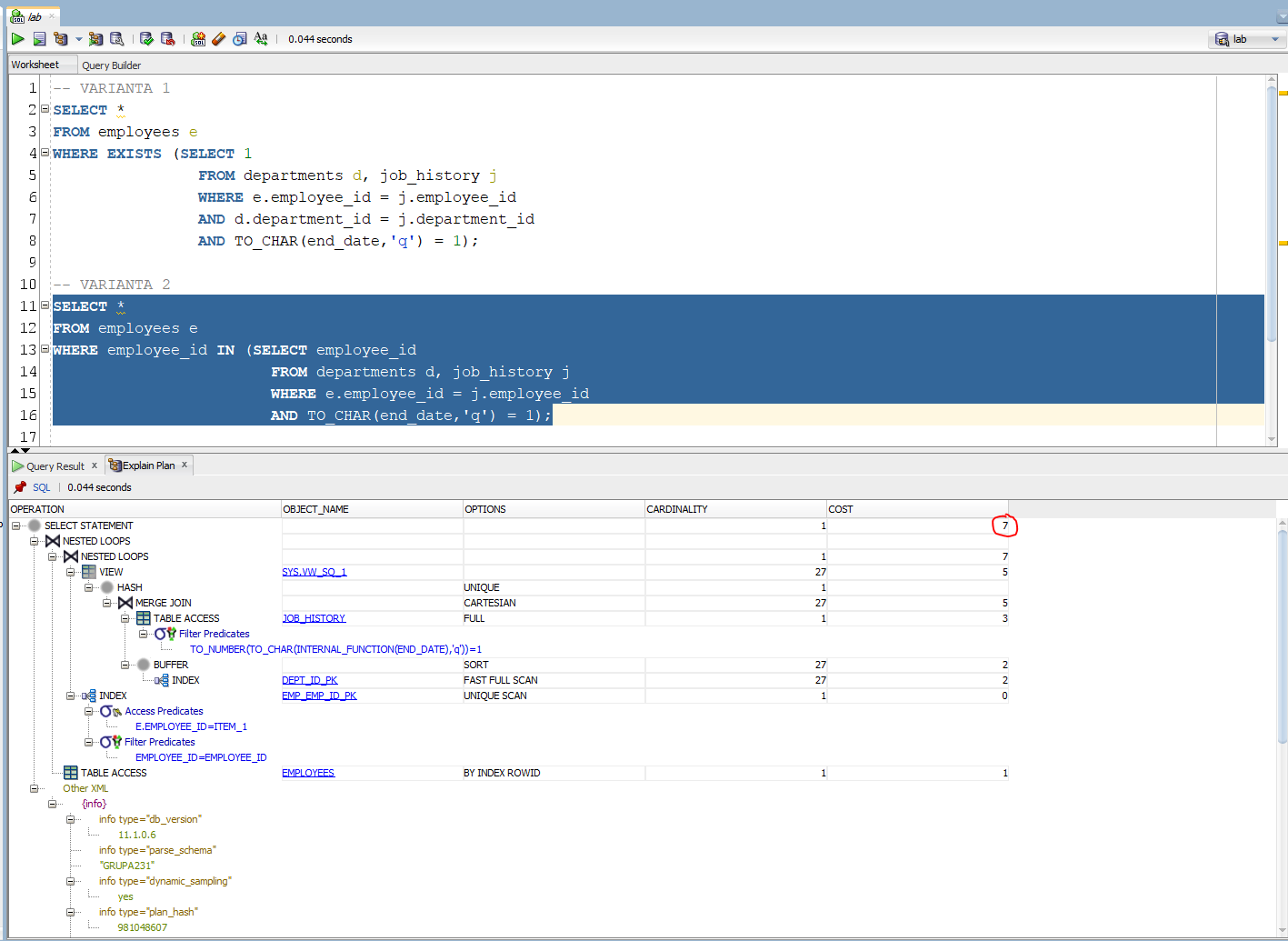
AND d.department\_id = j.department\_id

AND TO\_CHAR(end\_date,'q') = 1;

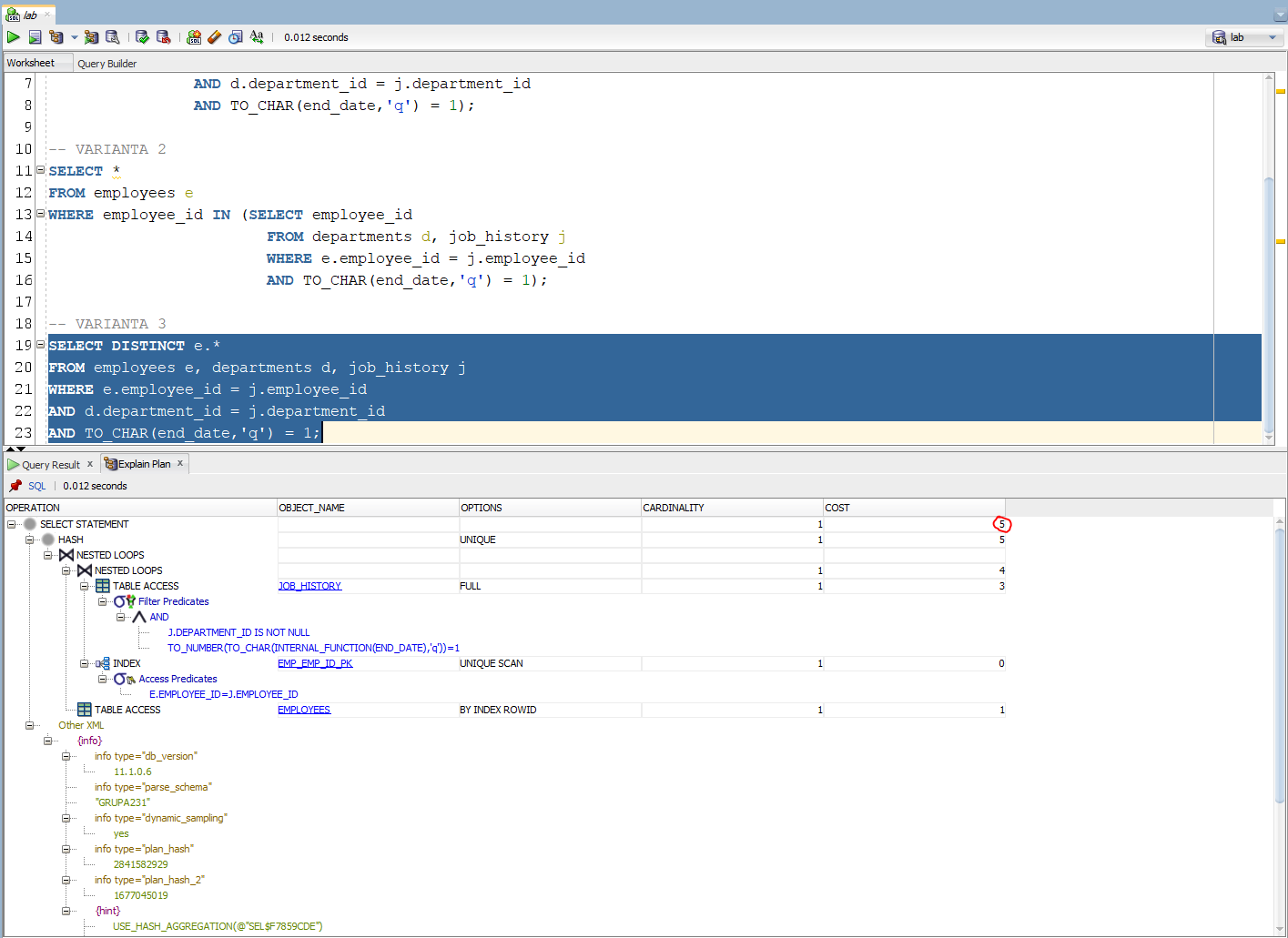
***Print-Screen:***



Pentru prima variantă avem costul 5.



Pentru a doua variantă avem costul 7.



Pentru a treia variantă avem costul 5.

Observăm că prima și a treia variantă au cel mai mic cost, 5. A doua variantă are costul cel mai mare, 7. Putem spune că prima variantă este puțin mai eficientă decât a treia (prima variantă costurile: 5, 3, 3. Total: 5; a treia variantă costurile: 5, 4, 3. Total 5);

***Cursoare imbricate vs Cursoare parametrizate (Exemplul 5.16 din Curs).***

Pentru a vedea diferențele între cele două cursoare vom folosi funcția ***systimestamp*** cu care vom vedea timpul la care a început/sfârșit excuția fiecăruia.

***Rezolvare:***

DECLARE

-- /CURSORUL IMBRICAT

CURSOR ang IS

SELECT first\_name,

CURSOR(SELECT MAX(department\_name)

FROM departments d, job\_history j

WHERE e.employee\_id = j.employee\_id

AND d.department\_id = j.department\_id

GROUP BY d.department\_id

ORDER BY 1, SUM(employee\_id) desc)

FROM employees e

WHERE salary > 3000;

c\_nume\_ang2 employees.first\_name%TYPE;

v\_cursor SYS\_REFCURSOR;

TYPE tab\_dept IS TABLE OF departments.department\_name%TYPE

INDEX BY BINARY\_INTEGER;

v\_dept tab\_dept;

-- /CURSORUL PARAMETRIZAT

CURSOR ang2 IS SELECT employee\_id, first\_name

FROM employees

WHERE salary > 3000;

CURSOR dept\_param(v\_ang2 employees.employee\_id%TYPE) IS

SELECT MAX(d.department\_name), SUM(employee\_id)

FROM departments d, job\_history j

WHERE v\_ang2 = j.employee\_id

AND d.department\_id = j.department\_id

GROUP BY d.department\_id

ORDER BY 1,2 desc;

c2\_nume employees.first\_name%TYPE;

c\_id employees.employee\_id%TYPE;

p\_dept\_nume departments.department\_name%TYPE;

p\_numar NUMBER;

BEGIN

-- CURSOR IMBRICAT START

dbms\_output.put\_line('Inainte de a deschide cursorul imbricat: ' || systimestamp);

OPEN ang;

LOOP

FETCH ang INTO c\_nume\_ang2, v\_cursor;

EXIT WHEN ang%NOTFOUND;

-- DBMS\_OUTPUT.PUT\_LINE(c\_nume\_ang2);

-- DBMS\_OUTPUT.PUT\_LINE('---------------------------');

FETCH v\_cursor BULK COLLECT INTO v\_dept LIMIT 3;

-- IF v\_dept.COUNT = 0 THEN

-- DBMS\_OUTPUT.PUT\_LINE('Nu avem angajati');

-- ELSE

-- FOR i IN v\_dept.FIRST..v\_dept.LAST LOOP

-- DBMS\_OUTPUT.PUT\_LINE(i|| '. '|| v\_dept(i));

-- END LOOP;

-- END IF;

--

-- DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

CLOSE ang;

dbms\_output.put\_line('Dupa ce s-a inchis cursorul imbricat: ' || systimestamp);

-- CURSOR IMBRICAT GATA

-- CURSOR PARAMETRIZAT START

dbms\_output.put\_line('Inainte de a deschide cursorul parametrizat: ' || systimestamp);

OPEN ang2;

LOOP

FETCH ang2 INTO c\_id, c2\_nume;

EXIT WHEN ang2%NOTFOUND;

-- DBMS\_OUTPUT.PUT\_LINE(c2\_nume);

-- DBMS\_OUTPUT.PUT\_LINE('------------------------');

OPEN dept\_param(c\_id);

LOOP

FETCH dept\_param INTO p\_dept\_nume, p\_numar;

EXIT WHEN dept\_param%NOTFOUND OR dept\_param%ROWCOUNT > 3;

END LOOP;

CLOSE dept\_param;

END LOOP;

CLOSE ang2;

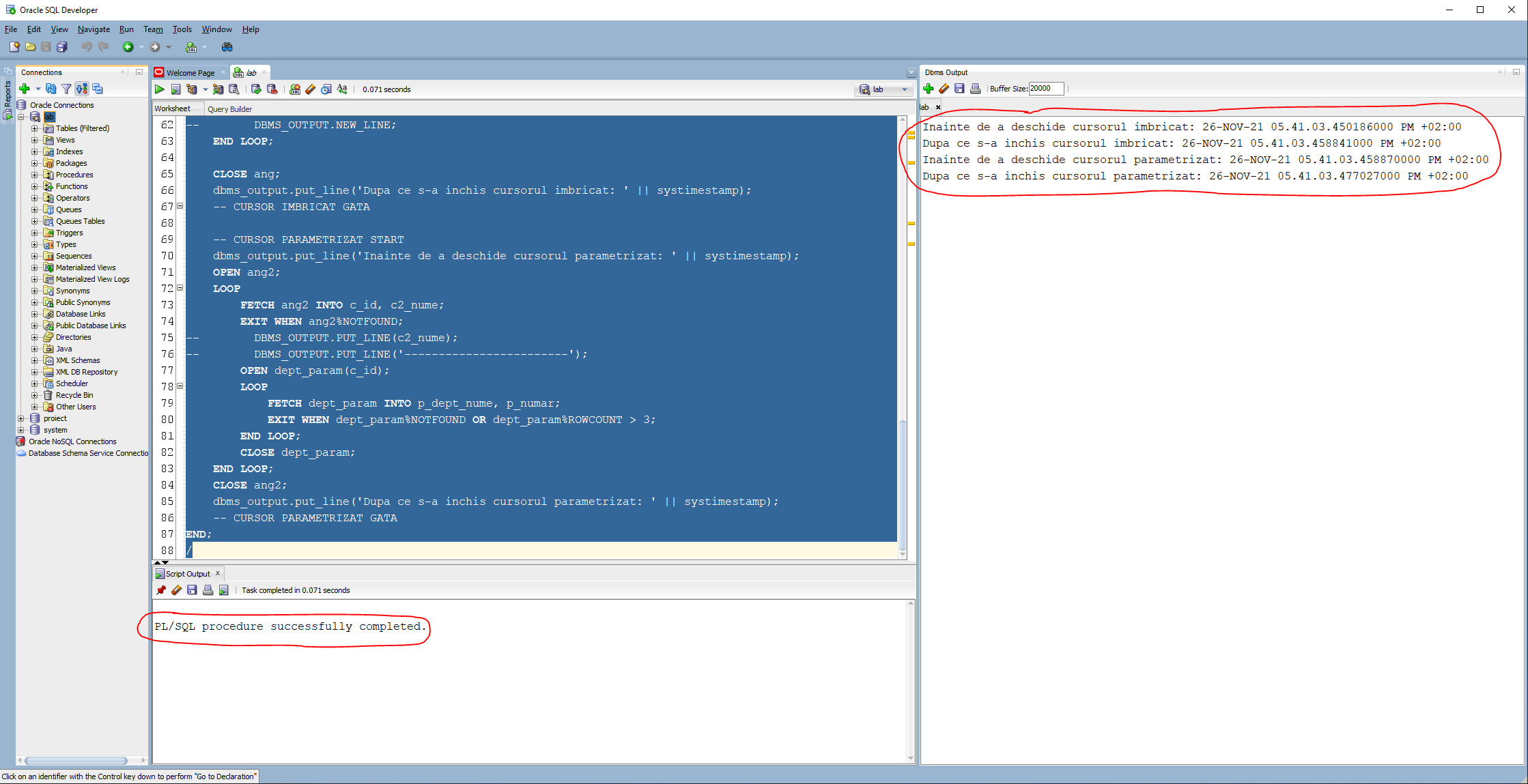
dbms\_output.put\_line('Dupa ce s-a inchis cursorul parametrizat: ' || systimestamp);

-- CURSOR PARAMETRIZAT GATA

END;

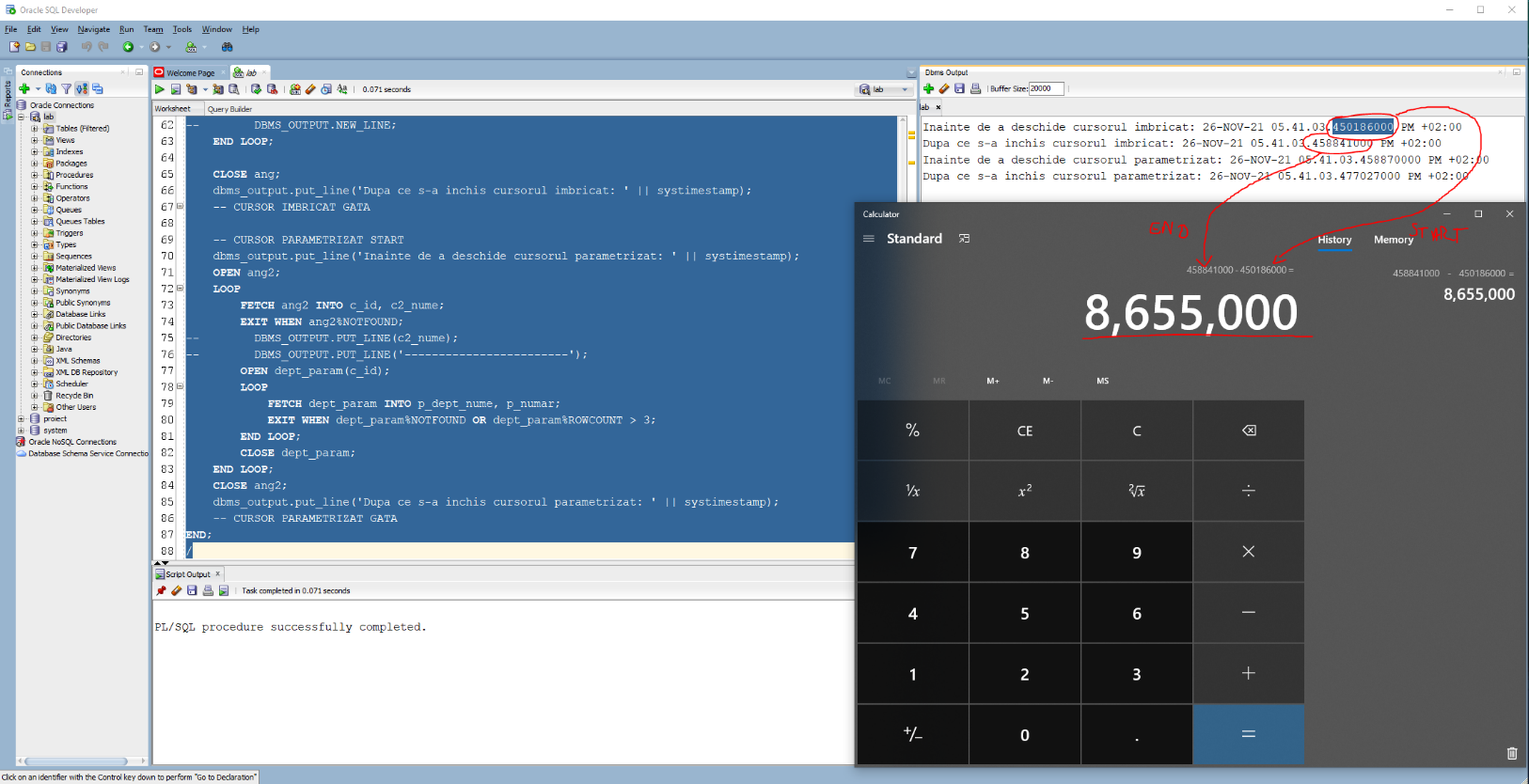
/

***Print-Screen:***

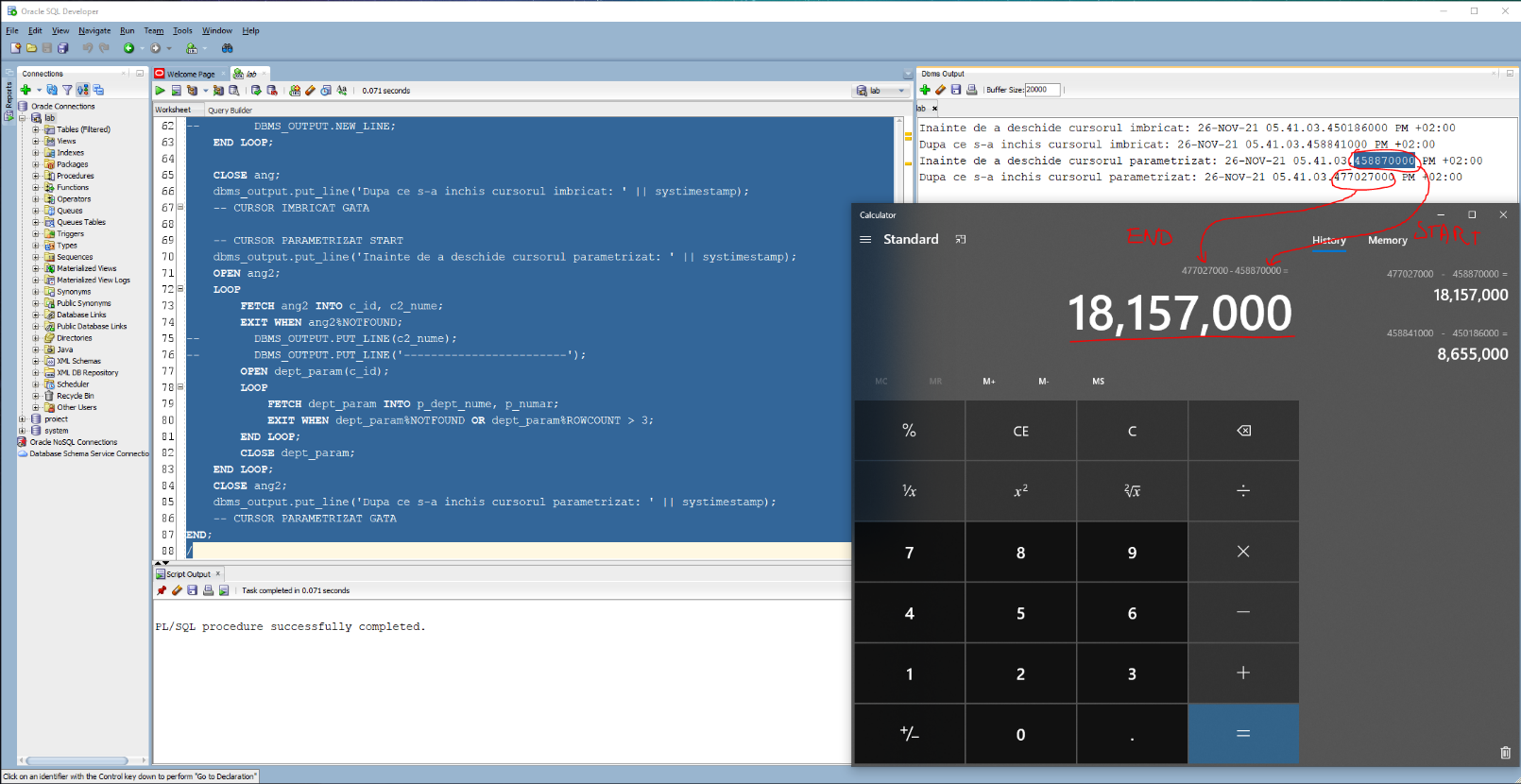


Putem observa că blocul nostru PL/SQL s-a executat cu succes. În dreapta vedem pe prima linie, respectiv a doua, ora la care a început/sfârșit monitorizarea pentru cursorul *imbricat*, respectiv *parametrizat*.

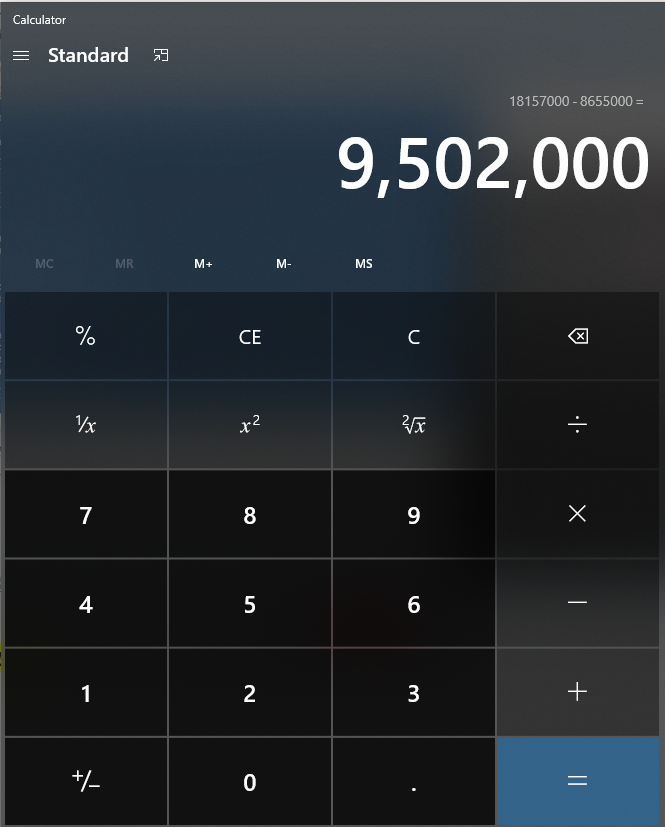
Diferențele sunt infime și în cazul acesta! Ele se pot observa la ordinul milisecundelor. Pentru a vedea cu exactitate vom face un diferența între timpul de început și timpul de sfâșit folosind calculatorul.



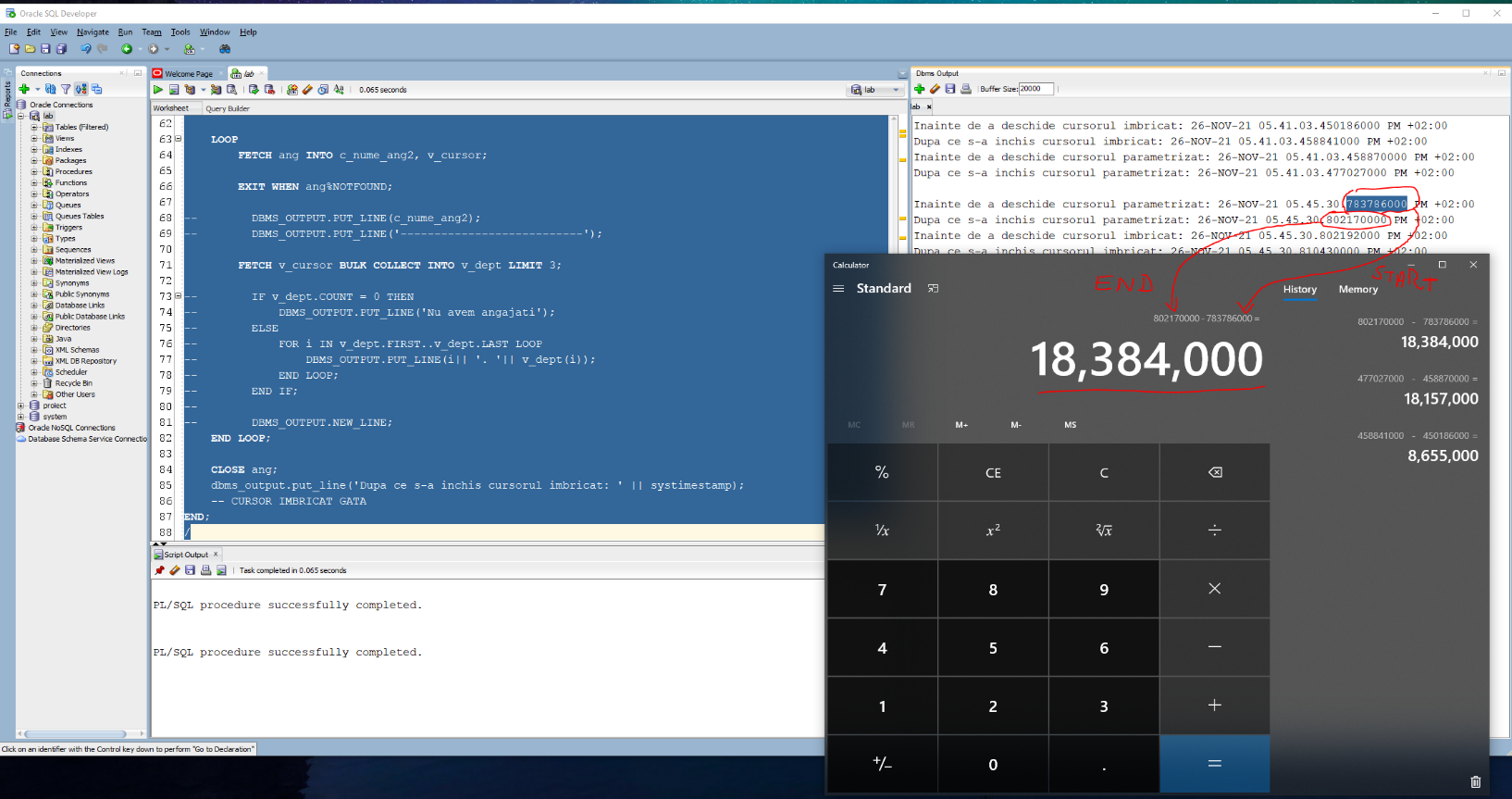
Observăm că a durat ***8.655.000*** unități (ordinul milisecundelor), execuția cursorului *imbricat*. Acum vom face calculul pentru cursorul *parametrizat*.



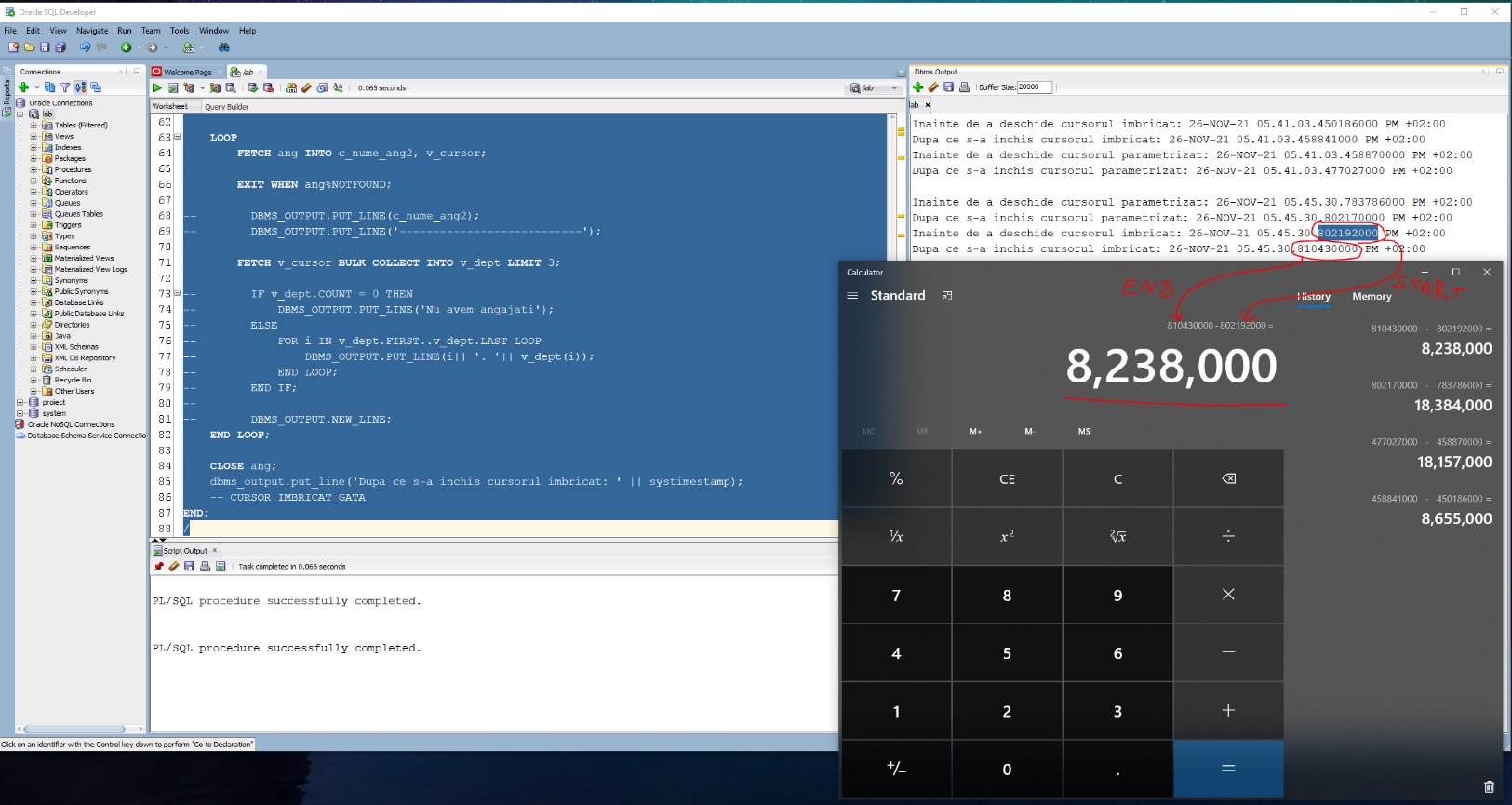
Observăm că cursorul *parametrizat* a durat ***18.157.000*** unități (ordinul milisecundelor). Deci s-a executat cu ***9.502.000*** de unități mai încet decât cursorul *imbricat*!



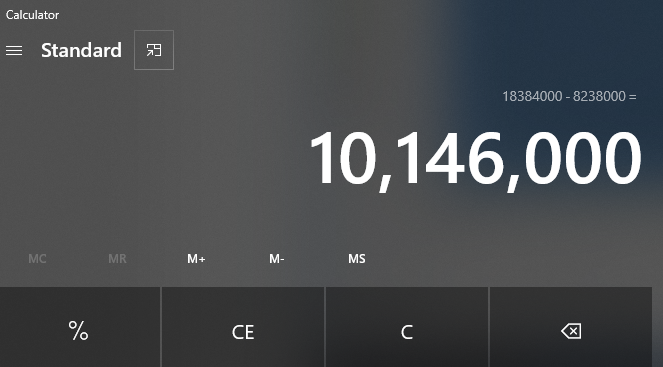
Dacă testăm mai întâi pentru cursorul *parametrizat* și după accea pentru cursorul *imbricat*, vom observa că indiferent de ordinea execuției cursorul *imbricat* se va executa mult mai repede decât cursorul *parametrizat*.



Avem ***18.384.000*** unități pentru cursorul parametrizat.



Avem ***8.238.000*** unități pentru cursorul *imbricat*. Deci din nou avem o diferență observabilă între cele două, la nivelul microsecundelor de ***10.146.000*** unități.



De aici putem trage concluzia că, cursoarele *imbricate* sunt mult mai rapide decât cursoarele *parametrizate*! De ce? Pentru că cursoarele *parametrizate* se comportă asemenea unor subprograme (se fac copii pe date, verificări pentru parametri/tipuri etc.).

Popescu Paullo Robertto Karloss

Grupa 231

Temă SGBD #8