Proiect SGBD

Gestiunea datelor dintr-un lant hotelier

Profesor coordonator: Student:

Iuliana Botha Simionescu Alexandru

Grupa 1063, seria E

1. Descrierea problemei economice alese:

Pentru o functionare corespunzatoare a oricarei firme este necesara o baza de date care sa stocheze diverse informatii si sa tina evidenta tuturor activitatilor care au loc in cadrul firmei respective.

Am facut o baza de date a 4 hoteluri, care au acelasi patron.

Hotelul cu id-ul 1, denumit Hotel Central Calimanesti are 5 etaje si are 20 camere pe etaj/per total avand 100 camere, la parter fiind sala de mese si receptia, acesta are 89 angajati.

Hotelul cu id-ul 2, denumit Hotel Cozia are 8 etaje si are 25 camere pe etaj/per total avand 200 camere, la parter fiind sala de mese si receptia, acesta are 193 angajati.

Hotelul cu id-ul 3, denumit Hotel Caciulata are 7 etaje si are 25 camere pe etaj/per total avand 175 camere, la parter fiind sala de mese si receptia, acesta are 143 angajati.

Hotelul cu id-ul 4, denumit Hotel Oltul are 7 etaje si are 25 camere pe etaj/per total avand 175 camere, la parter fiind sala de mese si receptia, acesta are 155 angajati.

Tipurile camerelor puse la dispozitie sunt: single, double si family.

Camerele care au balcon sunt trecute cu nr 1, cele care nu au balcon cu 0.

Id-ul camerelor este compus din numarul alocat etajului si numarul camerelor: ca sa se diferentieze id-ul tuturor camerelor din cele 4 hoteluri, fiecare etaj are un numar, numaratoarea incepand cu primul etaj de la hotelul cu id-ul 1, si se termina cu ultimul etaj de la hotelul cu id-ul 4, in total in cele 4 hoteluri sunt 27 etaje, iar urmatoarele cifre din id sunt usor de descifrat facand referire la numarul camerei respective.

De ex: camera 20 de la etajul 5 din hotelul cu id-ul 2 , are id-ul 1020.

Id-ul ficarui obiect de inventar este compus din: fiecare obiect are un numar destinate, in camera se afla urmatoarele obiecte de inventar: 1 -> masa, 2 -> scaun, 3 -> pat, iar urmatorul numar face referire la id-ul camerei in care se gasesc obiectele de inventar respective.

De ex: in camera cu id-ul 104 in care se gaseste o masa, id-ul acelei mese o sa fie 1104, primul 1 venind de la masa, am explicat mai sus, iar 104 este id-ul camerei.

Id-ul clientului este numerotat de la 1, la infinit, numarul id-ului se opreste practic la ultimul client.

Id-ul angajatului are in compozitie urmatoarele: prima cifra se refera la id-ul hotelului in care se afla angajatul, iar urmatoarele numere sunt puse in ordine crescatoare.

De ex: angajatul Ionescu Mihai, din cadrul hotelului cu id-ul 1, are id-ul: 111.

Id-ul rezervare, este pus in ordine crescatoare incepand cu nr 1111, pana la infinit.

Id-ul recenzie, este pus in ordine crescatoare incepand cu nr 11111, pana la infinit.

Calificativele recenziilor fiind: Foarte slab, slab, multumit, foarte multumit. Foarte slab fiind valorificat cu o stea, slab cu doua, multumit cu trei, foarte multumit cu maxim, adica cu 4 stele.

B. Schema conceptuala a bazei de date:

A diagram of a data flow

Description automatically generated

1. Interacţiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD şi LMD):

--1. Intr-un bloc PL/SQL sa se mareasca pretul camerelor cu 2% al caror pret este mai mic de 200 lei:

SET SERVEROUTPUT ON

DECLARE

v\_marire NUMBER := 0.02;

BEGIN

UPDATE P\_Camera

SET pret = pret \* (1 + v\_marire)

WHERE pret < 200;

DBMS\_OUTPUT.PUT\_LINE('Preturile camerelor au fost actualizate cu success!');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A apărut o eroare: ' || SQLERRM);

END;

/

A white background with black text

Description automatically generated

--Vizualizare preturi inainte de update:

SELECT \* FROM P\_CAMERA;

A screenshot of a computer

Description automatically generated

--Vizualizare preturi dupa update:

A screenshot of a table

Description automatically generated

--Putem observa ca pretul camerelor cu ID-ul 1020, 1909, 2412, s-au modificat.

--2. Sa se afiseze printr-un bloc PL/SQL, datele de contact(numarul de telefon si adresa de e-mail) al angajatului cu ID-ul 411:

SET SERVEROUTPUT ON

DECLARE

v\_telefon p\_angajat.telefon%TYPE;

v\_email p\_angajat.email%TYPE;

BEGIN

SELECT telefon, email INTO v\_telefon, v\_email

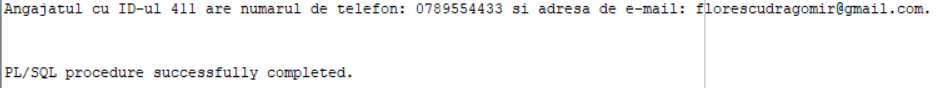
FROM p\_angajat

WHERE id\_angajat = 411;

DBMS\_OUTPUT.PUT\_LINE('Angajatul cu ID-ul 411 are numarul de telefon: '|| v\_telefon ||' si adresa de e-mail: '|| v\_email || '.');

END;

/



--3. Sa se insereze in tabela p\_client, folosind un bloc PL/SQL, un nou nou client cu numele: Calin, prenumele: Andrei, telefon: 0731425362, email: [calinandrei@gmail.com](mailto:calinandrei@gmail.com).

SET SERVEROUTPUT ON

DECLARE

v\_id p\_client.id\_client%TYPE;

BEGIN

SELECT MAX(id\_client) INTO v\_id

FROM p\_client

INSERT INTO p\_client VALUES(v\_id + 1, ‘Calin’, ‘Andrei’ , 0731425362, ‘[calinandrei@gmail.com](mailto:calinandrei@gmail.com)’);

END;

/



A screenshot of a computer

Description automatically generated

--4. Printr-un bloc PL/SQL sa se modifice toate denumirile obiectelor de inventar astfel incat sa fie scrise cu litere mari: pat -> PAT.

SET SERVEROUTPUT ON

BEGIN

UPDATE p\_obiecte\_inventar

SET nume = UPPER(nume);

DBMS\_OUTPUT.PUT\_LINE('Toate valorile din coloana "nume" au fost convertite la majuscule.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END;

/

--inainte ca modificarea sa aiba loc:

A screenshot of a computer

Description automatically generated

--dupa ce modificarea a avut loc:

A screenshot of a computer

Description automatically generated

--5. Printr-un bloc PL/SQL sa se actualizeze tipul de camera din single in family, pentru camera cu id-ul 104:



--Tipul camerei inainte de a se face modificarea:



--Tipul camerei dup ace modificarea a fost facuta:

A screenshot of a computer

Description automatically generated

--6. Printr-un bloc PL/SQL sa se actualizeze pretul din 1800 in 300, pentru camera cu id-ul 104:

SET SERVEROUTPUT ON

DECLARE

v\_id p\_camera.id\_camera%TYPE := 104;

v\_pret p\_camera.pret%TYPE := 300;

BEGIN

UPDATE p\_camera

SET pret = v\_pret

WHERE id\_camera = v\_id AND pret = 1800;

END;

/



--Pretul camerei inainte de modificare:



--Pretul camerei dupa modificare:

A screenshot of a computer

Description automatically generated

1. Structuri de control:

--1. Sa se compare pretul camerelor cu id-ul introdus de la tastatura, cu o variabila egala cu 300 si afisati mesajul respectiv.

SET SERVEROUTPUT ON

ACCEPT p PROMPT 'Introduceti ID-ul camerei:'

DECLARE

v\_var NUMBER := 350;

v\_pret p\_camera.pret%TYPE;

BEGIN

SELECT pret INTO v\_pret FROM p\_camera WHERE id\_camera = &p;

IF v\_pret < v\_var THEN

DBMS\_OUTPUT.PUT\_LINE('Camera introdusa are pretul mai mic.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Camera introdusa are pretul mai mare.');

END IF;

END;

/

A screenshot of a computer error message

Description automatically generated

--Vizualizare pretul camerei 104:

A screenshot of a computer

Description automatically generated

--Vizualizare rezultat daca variabila data cu preul 350 este mai mic sau mai mare decat pretul camerei cu id-ul 104 introdus de la tastatura 300:

A white background with black text

Description automatically generated

--2. In functie de pretul camerelor, avand id-ul citit de la tastatura, afisati pe ecran noua valoare a pretului:

SET SERVEROUTPUT ON

ACCEPT p PROMPT 'Introduceti ID-ul camerei:'

DECLARE

v\_pret p\_camera.pret%TYPE;

BEGIN

SELECT pret INTO v\_pret FROM p\_camera WHERE id\_camera = &p;

DBMS\_OUTPUT.PUT\_LINE('Pretul initial: '||v\_pret);

CASE

WHEN v\_pret < 300 THEN v\_pret := v\_pret \* 1.3;

WHEN v\_pret BETWEEN 300 and 500 THEN v\_pret := v\_pret \* 1.2;

ELSE v\_pret := v\_pret \* 1.1;

END CASE;

DBMS\_OUTPUT.PUT\_LINE('Pretul final este egal cu: '||v\_pret);

END;

/

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

--3. Sa se afiseze in ordine camerele cu numarul intre 19 si 22, atat timp cat pretul acestora este mai mic decat pretul mediu.

SET SERVEROUTPUT ON

DECLARE

v\_pret p\_camera.pret%TYPE;

v\_mediu v\_pret%TYPE;

i NUMBER := 19;

BEGIN

LOOP

SELECT AVG(pret) INTO v\_mediu FROM p\_camera;

SELECT pret INTO v\_pret FROM p\_camera WHERE numar = i;

if v\_pret < v\_mediu AND i < 22 THEN

DBMS\_OUTPUT.PUT\_LINE('Camera cu numarul ' ||i|| ' are pretul: ' ||v\_pret || ' si satisface cerinta data.');

END IF;

i := i + 1;

EXIT WHEN i > 22;

END LOOP;

END;

/

--afisare pret mediu al camerelor:

A screenshot of a computer

Description automatically generated

--deci doar camera cu numarul 20 ar trebui sa satisfaca aceste cerinte:

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

--4. Sa se afiseze printr-un bloc PL/SQL numele, prenumele si domiciliul tuturor angajatilor folosind o structura repetitive de tip for:

SET SERVEROUTPUT ON

DECLARE

v\_min p\_angajat.id\_angajat%TYPE;

v\_max p\_angajat.id\_angajat%TYPE;

v\_nume p\_angajat.nume%TYPE;

v\_prenume p\_angajat.prenume%TYPE;

v\_domiciliu p\_angajat.domiciliu\_angajat%TYPE;

BEGIN

SELECT MIN(id\_angajat), MAX(id\_angajat) INTO v\_min, v\_max FROM p\_angajat;

FOR i IN v\_min..v\_max LOOP

BEGIN

SELECT nume, prenume, domiciliu\_angajat

INTO v\_nume, v\_prenume, v\_domiciliu

FROM p\_angajat

WHERE id\_angajat = i;

DBMS\_OUTPUT.PUT\_LINE('Angajatul cu numele: ' || v\_nume || ' ' || v\_prenume || ' are domiciliul in: ' || v\_domiciliu);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL;

END;

END LOOP;

END;

/

A screenshot of a computer

Description automatically generated

--5. Sa se afiseze toate hotelurile folosind o structura repetitive while:

SET SERVEROUTPUT ON

DECLARE

i p\_hotel.id\_hotel%TYPE;

v\_min p\_hotel.id\_hotel%TYPE;

v\_max p\_hotel.id\_hotel%TYPE;

v\_nume p\_hotel.nume%TYPE;

BEGIN

SELECT MIN(id\_hotel), MAX(id\_hotel) INTO v\_min, v\_max FROM p\_hotel;

i := v\_min;

WHILE i <= v\_max LOOP

SELECT nume INTO v\_nume FROM p\_hotel WHERE id\_hotel = i;

i := i + 1;

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

END LOOP;

END;

/

A white background with black text

Description automatically generated

--6. Sa se afiseze toti angajatii cu Id-ul intre 111 si 113 atat timp cat au fost angajati inainte de anul 2024:

SET SERVEROUTPUT ON

DECLARE

v\_data p\_angajat.data\_ang%TYPE;

i NUMBER := 111;

BEGIN

WHILE i <= 113 LOOP

SELECT data\_ang INTO v\_data FROM p\_angajat WHERE id\_angajat = i;

IF EXTRACT(YEAR FROM v\_data) < 2024 THEN

DBMS\_OUTPUT.PUT\_LINE('Angajatul cu ID-ul: ' ||i||' s-a angajat la data de '||v\_data);

END IF;

i := i + 1;

EXIT WHEN i = 114;

END LOOP;

END;

/

A close-up of a white background

Description automatically generated

1. Tratarea exceptiilor:

1. Exceptii implicite:

--1.1 Sa se afiseze numarul camerelor de un anumit tip introdus de la tastatura, daca aceasta nu exista sa se afiseze o exceptie implicita:

SET SERVEROUTPUT ON

ACCEPT tip PROMPT 'Introduceti tipul camerei:'

DECLARE

v\_tip\_camera p\_camera.tip\_camera%TYPE;

CURSOR c IS SELECT id\_camera FROM p\_camera WHERE tip\_camera = v\_tip\_camera;

v\_count NUMBER;

BEGIN

v\_tip\_camera := '&tip';

SELECT COUNT(\*) INTO v\_count FROM p\_camera

WHERE tip\_camera = v\_tip\_camera;

IF v\_count > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('S-au gasit inregistrate: ' || v\_count || ' camere de acest tip ' || v\_tip\_camera);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu exista acest tip de camera: ' || v\_tip\_camera);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista acest tip de camera: ' || v\_tip\_camera);

END;

Exemplu pentru un tip de camera existent:

A white background with black text

Description automatically generated

Exemplu pentru un tip de camera care nu exista:

A white background with black text

Description automatically generated

--1.2. Sa se afiseze angajatii care au salariul mai mare de 50000, in caz ca nu exista sa se afiseze o exceptie:

DECLARE

CURSOR c\_angajati IS

SELECT nume, prenume, salariul

FROM p\_angajat

WHERE salariul > 5000;

v\_nume p\_angajat.nume%TYPE;

v\_prenume p\_angajat.prenume%TYPE;

v\_salariu p\_angajat.salariul%TYPE;

BEGIN

OPEN c\_angajati;

LOOP

FETCH c\_angajati INTO v\_nume, v\_prenume, v\_salariu;

EXIT WHEN c\_angajati%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Angajatul: ' || v\_nume || ' ' || v\_prenume || ' ,are salariul: ' || v\_salariu);

END LOOP;

CLOSE c\_angajati;

IF c\_angajati%NOTFOUND THEN

RAISE invalid\_cursor;

END IF;

EXCEPTION

WHEN invalid\_cursor THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista angajati cu salariul mai mare de 5000!');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END;

/

A white background with black text

Description automatically generated

--1.3. Sa se insereze un nou client in tabela p\_client, sa se finalizeze cu succes atunci cand id-ul introdus nu exista deja, iar daca exista sa se arunce o exceptie:

ACCEPT id\_client PROMPT 'Introduceti id-ul clientului:'

ACCEPT nume\_client PROMPT 'Introduceti numele clientului:'

ACCEPT prenume\_client PROMPT 'Introduceti prenumele clientului:'

ACCEPT telefon\_client PROMPT 'Introduceti telefonul clientului:'

ACCEPT email\_client PROMPT 'Introduceti email-ul clientului:'

DECLARE

v\_id\_client P\_Client.id\_client%TYPE := &id\_client;

v\_nume P\_Client.nume%TYPE := '&nume\_client';

v\_prenume P\_Client.prenume%TYPE := '&prenume\_client';

v\_telefon P\_Client.telefon%TYPE := '&telefon\_client';

v\_email P\_Client.email%TYPE := '&email\_client';

BEGIN

INSERT INTO P\_Client(id\_client, nume, prenume, telefon, email)

VALUES (v\_id\_client, v\_nume, v\_prenume, v\_telefon, v\_email);

DBMS\_OUTPUT.PUT\_LINE('Clientul cu ID-ul ' || v\_id\_client || ' a fost inserat cu succes!');

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Clientul cu ID-ul ' || v\_id\_client || ' exista deja în baza de date!');

END;

/

Daca id-ul nu exista, iar clientul a fost adaugat cu success:

A white background with black text

Description automatically generated

Daca id-ul exista deja:

A close-up of a white background

Description automatically generated

--1.4. Sa se afiseze numele angajatului al carui id corespunde cu cel introdus de la tastatura in caz contrar sa se afiseze o exceptie:

ACCEPT id\_angajat PROMPT 'Introduceti id-ul angajatului:'

SET SERVEROUTPUT ON

DECLARE

v\_nume p\_angajat.nume%TYPE;

v\_id\_angajat p\_angajat.id\_angajat%TYPE := &id\_angajat;

v\_exista\_angajat BOOLEAN := FALSE;

CURSOR c\_angajat IS

SELECT nume

FROM p\_angajat

WHERE id\_angajat = v\_id\_angajat;

BEGIN

OPEN c\_angajat;

FETCH c\_angajat INTO v\_nume;

IF c\_angajat%FOUND THEN

v\_exista\_angajat := TRUE;

END IF;

CLOSE c\_angajat;

IF v\_exista\_angajat THEN

dbms\_output.put\_line('Numele angajatului cu codul ' || v\_id\_angajat || ' este: ' || v\_nume);

ELSE

dbms\_output.put\_line('Nu exista angajatul cu codul ' || v\_id\_angajat || '!');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Nu exista angajatul cu codul ' || v\_id\_angajat || '!');

END;

/

Atunci cand id-ul nu corespunde cu un angajat existent:

A white background with black text

Description automatically generated

Atunci cand angajatul cu id-ul introdus exista:

A white background with black text

Description automatically generated

2. Exceptii explicite:

--2.1. Sa se actualizeze adresa, unde id-ul clientului si adresa sa fie introdusa de la tastatura, sa se arunce o exceptie in cazul in care id-ul clientului nu exista

ACCEPT id PROMPT 'Introduceti id-ul clientului:'

ACCEPT email PROMPT 'Introduceti email-ul actualizat:'

DECLARE

invalid\_idf EXCEPTION;

v\_id p\_client.id\_client%TYPE := &id;

v\_email p\_client.email%TYPE := '&email'; -- Inconjuram valoarea introdusa de la tastatura cu ghilimele simple

BEGIN

UPDATE p\_client

SET email = v\_email

WHERE id\_client = v\_id;

IF SQL%NOTFOUND THEN

RAISE invalid\_idf;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Clientul s-a actualizat cu succes!');

END IF;

EXCEPTION

WHEN invalid\_idf THEN

DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit clientul cu acest id.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut eroarea: ' || SQLERRM);

END;

Daca id-ul este correct si s-a produs modificarea:

A white background with black text

Description automatically generated

Daca id-ul introdus este unul inexistent:

A screenshot of a computer error

Description automatically generated

--2.2. Creati un bloc PL/SQL prin care sa se creasca salariul cu 10% al angajatilor din hotelul al carui ID este introdus de la tastatura. Sa se afiseze numarul de inregistrari modificate si sa se ridice o exceptie in cazul in care nu sunt modificari de efectuat:

SET SERVEROUTPUT ON

ACCEPT P PROMPT 'Introduceti ID-ul hotelului:'

DECLARE

v\_id p\_hotel.id\_hotel%TYPE := '&p';

e EXCEPTION;

BEGIN

UPDATE p\_angajat

SET salariul = 1.10 \* salariul

WHERE id\_angajat IN (SELECT id\_angajat FROM p\_angajat WHERE id\_hotel = v\_id);

IF SQL%ROWCOUNT = 0

THEN RAISE e;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Numarul de modificari produse este: '|| sql%rowcount);

END IF;

EXCEPTION

WHEN e THEN

DBMS\_OUTPUT.PUT\_LINE('Nu s-a produs nicio modificare.');

END;

/

--Atunci cand ID-ul hotelului exista:

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

--Atunci cand ID-ul hotelului nu exista:

A screenshot of a computer

Description automatically generated

A close-up of words

Description automatically generated

--2.3. Sa se afiseze pretul tuturor camerelor de acelasi tip, pe rand, iar daca acesta nu exista sa se ridice o exceptie si sa se afiseze un mesaj:

SET SERVEROUTPUT ON

DECLARE

v\_pret p\_camera.pret%TYPE;

e\_exceptie EXCEPTION;

CURSOR c\_camere IS

SELECT pret FROM p\_camera

WHERE tip\_camera LIKE 'single%';

BEGIN

OPEN c\_camere;

FETCH c\_camere INTO v\_pret;

IF c\_camere%NOTFOUND THEN

RAISE e\_exceptie;

ELSE

LOOP

EXIT WHEN c\_camere%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Pretul camerei single este: ' || v\_pret);

FETCH c\_camere INTO v\_pret;

END LOOP;

END IF;

CLOSE c\_camere;

EXCEPTION

WHEN e\_exceptie THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista camere de tipul selectat.');

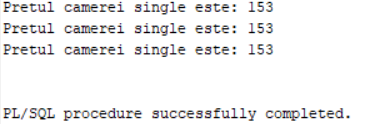
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END;

/

--Daca tipul selectat exista:



--Iar daca acesta nu exista:

A white background with black text

Description automatically generated

1. Gestionarea cursorilor:

1. Cursori impliciti:

--1.1. Sa se mareasca cu 20% pretul camerelor al carui tip este introdus de la tastatura:

SET SERVEROUTPUT ON

ACCEPT g\_tip\_camera PROMPT 'Introduceti tipul camerei:'

DECLARE

v\_tip\_camera p\_camera.tip\_camera%TYPE :=&g\_tip\_camera;

v\_test NUMBER;

BEGIN

UPDATE p\_camera

SET pret = pret + 0.2 \* pret

WHERE tip\_camera = v\_tip\_camera;

IF SQL%NOTFOUND THEN

SELECT COUNT(\*) INTO v\_test

FROM p\_camera

WHERE tip\_camera = v\_tip\_camera;

IF v\_test = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista acest tip de camera');

END IF;

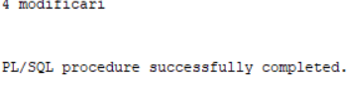
ELSE

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' modificari');

END IF;

END;

In exemplul de mai sus am schimbat pretul camerelor single, crescand pretul de la 150 lei cu 20% la 180 lei.



Iar in exemplul de mai jos am introdus un tip de camera care nu exista:

A white background with black text

Description automatically generated

--1.2. Sa se mareasca cu 10% salariile angajatilor care au deja salariul peste medie:

DECLARE

v\_salariu\_mediu NUMBER;

BEGIN

SELECT AVG(salariul) INTO v\_salariu\_mediu FROM P\_Angajat;

UPDATE P\_Angajat

SET salariul = salariul \* 1.1

WHERE salariul > v\_salariu\_mediu;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Salariile angajatilor cu salariu peste medie au fost actualizate.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu exista angajați cu salariu peste medie.');

END IF;

END;

A white background with black text

Description automatically generated

--1.3. Sa se se introduca un salariul minim care sa fie inlocuit cu un salariu nou si sa se arate cate actualizari s-au facut:

ACCEPT salariu\_minim\_introdus PROMPT 'Introduceti salariul minim de la care se va face schimbarea:'

ACCEPT salariu\_nou\_introdus PROMPT 'Introduceti salariul nou:'

DECLARE

v\_salariu\_nou NUMBER;

v\_salariu\_minim NUMBER;

BEGIN

v\_salariu\_minim := &salariu\_minim\_introdus;

v\_salariu\_nou := &salariu\_nou\_introdus;

UPDATE P\_Angajat

SET salariul = v\_salariu\_nou

WHERE salariul < v\_salariu\_minim;

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' angajati au fost actualizati cu noul salariu.');

END;

A white background with black text

Description automatically generated

2. Cursori expliciti:

--2.1. Sa se afiseze clientii care au facut o rezervare :

DECLARE

v\_id\_client NUMBER;

v\_nume\_client p\_client.nume%TYPE;

CURSOR c\_clienti\_rezervari IS

SELECT DISTINCT id\_client, nume || ' ' || prenume AS nume\_complet

FROM P\_Client

WHERE id\_client IN (SELECT id\_client FROM P\_Rezervari);

BEGIN

OPEN c\_clienti\_rezervari;

LOOP

FETCH c\_clienti\_rezervari INTO v\_id\_client, v\_nume\_client;

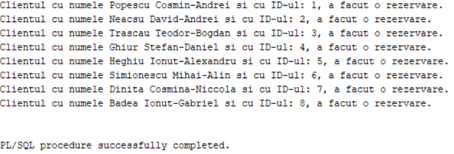
EXIT WHEN c\_clienti\_rezervari%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Clientul cu numele ' || v\_nume\_client || ' si cu ID-ul: ' || v\_id\_client || ', a facut o rezervare.');

END LOOP;

CLOSE c\_clienti\_rezervari;

END;



--2.2. Sa se afiseze rezervarile care au fost facute in anul 2022:

SET SERVEROUTPUT ON

DECLARE

CURSOR c1 IS SELECT id\_rezervare, data\_check\_in FROM P\_REZERVARI

WHERE DATA\_CHECK\_IN BETWEEN TO\_DATE('01.01.2022','DD.MM.YYYY') AND TO\_DATE('31.12.2022','DD.MM.YYYY');

rec\_rez c1%ROWTYPE;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO rec\_rez;

EXIT WHEN c1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Rezervarea '||rec\_rez.id\_rezervare||' a fost din data de: '||rec\_rez.data\_check\_in);

END LOOP;

CLOSE c1;

END;

A close up of black text

Description automatically generated

--2.3. Sa se afiseze numele si salariul angajatilor din Hotelul cu ID-ul numarul 2 folosind un cursor:

SET SERVEROUTPUT ON

DECLARE

CURSOR c IS SELECT nume, salariul FROM p\_angajat WHERE id\_hotel = 2;

v\_nume p\_angajat.nume%TYPE;

v\_salariul p\_angajat.salariul%TYPE;

BEGIN

OPEN c;

LOOP

FETCH c INTO v\_nume, v\_salariul;

EXIT WHEN c%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Angajatul ' ||v\_nume|| ' are salariul ' ||v\_salariul);

END LOOP;

CLOSE c;

END;

/

A white background with black text

Description automatically generated

1. Funcţii, proceduri, includerea acestora în pachete:
2. Functii:

--1.1. Sa se creeze o functie care primeste id-ul hotelului si returneaza numarul total de angajati:

CREATE OR REPLACE FUNCTION CalculeazaNrAngajatiHotel (

p\_id\_hotel IN P\_Hotel.id\_hotel%TYPE

) RETURN NUMBER AS

v\_nr\_angajati NUMBER;

BEGIN

SELECT nr\_angajati INTO v\_nr\_angajati FROM P\_Hotel WHERE id\_hotel = p\_id\_hotel;

RETURN v\_nr\_angajati;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista hotelul cu id-ul specificat!');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

RETURN NULL;

END;

/



--testarea functiei calculeazaNrAngajatiHotel:

DECLARE

v\_nr\_angajati NUMBER;

BEGIN

v\_nr\_angajati := CalculeazaNrAngajatiHotel(2);

DBMS\_OUTPUT.PUT\_LINE('Numarul total de angajati din hotelul cerut este: ' || v\_nr\_angajati);

END;

/

A white background with black text

Description automatically generated

--1.2. Sa se creeze o functie care primeste id-ul clientului si returneaza numarul total de rezervari facute de acesta:

CREATE OR REPLACE FUNCTION CalculeazaNrRezervariClient (

p\_id\_client IN P\_Client.id\_client%TYPE

) RETURN NUMBER AS

v\_nr\_rezervari NUMBER := 0;

BEGIN

-- numararea rezervarilor facute de client, printr-un cursor

FOR rez IN (SELECT COUNT(\*) AS total\_rezervari FROM P\_Rezervari WHERE id\_client = p\_id\_client) LOOP

v\_nr\_rezervari := rez.total\_rezervari;

END LOOP;

RETURN v\_nr\_rezervari;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista rezervari pentru clientul specificat');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

RETURN NULL;

END;

/



--Testarea functiei calculeazaNrRezervariClient:

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Numarul total de rezervari pentru clientul cu id-ul 1 este: ' || CalculeazaNrRezervariClient(1));

END;

A close up of words

Description automatically generated

--1.3. Sa se creeze o funcție pentru calculul pretului total al unei rezervari:

CREATE OR REPLACE FUNCTION CalculPretTotalFunc(

in\_nr\_nopti IN NUMBER,

in\_pret\_noapte IN NUMBER

)

RETURN NUMBER

AS

v\_pret\_total NUMBER;

BEGIN

--validarea datelor de intrare

IF in\_nr\_nopti <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Numarul de nopti trebuie sa fie mai mare decat zero.');

END IF;

IF in\_pret\_noapte <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Pretul pe noapte trebuie sa fie mai mare decat zero.');

END IF;

v\_pret\_total := in\_nr\_nopti \* in\_pret\_noapte;

RETURN v\_pret\_total;

END CalculPretTotalFunc;

/



--testarea functiei calculPretTotalFct:

DECLARE

v\_nr\_nopti NUMBER := 5;

v\_pret\_noapte NUMBER := 150;

v\_pret\_total NUMBER;

BEGIN

v\_pret\_total := CalculPretTotalFunc(v\_nr\_nopti, v\_pret\_noapte);

DBMS\_OUTPUT.PUT\_LINE('Pretul total al rezervarii este: ' || v\_pret\_total);

END;

/

A white background with black text

Description automatically generated

1. Proceduri:

--2.1. Sa se creeze o procedura care primeste un id de Client si in functie de acesta arata numele acestuia:

CREATE OR REPLACE PROCEDURE AfiseazaNumeClient (

p\_id\_client IN P\_Client.id\_client%TYPE

) AS

v\_nume P\_Client.nume%TYPE;

v\_prenume P\_Client.prenume%TYPE;

BEGIN

--selectare nume si prenume client printr-un cursor

FOR c IN (SELECT nume, prenume FROM P\_Client WHERE id\_client = p\_id\_client) LOOP

v\_nume := c.nume;

v\_prenume := c.prenume;

DBMS\_OUTPUT.PUT\_LINE('Nume client: ' || v\_nume || ', prenume client: ' || v\_prenume);

END LOOP;

--cazul in care nu exista clientul cu id-ul specificat

IF v\_nume IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista clientul cu id-ul specificat!');

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END;

/



--testarea procedurii afiseazaNumeClient:

Execute AfiseazaNumeClient(1);

A close-up of words

Description automatically generated

--2.2. Sa se creeze o procedura care primeste un id de camera si afiseaza numele hotelului în care se afla camera și numarul total de obiecte de inventar

CREATE OR REPLACE PROCEDURE AfiseazaDetaliiCamera (

p\_id\_camera IN P\_Camera.id\_camera%TYPE

) AS

v\_nume\_hotel P\_Hotel.nume%TYPE;

v\_nr\_ob\_inventar NUMBER;

BEGIN

v\_nume\_hotel := NULL;

v\_nr\_ob\_inventar := 0;

-- obtinerea numelui hotelului in care se afla camera printr-un cursor

FOR hotel IN (SELECT h.nume FROM P\_Hotel h JOIN P\_Camera c ON h.id\_hotel = c.id\_hotel WHERE c.id\_camera = p\_id\_camera) LOOP

v\_nume\_hotel := hotel.nume;

END LOOP;

--numararea obiectelor de inventar din camera respectiva printr-un cursor

FOR obiecte IN (SELECT COUNT(\*) AS total\_ob\_inventar FROM P\_Obiecte\_inventar WHERE id\_camera = p\_id\_camera) LOOP

v\_nr\_ob\_inventar := obiecte.total\_ob\_inventar;

END LOOP;

IF v\_nume\_hotel IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Camera cu id-ul ' || p\_id\_camera || ' se afla in hotelul ' || v\_nume\_hotel || ' si contine ' || v\_nr\_ob\_inventar || ' obiecte de inventar.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu s-au găsit date pentru camera specificata.');

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

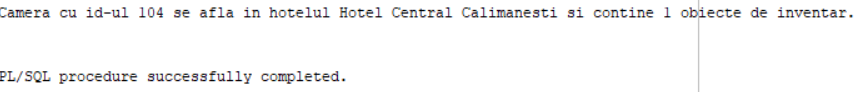
END;

/



--Testare procedurii afiseazaDetaliiCamera

EXECUTE AfiseazaDetaliiCamera(104);



--2.3. Sa se creeze o procedura pentru calculul pretului total al unei rezervari:

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE CalculPretTotal(

in\_id\_rezervare IN INT,

out\_pret\_total OUT NUMBER

)

AS

v\_nr\_nopti NUMBER;

v\_pret\_noapte NUMBER;

BEGIN

--obtinerea datelor necesare folosind cursorul

FOR c\_rezervare IN (SELECT data\_check\_in, data\_check\_out FROM P\_Rezervari WHERE id\_rezervare = in\_id\_rezervare) LOOP

v\_nr\_nopti := c\_rezervare.data\_check\_out - c\_rezervare.data\_check\_in;

END LOOP;

--gestionarea cazurilor cand nu se gaseste rezervarea respectiva sau datele suntr introduse incorect

IF v\_nr\_nopti IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Rezervarea nu a fost gasită sau datele sunt incorecte!');

END IF;

--obtinerea pretului pe noapte printr-un cursor implicit

SELECT pret INTO v\_pret\_noapte

FROM P\_Camera

WHERE id\_camera = (SELECT id\_camera FROM P\_Rezervari WHERE id\_rezervare = in\_id\_rezervare);

IF v\_pret\_noapte IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Pretul camerei nu a fost gasit sau datele sunt incorecte!');

END IF;

out\_pret\_total := v\_nr\_nopti \* v\_pret\_noapte;

END CalculPretTotal;

/



--Testarea procedurii calculPretTotal :

DECLARE

v\_id\_rezervare INT := 1111;

v\_pret\_total NUMBER;

BEGIN

CalculPretTotal(v\_id\_rezervare, v\_pret\_total);

DBMS\_OUTPUT.PUT\_LINE('Pretul total al rezervarii cu ID-ul ' || v\_id\_rezervare || ' este: ' || v\_pret\_total);

END;

/

A close up of words

Description automatically generated

1. Pachete:

--3.1. Sa se creeze un pachet care sa aiba in compozitia sa functia CalculeazaNrAngajatiHotel si procedura AfiseazaNumeClient:

CREATE OR REPLACE PACKAGE HotelPackage AS

FUNCTION CalculeazaNrAngajatiHotel (

p\_id\_hotel IN P\_Hotel.id\_hotel%TYPE

) RETURN NUMBER;

PROCEDURE AfiseazaNumeClient (

p\_id\_client IN P\_Client.id\_client%TYPE

);

END HotelPackage;

/



CREATE OR REPLACE PACKAGE BODY HotelPackage AS

-- Implementarea funcției CalculeazaNrAngajatiHotel

FUNCTION CalculeazaNrAngajatiHotel (

p\_id\_hotel IN P\_Hotel.id\_hotel%TYPE

) RETURN NUMBER AS

v\_nr\_angajati NUMBER;

BEGIN

SELECT nr\_angajati INTO v\_nr\_angajati FROM P\_Hotel WHERE id\_hotel = p\_id\_hotel;

RETURN v\_nr\_angajati;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista hotelul cu id-ul specificat!');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

RETURN NULL;

END CalculeazaNrAngajatiHotel;

-- Implementarea procedurii AfiseazaNumeClient

PROCEDURE AfiseazaNumeClient (

p\_id\_client IN P\_Client.id\_client%TYPE

) AS

v\_nume P\_Client.nume%TYPE;

v\_prenume P\_Client.prenume%TYPE;

BEGIN

-- Selectare nume și prenume client printr-un cursor

FOR c IN (SELECT nume, prenume FROM P\_Client WHERE id\_client = p\_id\_client) LOOP

v\_nume := c.nume;

v\_prenume := c.prenume;

DBMS\_OUTPUT.PUT\_LINE('Nume client: ' || v\_nume || ', prenume client: ' || v\_prenume);

END LOOP;

-- Cazul în care nu există clientul cu id-ul specificat

IF v\_nume IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu exista clientul cu id-ul specificat!');

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END AfiseazaNumeClient;

END HotelPackage;

/



--testare:

DECLARE

v\_nr\_angajati NUMBER;

v\_id\_hotel P\_Hotel.id\_hotel%TYPE := 1;

v\_id\_client P\_Client.id\_client%TYPE := 9;

BEGIN

--Testam functia CalculeazaNrAngajatiHotel

v\_nr\_angajati := HOTELPACKAGE.CalculeazaNrAngajatiHotel(v\_id\_hotel);

IF v\_nr\_angajati IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Numarul de angajati pentru hotelul ' || v\_id\_hotel || ' este: ' || v\_nr\_angajati);

END IF;

--Testam procedura AfiseazaNumeClient

DBMS\_OUTPUT.PUT\_LINE('Numele si prenumele clientului cu id-ul ' || v\_id\_client || ' sunt:');

HOTELPACKAGE.AfiseazaNumeClient(v\_id\_client);

END;

/

A close up of a text

Description automatically generated

--3.2. Sa se creeze un pachet care sa continua functia CalculeazaNrRezervariClient si procedura CalculPretTotal:

CREATE OR REPLACE PACKAGE RezervariPackage AS

FUNCTION CalculeazaNrRezervariClient (

p\_id\_client IN P\_Client.id\_client%TYPE

) RETURN NUMBER;

PROCEDURE CalculPretTotal(

in\_id\_rezervare IN INT,

out\_pret\_total OUT NUMBER

);

END RezervariPackage;

/

A close-up of a package

Description automatically generated

CREATE OR REPLACE PACKAGE BODY RezervariPackage AS

FUNCTION CalculeazaNrRezervariClient (

p\_id\_client IN P\_Client.id\_client%TYPE

) RETURN NUMBER AS

v\_nr\_rezervari NUMBER := 0;

BEGIN

-- numararea rezervarilor facute de client, printr-un cursor

SELECT COUNT(\*) INTO v\_nr\_rezervari FROM P\_Rezervari WHERE id\_client = p\_id\_client;

RETURN v\_nr\_rezervari;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista rezervari pentru clientul specificat');

RETURN NULL;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

RETURN NULL;

END CalculeazaNrRezervariClient;

PROCEDURE CalculPretTotal(

in\_id\_rezervare IN INT,

out\_pret\_total OUT NUMBER

) AS

v\_nr\_nopti NUMBER;

v\_pret\_noapte NUMBER;

BEGIN

--obtinerea datelor necesare folosind cursorul

SELECT data\_check\_out - data\_check\_in INTO v\_nr\_nopti

FROM P\_Rezervari

WHERE id\_rezervare = in\_id\_rezervare;

--gestionarea cazurilor cand nu se gaseste rezervarea respectiva sau datele suntr introduse incorect

IF v\_nr\_nopti IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Rezervarea nu a fost gasită sau datele sunt incorecte!');

END IF;

--obtinerea pretului pe noapte printr-un cursor implicit

SELECT pret INTO v\_pret\_noapte

FROM P\_Camera

WHERE id\_camera = (SELECT id\_camera FROM P\_Rezervari WHERE id\_rezervare = in\_id\_rezervare);

IF v\_pret\_noapte IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Pretul camerei nu a fost gasit sau datele sunt incorecte!');

END IF;

out\_pret\_total := v\_nr\_nopti \* v\_pret\_noapte;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare: ' || SQLERRM);

END CalculPretTotal;

END RezervariPackage;

/



--testare:

DECLARE

v\_nr\_rezervari NUMBER;

v\_pret\_total NUMBER;

v\_id\_client P\_Client.id\_client%TYPE := 1;

v\_id\_rezervare P\_Rezervari.id\_rezervare%TYPE := 1111;

BEGIN

v\_nr\_rezervari := RezervariPackage.CalculeazaNrRezervariClient(v\_id\_client);

IF v\_nr\_rezervari IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Numarul de rezervari ale clientului ' || v\_id\_client || ' este: ' || v\_nr\_rezervari);

END IF;

RezervariPackage.CalculPretTotal(v\_id\_rezervare, v\_pret\_total);

IF v\_pret\_total IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Pretul total pentru rezervarea ' || v\_id\_rezervare || ' este: ' || v\_pret\_total);

END IF;

END;

/

A white background with black text

Description automatically generated

1. Declansatori:

-- 1.Declansator pt actualizarea pretului unei camere din tabelul P\_Camera

--care actualizeaza pretul camerei din tabelul P\_Camera atunci cand se adauga sau se elimina un obiect din inventarul camerei

SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER tr\_update\_pret\_camera

BEFORE INSERT OR DELETE ON P\_Obiecte\_inventar

FOR EACH ROW

DECLARE

v\_camera\_id P\_Obiecte\_inventar.id\_camera%TYPE;

v\_new\_price P\_Camera.pret%TYPE;

BEGIN

IF INSERTING THEN

--ID-ul camerei

v\_camera\_id := :NEW.id\_camera;

--calc noul pret al camerei

SELECT count(nr\_bucati) INTO v\_new\_price

FROM P\_Obiecte\_inventar oi

JOIN P\_Camera c ON oi.id\_camera = c.id\_camera

WHERE c.id\_camera = v\_camera\_id;

v\_new\_price := 20\*v\_new\_price;

--actualizez pretul

UPDATE P\_Camera

SET pret = pret + v\_new\_price

WHERE id\_camera = v\_camera\_id;

ELSIF DELETING THEN

--obt id-ul camerei

v\_camera\_id := :OLD.id\_camera;

--calc noul pret al camerei

SELECT count(nr\_bucati) INTO v\_new\_price

FROM P\_Obiecte\_inventar oi

JOIN P\_Camera c ON oi.id\_camera = c.id\_camera

WHERE c.id\_camera = v\_camera\_id;

v\_new\_price := v\_new\_price\*20;

--actualizez pretul

UPDATE P\_Camera

SET pret = pret - v\_new\_price

WHERE id\_camera = v\_camera\_id;

END IF;

END;

/



--testez declansatorul, se poate observa pretul initial al camerelor cu id-ul 1325, 1522

A screenshot of a computer

Description automatically generated

--se adauga obiecte de inventar in camerele respective

insert into p\_obiecte\_inventar

values(401,'scaun',4,1522);

insert into p\_obiecte\_inventar

values(402,'pat',4,1325);

--aici se pot observa preturile care sunt modificate dupa adaugarea unor obiecte de inventar

select \* from p\_camera;

A screenshot of a computer

Description automatically generated

Sa se creeze un declansator care atunci cand se adauga un angajat in p\_angajat sa se mareasca nr de angajati din tabela p\_hotel

CREATE OR REPLACE TRIGGER TR\_INCREMENT\_EMPLOYEE\_COUNT

AFTER INSERT ON P\_Angajat

FOR EACH ROW

BEGIN

-- Incrementăm numărul de angajați din hotelul în care a fost adăugat noul angajat

UPDATE P\_Hotel

SET nr\_angajati = nr\_angajati + 1

WHERE id\_hotel = :NEW.id\_hotel;

END;

/



--testam declansatorul inainte de actualizare, vedem cati angajati erau in hotelul cu id-ul 1:

SELECT nr\_angajati FROM P\_Hotel WHERE id\_hotel = 1;

A white rectangular sign with black text

Description automatically generated

--inserez un nou angajat in p\_angajat

INSERT INTO P\_Angajat(id\_angajat, nume, prenume, telefon, email, data\_ang, salariul, domiciliu\_angajat, id\_hotel)

VALUES(1131, 'Mircea', 'Ion', '0751244447', 'mirceaion@gmail.com', TO\_DATE('2023-03-15', 'YYYY-MM-DD'), 3100, 'Strada Pacii, nr. 15', 1);



--acum putem vedea ca nr angajatiilor s-a actualizat si s-a crescut cu 1:

SELECT nr\_angajati FROM P\_Hotel WHERE id\_hotel = 1;

A close up of a sign

Description automatically generated

--3. Sa se actualizeze salariul angajatului cu cea mai veche data de angajare din acelasi hotel la valoarea salariului mediu al angajaților din acel hotel:

CREATE OR REPLACE TRIGGER tr\_update\_salary\_oldest\_employee

INSTEAD OF INSERT OR DELETE OR UPDATE ON ang\_view

FOR EACH ROW

DECLARE

v\_avg\_salary P\_Angajat.salariul%TYPE;

BEGIN

IF DELETING THEN

DELETE FROM p\_angajat

WHERE id\_angajat = :new.id\_angajat;

ELSIF INSERTING THEN

INSERT INTO p\_angajat

values(:new.id\_angajat,:new.nume,:new.prenume,:new.telefon,:new.email,:new.data\_ang,:new.salariul,:new.domiciliu\_angajat,:new.id\_hotel);

ELSE

UPDATE p\_angajat

set salariul = :new.salariul

WHERE id\_angajat = :new.id\_angajat;

END IF;

-- calculez salariul mediu al ang din acelasi hotel

SELECT AVG(salariul) INTO v\_avg\_salary

FROM P\_Angajat

WHERE id\_hotel = :NEW.id\_hotel;

--actualizez salariul angajatului cu cea mai veche data de angajare

UPDATE P\_Angajat

SET salariul = v\_avg\_salary

WHERE id\_angajat = (

SELECT id\_angajat

FROM (

SELECT id\_angajat

FROM P\_Angajat

WHERE id\_hotel = :NEW.id\_hotel

ORDER BY data\_ang ASC

)

WHERE ROWNUM = 1

) AND id\_hotel = :NEW.id\_hotel;

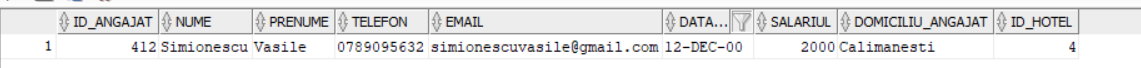
END;

/



--testez declansatorul

--se poate vedea salariul initial al celui mai vechi angajat



--aici se poate vedea salariul actualizat

insert into ang\_view(id\_angajat,id\_hotel)

values(2005,4);

