Voiculescu Alina
Curs SGBD

CURRENT OF vs FORALL

Pentru a putea observa care dintre cele două variante este optimă, ne creăm un tabel numit CATEGORII (asemenea exemplului 5.12), în care ne inserăm 1.000.000 înregistrări pentru a putea vizualiza diferența timpului de rulare dintre cele două.

```
create table categorii ( id_categorie int primary key,
                         denumire varchar2(40),
                         id_parinte int default null);

insert into categorii (id_categorie)
select level
from dual
connect by level <= 1000000;
```

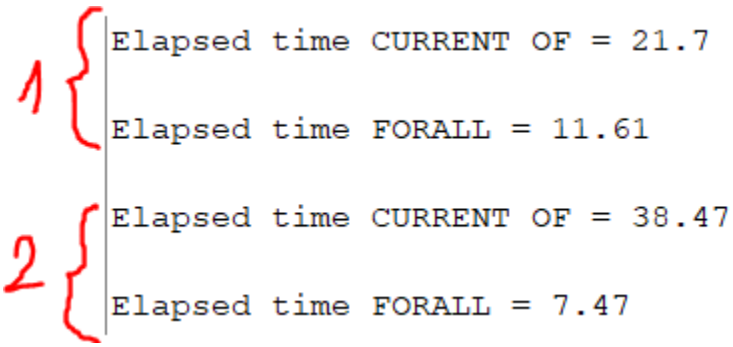
CURRENT OF:

```
DECLARE
    TYPE tab_imb IS TABLE OF categorii%ROWTYPE;
    v_categorii tab_imb;

    CURSOR c IS
            SELECT * FROM categorii
            WHERE id_parinte IS NULL
            FOR UPDATE of denumire NOWAIT;


    time1 NUMBER := dbms_utility.get_time;
BEGIN
    FOR i IN c LOOP
        UPDATE categorii
        SET denumire = 'Modificata de CURRENT OF'
        WHERE CURRENT OF c;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Elapsed time CURRENT OF = ' || (dbms_utility.get_time - time1)/100);
END;
/
```

FORALL:

```
DECLARE
    TYPE tab_imb IS TABLE OF categorii%ROWTYPE;
    v_categorii tab_imb;

    time1 NUMBER := dbms_utility.get_time;
BEGIN
    SELECT * BULK COLLECT INTO v_categorii
    FROM categorii
    WHERE id_parinte IS NULL;

    FORALL i IN v_categorii.first..v_categorii.last
        UPDATE categorii
        SET denumire = 'Modificata de FORALL'
        WHERE id_categorie = v_categorii(i).id_categorie;

    DBMS_OUTPUT.PUT_LINE('Elapsed time FORALL = ' || (dbms_utility.get_time - time1)/100);

    COMMIT;
END;
/
```

După rularea blocurilor (am rulat cele două blocuri de două ori, pentru a vizualiza mai bine diferența dintre acestea):



Așadar, rezultatul a fost cel așteptat: FORALL este mai rapid decât CURRENT OF.

Voiculescu Alina
Curs SGBD

Cursor imbricat vs Cursor parametrizat

Pentru a putea observa care dintre cele două variante este optimă preluăm rezolvările de la exercițiul 1 din laboratorul 3. Ne ajutăm de SYSTIMESTAMP pentru a vedea timpul exact de rulare (pe înregistrări atât de puține, funcția DBMS_UTILITY.GET_TIME ar returna 0).

CURSOR IMBRICAT:

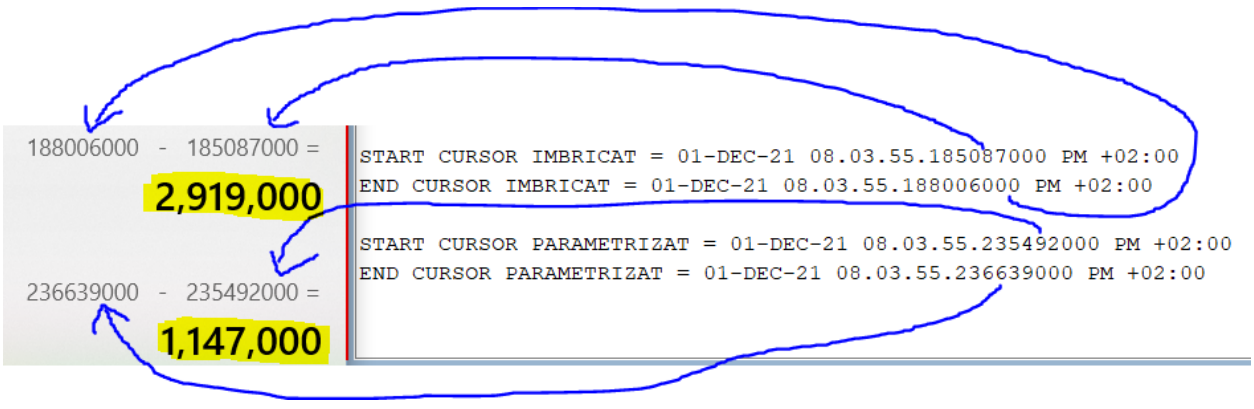
```
DECLARE
    type refcursor is ref cursor;
    cursor c_job is select job_title,
                    cursor (select last_name, salary
                            from employees e
                            where e.job_id = j.job_id)
                    from jobs j;
    nume employees.last_name%type;
    salariu employees.salary%type;
    titlu_job jobs.job_title%type;
    v_cursor refcursor;
BEGIN
    dbms_output.put_line('START CURSOR IMBRICAT = ' || systimestamp);
    open c_job;
    loop
        fetch c_job into titlu_job, v_cursor;
        exit when c_job%notfound;
        loop
            fetch v_cursor into nume, salariu;
            exit when v_cursor%notfound;
        end loop;
    end loop;
    close c_job;
    dbms_output.put_line('END CURSOR IMBRICAT = ' || systimestamp);
END;
/
```

CURSOR PARAMETRIZAT:

```
--cursor parametrizat
DECLARE
    cursor c1 is select job_id, job_title
                 from jobs;
    cursor c2(job_curent varchar2) is select last_name, salary
                                      from employees
                                      where job_id = job_curent;
    nr number;
BEGIN
    dbms_output.put_line('START CURSOR PARAMETRIZAT = ' || systimestamp);
    for i in c1 loop
        nr := 0;
        for j in c2(i.job_id) loop
            nr := nr + 1;
        end loop;
    end loop;
    dbms_output.put_line('END CURSOR PARAMETRIZAT = ' || systimestamp);
END;
/
```

După rularea blocurilor:

```
START CURSOR IMBRICAT = 01-DEC-21 08.03.55.185087000 PM +02:00
END CURSOR IMBRICAT = 01-DEC-21 08.03.55.188006000 PM +02:00

START CURSOR PARAMETRIZAT = 01-DEC-21 08.03.55.235492000 PM +02:00
END CURSOR PARAMETRIZAT = 01-DEC-21 08.03.55.236639000 PM +02:00
```



Așadar, observăm că al doilea cursor (cel parametrizat) este mai rapid decât cel imbricat.