# Ex 5.11

Avand 2 sesiuni, rulez in prima:

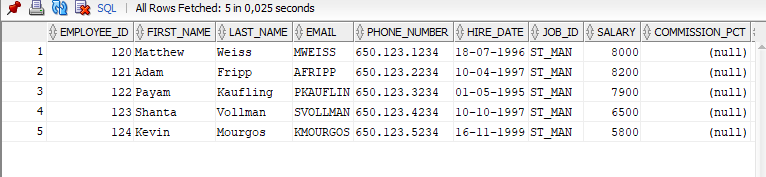
Ceea ce blocheaza liniile corespunzatoare angajatilor cu job\_id ST\_MAN.

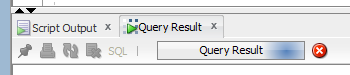
select \*

from emp\_bma

where job\_id = 'ST\_MAN'

for update;

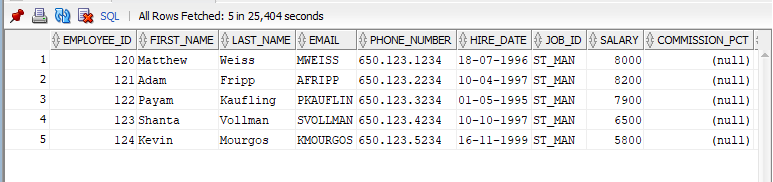
Fiind blocate, daca incerc si din a doua sesiune sa le blochez folosind aceeasi comanda, voi observa ca sql developer-ul va ramane blocat incercand sa acceseze acele linii blocate



pana cand voi anula cererea, sau pana cand le eliberez din prima sesiune:

Acum ca s-au eliberat din prima sesiune, in sesiunea a 2a se va executa comanda ceruta si va bloca din nou liniile.

commit;



Daca rulez acum in prima sesiune:

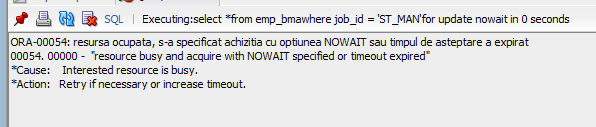
Observ ca imi va aparea o eroare care imi spune ca liniile sunt blocate si ca nu le va astepta dupa cum i-am specificat folosind argumentul NOWAIT.

select \*

from emp\_bma

where job\_id = 'ST\_MAN'

for update nowait;

In schimb, daca folosesc argumentul WAIT:

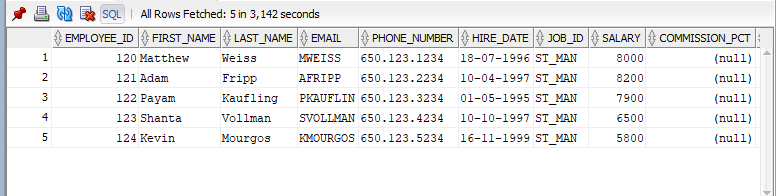
Sql developer-ul va astepta maxim 10 secunde in cazul in care liniile sunt blocate,

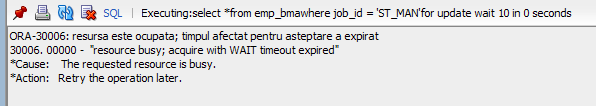
select \*

from emp\_bma

where job\_id = 'ST\_MAN'

for update wait 10;

si daca nu se deblocheaza in acest timp va aparea din nou o eroare.

Mai exista si posibilitatea folosirii argumentului SKIP LOCKED:

select \*

from emp\_bma

where job\_id = 'ST\_MAN'

for update skip locked;

care returneaza doar liniile ce nu sunt blocate.

In acest caz nu returneaza nimic pentru ca incerc sa blochez exact liniile care sunt blocate, dar daca incerc sa blochez si alte linii:

Atunci voi selecta si bloca doar liniile angajatilor cu job\_id AD\_VP.

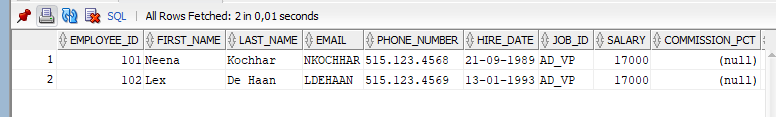
select \*

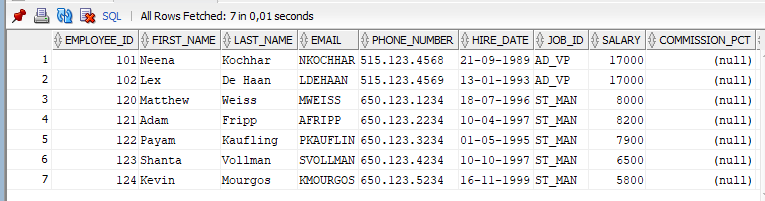
from emp\_bma

where job\_id = 'ST\_MAN'

or job\_id = 'AD\_VP'

for update skip locked;

Daca eliberez liniile din cealalta sesiune si repet ultima comanda, voi vedea ca toate liniile cerute vor fi afisate.



# Ex 5.12

Daca pun commit-ul in loop dupa update:

Se va face primul update si apoi se va da commit, ceea ce va duce la o eroare cand se va mai “incerca” un fetch (If the cursor has been opened with the FOR UPDATE clause, fetching after a COMMIT has been issued will return the error.)

DECLARE

CURSOR c IS

SELECT employee\_id

FROM emp\_bma

WHERE department\_id IN

(SELECT department\_id

FROM departments

WHERE department\_name = 'Executive')

FOR UPDATE OF salary NOWAIT;

BEGIN

FOR i IN c LOOP

UPDATE emp\_bma

SET salary = salary-2000

WHERE CURRENT OF c;

COMMIT;

END LOOP;

END;

/

Daca pun commit-ul inainte de insert atunci nu se va face niciun update si doar voi primi eroarea.

Se poate modifica si alta coloana decat cea alesa in for select:

Iar daca nu specific nicio coloana in for select rezultatul va fi acelasi, deoarece este referit doar un tabel.

DECLARE

CURSOR c IS

SELECT employee\_id

FROM emp\_bma

WHERE department\_id IN

(SELECT department\_id

FROM departments

WHERE department\_name = 'Executive')

FOR UPDATE OF salary NOWAIT;

BEGIN

FOR i IN c LOOP

UPDATE emp\_bma

SET department\_id = department\_id+10

WHERE CURRENT OF c;

END LOOP;

COMMIT;

END;

/