3. (-- mai trebuie modificat --)

CREATE OR REPLACE FUNCTION f3\_ggl (oras locations.city%type default 'Seattle')

RETURN NUMBER IS

v\_num number;

v\_2jobs number;

BEGIN

select count(\*)

into v\_num

from locations

where lower(city) = lower(oras);

if v\_num = 0 then

insert into info\_ggl values(user, sysdate, 'select count(\*)',

v\_num, 'NU exista Oras!');

else

select count(employee\_id)

into v\_num

from emp\_ggl join departments using(department\_id)

join locations using(location\_id)

where lower(city) = lower(oras);

if v\_num = 0 then

insert into info\_ggl values(user, sysdate, 'select count(employee\_id)',

v\_num, 'NU exista Angajati in Oras.');

else

select count(employee\_id)

into v\_2jobs

from emp\_ggl join departments using(department\_id)

join locations using(location\_id)

join job\_history using(employee\_id)

where lower(city) = lower(oras)

group by employee\_id

having count(job\_history.job\_id) >= 2;

insert into info\_ggl values(user, sysdate, 'select count(count(\*))',

v\_2jobs, 'SUCCESS');

end if;

end if;

DBMS\_OUTPUT.PUT\_LINE(v\_2jobs);

return v\_2jobs;

END f3\_ggl;

/

select \* from info\_ggl;

delete from info\_ggl where datas >= '08-DEC-2022';

variable x number

execute :x := f3\_ggl('New York');

execute :x := f3\_ggl('London');

execute :x := f3\_ggl('Bucuresti');

execute :x := f3\_ggl('Oradea');

execute :x := f3\_ggl('Beijing');

drop function f3\_ggl;

-- exercitiul 3 idee rezolvare prof (foarte similar cu rezolvarea de mai sus)

CREATE OR REPLACE FUNCTION ex3\_prof(p\_city IN locations.city%TYPE)

RETURN INTEGER

IS

v\_nr\_linii NUMBER(4,0);

v\_location\_id locations.location\_id%TYPE;

BEGIN

--verifica ca exista un singur oras cu numele dat si trateaza celelalte cazuri

SELECT COUNT(location\_id)

INTO v\_nr\_linii

FROM locations

WHERE city = p\_city;

IF v\_nr\_linii = 0 THEN

INSERT INTO info\_prof(user\_name, cmd\_name, cmd\_error) values ('prof', 'exercitiul 3', 'nu exita un oras cu numele dat');

RETURN 0;

END IF;

IF v\_nr\_linii > 1 THEN

--TODO adapteaza si foloseste o colectie de location\_id?

INSERT INTO info\_prof(user\_name, cmd\_name, cmd\_error) values ('prof', 'exercitiul 3', 'exista mai multe orase cu numele dat');

RETURN 0;

END IF;

--TODO trateaza si insereaza separat daca niciun angajat nu lucreaza in orasul dat

--gaseste si returneaza numarul de angatati

SELECT location\_id

INTO v\_location\_id

FROM locations

WHERE city = p\_city;

SELECT COUNT(employee\_id)

INTO v\_nr\_linii

FROM employees e

WHERE

(SELECT COUNT(DISTINCT job\_id) FROM job\_history WHERE employee\_id = e.employee\_id) >= 2

AND

EXISTS (SELECT department\_id FROM departments WHERE department\_id = e.department\_id AND location\_id = v\_location\_id)

;

INSERT INTO info\_prof(user\_name, cmd\_name, cmd\_nr\_linii) values ('prof', 'exercitiul 3', v\_nr\_linii);

RETURN v\_nr\_linii;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Numarul este '|| ex3\_prof('Seattle'));

END;

/

SELECT \* FROM info\_prof ORDER BY cmd\_date DESC;

-- exercitiul 4 (idee prof functie recursiva de a verifica daca un angajat are un manager anume)

CREATE OR REPLACE PROCEDURE ex4\_prof(p\_manager\_id IN employees.employee\_id%TYPE)

IS

--functie care determina daca un emp param 1 are ca manager direct sau indirect pe emp param 2

FUNCTION has\_manager(p\_employee\_id IN employees.employee\_id%TYPE, p\_manager\_id IN employees.employee\_id%TYPE)

RETURN NUMBER

IS

v\_manager\_id employees.employee\_id%TYPE;

BEGIN

SELECT manager\_id

INTO v\_manager\_id

FROM employees

WHERE employee\_id = p\_employee\_id;

IF v\_manager\_id IS NULL THEN

RETURN 0;

ELSIF v\_manager\_id = p\_manager\_id THEN

RETURN 1;

ELSE

return has\_manager(v\_manager\_id, p\_manager\_id);

END IF;

END;

BEGIN

--TODO mareste salariul tutoror angajatilor cu managerul dat ca param si insereaza in tabela de info

DBMS\_OUTPUT.PUT\_LINE('Are manager ' || has\_manager('103',p\_manager\_id));

END;

/

BEGIN

ex4\_prof('100');

END;

/

-- ex 4 – Gheorghe Robert-Mihai (update: problema rezolvata integral)

CREATE OR REPLACE FUNCTION GET\_MANAGER (e\_id EMP\_MISU.employee\_id%TYPE, m\_id EMP\_MISU.manager\_id%TYPE)

RETURN NUMBER IS

pragma autonomous\_transaction;

manager\_id EMP\_MISU.manager\_id%TYPE;

BEGIN

SELECT MANAGER\_ID INTO manager\_id FROM EMP\_MISU WHERE e\_id = EMPLOYEE\_ID;

IF manager\_id IS NULL THEN

return 0;

ELSIF m\_id = manager\_id THEN

return 1;

ELSE

return GET\_MANAGER(manager\_id, m\_id);

END IF;

END;

CREATE OR REPLACE PROCEDURE p4\_misu (m\_id IN EMP\_MISU.manager\_id%TYPE)

IS

n\_rows NUMBER;

c\_user VARCHAR(30);

BEGIN

UPDATE EMP\_MISU

SET SALARY = SALARY + 0.1\*SALARY

WHERE GET\_MANAGER(EMPLOYEE\_ID, m\_id) = 1;

n\_rows := SQL%ROWCOUNT;

SELECT USER INTO c\_user FROM DUAL;

IF n\_rows = 0 THEN INSERT INTO INFO\_MISU VALUES(c\_user, current\_date, 'p4\_misu(' || m\_id || ')', n\_rows, 'No such manager.');

ELSE INSERT INTO INFO\_MISU VALUES(c\_user, current\_date, 'p4\_misu(' || m\_id || ')', n\_rows, NULL);

END IF;

END;

/

EXECUTE p4\_misu(103);