PROIECT BAZE DE DATE

CEURANU LOREDANA

A yellow box with black text

Description automatically generated with medium confidence

Proiectul este despre proiectarea și implementarea unei baze de date pentru gestiunea activității unei pensiuni.

Se consideră activitatea de evidență a funcționării unei pensiuni.

Descrierea proiectului:

Relațiile între aceste entități sunt esențiale pentru funcționalitatea sistemului nostru. În primul rând, rezervările sunt conectate cu camerele, clienții și plățile. Aceasta înseamnă că fiecare rezervare este asociată cu o cameră specifică, un client și o plată corespunzătoare. Astfel, putem urmări cu exactitate detaliile rezervărilor și să asigurăm o eficientă în gestionarea plăților.

De asemenea, avem și relația între plățile pentru servicii extra și rezervări. Aceasta înseamnă că fiecare plată pentru servicii extra este legată de o rezervare specifică și un serviciu extra oferit în cadrul acelei rezervări. Acest lucru ne permite să monitorizăm și să gestioneze cu precizie plata pentru serviciile extra oferite în timpul rezervării.

În tabela 'REZERVĂRI', avem câteva coloane importante precum: 'id\_rezervare' pentru identificarea unică a rezervării, 'client' pentru conectarea cu entitatea 'Clienți', 'camera' pentru asocierea rezervării cu o cameră specifică, 'datainceput' și 'datasfarsit' pentru a înregistra intervalul de timp al rezervării și 'status' pentru a indica dacă rezervarea este 'confirmată' sau nu.

Această bază de date ERD ne permite să gestionăm eficient rezervările, să urmărim detaliile specifice ale clienților, camerelor, plăților și serviciilor extra, precum și să oferim un sistem solid de înregistrare și monitorizare a tuturor aspectelor legate de rezervări în domeniul hospitalității.

Prin utilizarea acestei baze de date și a relațiilor între entități, suntem capabili să oferim un sistem robust și eficient de gestionare a rezervărilor, care va contribui la creșterea eficienței și a satisfacției clienților noștri.

1. **Implementați în PL/SQL un sistem pentru verificarea disponibilității unei camere de pensiune în intervalul 15-20 aprilie 2023. Sistemul va solicita ID-ul camerei și va folosi un cursor pentru a căuta în rezervările existente orice suprapunere de date care ar indica indisponibilitatea camerei în acest interval. La final, trebuie afișat un mesaj corespunzător disponibilității.**

\*cursor explicit

DECLARE

v\_id\_camera INT;

v\_data\_inceput DATE := TO\_DATE('2023-04-15', 'YYYY-MM-DD');

v\_data\_sfarsit DATE := TO\_DATE('2023-04-20', 'YYYY-MM-DD');

v\_camera\_disponibila BOOLEAN := TRUE;

CURSOR c\_rezervari\_camera IS

SELECT ID\_Rezervare

FROM REZERVARI

WHERE ID\_Camera = v\_id\_camera

AND (DataInceput BETWEEN v\_data\_inceput AND v\_data\_sfarsit OR

DataSfarsit BETWEEN v\_data\_inceput AND v\_data\_sfarsit);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Introduceti ID-ul camerei:');

v\_id\_camera := &v\_id\_camera;

OPEN c\_rezervari\_camera;

FETCH c\_rezervari\_camera INTO v\_id\_camera;

IF c\_rezervari\_camera%FOUND THEN

v\_camera\_disponibila := FALSE;

DBMS\_OUTPUT.PUT\_LINE('Camera nu este disponibila in intervalul specificat.');

ELSE

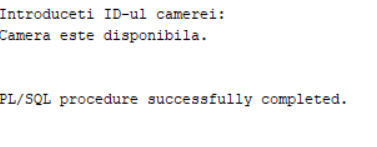
DBMS\_OUTPUT.PUT\_LINE('Camera este disponibila.');

END IF;

CLOSE c\_rezervari\_camera;

END;

/



2. **Realizați un bloc PL/SQL care solicită introducerea de la tastatură a ID-ului unui client. Apoi, blocul va selecta numele și prenumele clientului cu ID-ul respectiv din baza de date și va afișa lista rezervărilor și plăților asociate acelui client, inclusiv detaliile specifice pentru fiecare înregistrare. În plus, blocul trebuie să gestioneze situația în care nu există date disponibile pentru clientul introdus.**

DECLARE

v\_id\_client CLIENTI.ID\_Client%TYPE;

v\_nume CLIENTI.Nume%TYPE;

v\_prenume CLIENTI.Prenume%TYPE;

CURSOR client\_rezervari\_plati\_cursor IS

SELECT r.ID\_Rezervare, r.DataInceput, r.DataSfarsit, p.Suma

FROM REZERVARI r

JOIN PLATI p ON r.ID\_Rezervare = p.ID\_Rezervare

WHERE r.ID\_Client = v\_id\_client;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Introduceti ID-ul clientului pentru a afisa informatiile despre rezervari si plati:');

v\_id\_client := &id\_client;

SELECT Nume, Prenume INTO v\_nume, v\_prenume

FROM CLIENTI

WHERE ID\_Client = v\_id\_client;

DBMS\_OUTPUT.PUT\_LINE('Lista cu rezervarile si platile pentru clientul ' || v\_nume || ' ' || v\_prenume || ':');

FOR rec IN client\_rezervari\_plati\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Rezervarea ' || rec.ID\_Rezervare ||

' incepe in data de ' || TO\_CHAR(rec.DataInceput, 'dd-mm-yyyy') ||

' si se termina in data de ' || TO\_CHAR(rec.DataSfarsit, 'dd-mm-yyyy') ||

'. Suma platita: ' || rec.Suma || '.');

END LOOP;

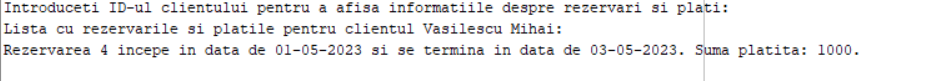
EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu au fost gasite rezervari sau plati pentru clientul cu ID-ul introdus.');

END;

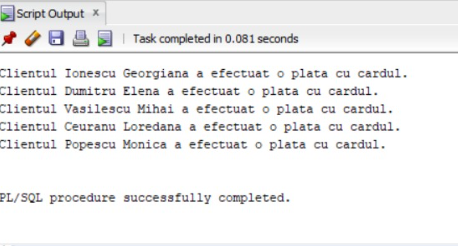
/



A close-up of a white background

Description automatically generated

3. **Creați un bloc PL/SQL care utilizează un cursor pentru a selecta și itera prin lista clienților care au efectuat plăți cu cardul**.



DECLARE

CURSOR c\_plati\_card IS

SELECT c.Nume, c.Prenume

FROM CLIENTI c

INNER JOIN REZERVARI r ON c.ID\_Client = r.ID\_Client

INNER JOIN PLATI p ON r.ID\_Rezervare = p.ID\_Rezervare

WHERE p.Modalitate = 'Card';

BEGIN

FOR client IN c\_plati\_card LOOP

DBMS\_OUTPUT.PUT\_LINE('Clientul ' || client.Nume || ' ' || client.Prenume || ' a efectuat o plata cu cardul.');

END LOOP;

END;

/

4.

**Într-un bloc PL/SQL actualizați plata unei rezervări al cărei ID îl citim de la tastatură, în funcție de următoarele cerințe: dacă suma plătită este mai mare de 500, se aplică o reducere de 5%, iar dacă suma plătită este mai mare de 1000, se aplică o reducere de 10%. În cazul în care suma plătită este sub 500, plata rămâne aceeași. Blocul PL/SQL trebuie să afișeze ID-ul și suma inițială, să efectueze actualizarea și apoi să afișeze noua sumă și ID-ul rezervării corespunzătoare.**

DECLARE

v\_id\_rezervare NUMBER;

v\_suma\_initiala NUMBER;

v\_noua\_suma NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Introduceti ID-ul rezervarii pentru actualizare:');

v\_id\_rezervare := &id\_rezervare;

SELECT Suma INTO v\_suma\_initiala FROM PLATI WHERE ID\_Rezervare = v\_id\_rezervare;

IF v\_suma\_initiala > 1000 THEN

v\_noua\_suma := v\_suma\_initiala - (v\_suma\_initiala \* 0.10);

ELSIF v\_suma\_initiala > 500 THEN

v\_noua\_suma := v\_suma\_initiala - (v\_suma\_initiala \* 0.05);

ELSE

v\_noua\_suma := v\_suma\_initiala;

END IF;

UPDATE PLATI SET Suma = v\_noua\_suma WHERE ID\_Rezervare = v\_id\_rezervare;

DBMS\_OUTPUT.PUT\_LINE('ID-ul rezervarii: ' || v\_id\_rezervare);

DBMS\_OUTPUT.PUT\_LINE('Suma initiala: ' || v\_suma\_initiala);

DBMS\_OUTPUT.PUT\_LINE('Noua suma: ' || v\_noua\_suma);

END;

/

A close up of words

Description automatically generated

5**. Într-un bloc PL/SQL să se parcurgă toate camerele, afișand pretul, numarul și tipul acestora**

DECLARE

CURSOR camere\_cursor IS SELECT PretNoapte, NumarCamera, Tip FROM CAMERE;

v\_pretNoapte CAMERE.PretNoapte%TYPE;

v\_numarCamera CAMERE.NumarCamera%TYPE;

v\_tip CAMERE.Tip%TYPE;

BEGIN

OPEN camere\_cursor;

FETCH camere\_cursor INTO v\_pretNoapte, v\_numarCamera, v\_tip;

WHILE camere\_cursor%FOUND LOOP

DBMS\_OUTPUT.PUT\_LINE('Numar: ' || v\_numarCamera || ', Tip: ' || v\_tip || ', Pret: ' || v\_pretNoapte);

FETCH camere\_cursor INTO v\_pretNoapte, v\_numarCamera, v\_tip;

END LOOP;

CLOSE camere\_cursor;

END;

/

A screenshot of a computer

Description automatically generated

EXCEPTII

1. Actualizează suma unei rezervări în sistemul hotelier, scăzând 0.1 unități monetare. Dacă ID-ul rezervării specificat nu există, sistemul trebuie să notifice că reducerea nu a fost aplicată. Folosește excepții PL/SQL pentru gestionarea succesului sau eșecului acestei operațiuni.

SET SERVEROUTPUT ON

DECLARE plata\_exception EXCEPTION;

BEGIN

UPDATE PLATI

SET SUMA=SUMA-0.1

WHERE id\_rezervare=&id;

IF sql%notfound then

raise plata\_exception;

else dbms\_output.put\_line('S-a aplicat reducerea');

end if;

EXCEPTION

WHEN plata\_exception THEN

dbms\_output.put\_line('NU S-A APLICAT NICIO REDUCERE');

END;

/

A close-up of a computer screen

Description automatically generated

A white background with black text

Description automatically generated

2. Sa se modifice pretul camerei cu id-ul 11, Dacă nu se produce nici o actualizare (valoarea atributului SQL%ROWCOUNT este 0) sau dacă apare o altă excepție (clauza OTHERS) atunci să se declanşeze o excepţie prin care să fie avertizat utilizatorul.

DECLARE

invalid\_camera EXCEPTION;

BEGIN

UPDATE CAMERE

SET PretNoapte = 450

WHERE ID\_Camera = 11;

IF SQL%NOTFOUND THEN

RAISE invalid\_camera;

END IF;

EXCEPTION

WHEN invalid\_camera THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista camera cu acest ID');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare! Nu se poate actualiza pretul camerei!');

END;



1. Caută un client în baza de date după numele "Doe". Dacă nu găsește niciun client cu acest nume, este generată o excepție specifică. Dacă nu sunt obținute date din cursor, afișează un mesaj corespunzător. În cazul apariției altor erori, se afișează un mesaj general de eroare.

DECLARE

CURSOR curs\_client IS SELECT ID\_Client FROM CLIENTI WHERE Nume = 'Doe';

id\_client\_negasit EXCEPTION;

id\_client\_val INT;

BEGIN

OPEN curs\_client;

FETCH curs\_client INTO id\_client\_val;

IF curs\_client%NOTFOUND THEN

RAISE id\_client\_negasit;

END IF;

CLOSE curs\_client;

EXCEPTION

WHEN id\_client\_negasit THEN

dbms\_output.put\_line('Exceptie: Clientul cu numele Doe nu a fost gasit.');

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('Niciun rand nu corespunde criteriului.');

WHEN OTHERS THEN

dbms\_output.put\_line('A aparut o alta eroare!');

END;

A white background with black text

Description automatically generated

Verifică valoarea de preț a unei camere în cadrul sistemului de gestionare a hotelului. Dacă valoarea camerei depășește 800 unități monetare, sistemul va genera o excepție pentru a indica că prețul camerei este mai mare decât limita admisă. Implementează această verificare folosind un bloc PL/SQL. Dacă valoarea camerei depășește 800, afișează un mesaj corespunzător. Acest proces permite aplicarea unor reguli și restricții referitoare la prețurile camerelor pentru a asigura coerența datelor și conformitatea cu standardele stabilite.

DECLARE

valoare\_invalida EXCEPTION;

pret\_camera DECIMAL(10,2) := 1000;

BEGIN

IF pret\_camera > 800 THEN

RAISE valoare\_invalida;

END IF;

EXCEPTION

WHEN valoare\_invalida THEN

dbms\_output.put\_line('Excepție: Valoarea camerei depășește limita admisă.');

END;

A screenshot of a computer

Description automatically generated

**Cursori Expliciți**

**Cursor Explicii fără Parametri**

DECLARE

CURSOR c\_camere IS

SELECT \* FROM CAMERE WHERE Etaj = 1;

v\_camere CAMERE%ROWTYPE;

BEGIN

OPEN c\_camere;

LOOP

FETCH c\_camere INTO v\_camere;

EXIT WHEN c\_camere%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Camera ' || v\_camere.NumarCamera || ' este la etajul ' || v\_camere.Etaj);

END LOOP;

CLOSE c\_camere;

END;

A white background with black text

Description automatically generated

**Cursor Expliciti cu Parametri**

DECLARE

v\_status\_rezervare VARCHAR2(50) := 'Confirmat'; -- Parametrul despre care efectuăm interogarea

CURSOR c\_rezervari(status\_rezervare VARCHAR2) IS

SELECT ID\_Rezervare, ID\_Client, ID\_Camera, DataInceput, DataSfarsit

FROM REZERVARI

WHERE Status = status\_rezervare;

v\_rezervare c\_rezervari%ROWTYPE;

BEGIN

OPEN c\_rezervari(v\_status\_rezervare);

LOOP

FETCH c\_rezervari INTO v\_rezervare;

EXIT WHEN c\_rezervari%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Rezervare ID: ' || v\_rezervare.ID\_Rezervare ||

', Client ID: ' || v\_rezervare.ID\_Client ||

', Camera ID: ' || v\_rezervare.ID\_Camera ||

', Incepe: ' || TO\_CHAR(v\_rezervare.DataInceput, 'DD-MM-YYYY') ||

', Se termina : ' || TO\_CHAR(v\_rezervare.DataSfarsit, 'DD-MM-YYYY'));

END LOOP;

CLOSE c\_rezervari;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare întâlnită: ' || SQLERRM);

END;

A table with numbers and letters

Description automatically generated with medium confidence

**Cursor Implicii Fără Parametri**

BEGIN

FOR v\_client IN (SELECT \* FROM CLIENTI WHERE Prenume LIKE 'A%')

LOOP

DBMS\_OUTPUT.PUT\_LINE('Client: ' || v\_client.Nume || ' ' || v\_client.Prenume);

END LOOP;

END;

A screenshot of a computer code

Description automatically generated

**Cursor Implicit cu Parametri**

DECLARE

v\_tip VARCHAR2(50) := 'Dubla Lux';

BEGIN

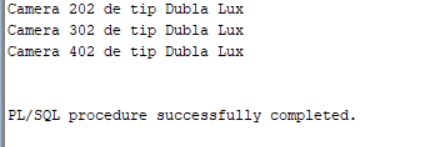
FOR v\_camera IN (SELECT \* FROM CAMERE WHERE Tip = v\_tip)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Camera ' || v\_camera.NumarCamera || ' de tip ' || v\_camera.Tip);

END LOOP;

END;



BEGIN

FOR v\_rezervare IN (SELECT r.ID\_Rezervare, r.DataInceput, c.NumarCamera, c.Tip

FROM REZERVARI r JOIN CAMERE c ON r.ID\_Camera = c.ID\_Camera

WHERE r.Status = 'Confirmat' AND c.Etaj = 2)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Rezervare ' || v\_rezervare.ID\_Rezervare || ' pentru camera ' ||

v\_rezervare.NumarCamera || ' (' || v\_rezervare.Tip ||

') începe la ' || TO\_CHAR(v\_rezervare.DataInceput, 'DD-MM-YYYY'));

END LOOP;

END;

A close-up of a text

Description automatically generated

Funcţii, proceduri, includerea acestora în pachete (minim 3 funcţii, 3 proceduri şi 2 pachete)

**Funcție 1: Calcularea prețului total pe baza prețului per noapte și numărului de nopți**

CREATE OR REPLACE FUNCTION CalculPretTotal(pret\_noapte IN NUMBER, nr\_nopti IN NUMBER) RETURN NUMBER IS

pret\_total NUMBER;

BEGIN

pret\_total := pret\_noapte \* nr\_nopti;

RETURN pret\_total;

END;

/

--apel:

DECLARE

v\_pret\_noapte NUMBER := 100; -- Prețul pe noapte

v\_nr\_nopti NUMBER := 3; -- Numărul de nopți

v\_pret\_total NUMBER; -- Variabilă pentru stocarea prețului total calculat

BEGIN

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

DBMS\_OUTPUT.PUT\_LINE('Pretul total pentru ' || v\_nr\_nopti || ' nopti la pretul de ' ||

v\_pret\_noapte || ' pe noapte este: ' || v\_pret\_total);

END;

/

A white background with black text

Description automatically generated

**Funcție 2: Verificarea dacă un client este fidel în funcție de numărul rezervărilor**

CREATE OR REPLACE FUNCTION VerificaClientFidel(numar\_rezervari IN NUMBER) RETURN VARCHAR2 IS

BEGIN

IF numar\_rezervari > 5 THEN

RETURN 'Clientul este fidel';

ELSE

RETURN 'Clientul nu este fidel';

END IF;

END;

/

--apel

A close-up of words

Description automatically generated

**Funcție 3: Extrage numele complet al unui client**

CREATE OR REPLACE FUNCTION GetNumeComplet(id\_client IN NUMBER) RETURN VARCHAR2 IS

nume\_complet VARCHAR2(255);

BEGIN

SELECT Nume || ' ' || Prenume INTO nume\_complet

FROM CLIENTI WHERE ID\_Client = id\_client AND ROWNUM = 1;

RETURN nume\_complet;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Numele clientului nu a putut fi găsit.';

END;

/

--apel

DECLARE

v\_id\_client NUMBER := 1; -- ID-ul clientului pentru care obținem numele complet

v\_nume\_complet VARCHAR2(255); -- Variabilă pentru stocarea numelui complet

BEGIN

v\_nume\_complet := GetNumeComplet(v\_id\_client); -- Apelarea funcției pentru a obține numele complet

DBMS\_OUTPUT.PUT\_LINE('Numele complet al clientului cu ID ' || v\_id\_client || ' este: ' || COALESCE(v\_nume\_complet, 'Numele nu a putut fi găsit.'));

-- Afișarea numelui complet sau un mesaj corespunzător dacă numele nu a fost găsit

END;

/

A white background with black text

Description automatically generated

**Procedura 1: Actualizarea stării unei rezervări la finalizat**

CREATE OR REPLACE PROCEDURE FinalizeazaRezervare(id\_rezervare IN NUMBER) IS

BEGIN

UPDATE REZERVARI SET Status = 'Finalizat' WHERE ID\_Rezervare = id\_rezervare;

COMMIT;

END;

/

--Apel

DECLARE

v\_id\_rezervare NUMBER := 1; -- ID-ul rezervării pe care dorim să o finalizăm

BEGIN

FinalizeazaRezervare(v\_id\_rezervare); -- Apelarea procedurii pentru a finaliza rezervarea specificată

DBMS\_OUTPUT.PUT\_LINE('Rezervarea cu ID ' || v\_id\_rezervare || ' a fost finalizata cu succes.'); -- Afișarea unui mesaj de confirmare

END;

A close-up of a computer screen

Description automatically generated

**Procedura 2: Anularea unei rezervări**

CREATE OR REPLACE PROCEDURE AnuleazaRezervare(id\_rezervare IN NUMBER) IS

BEGIN

DELETE FROM REZERVARI WHERE ID\_Rezervare = id\_rezervare;

COMMIT;

END;

/



--apel

DECLARE

v\_id\_rezervare NUMBER := 5; -- ID-ul rezervării pe care dorim să o finalizăm

BEGIN

FinalizeazaRezervare(v\_id\_rezervare); -- Apelarea procedurii pentru a finaliza rezervarea specificată

DBMS\_OUTPUT.PUT\_LINE('Rezervarea cu ID ' || v\_id\_rezervare || ' a fost finalizata cu succes.'); -- Afișarea unui mesaj de confirmare

END;

/

A close-up of a white background

Description automatically generated

CREATE OR REPLACE PROCEDURE VerificaDisponibilitateCamera1(id\_camera IN NUMBER, data\_inceput IN DATE, data\_sfarsit IN DATE) IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count

FROM REZERVARI

WHERE ID\_Camera = id\_camera

AND ((data\_inceput BETWEEN DataInceput AND DataSfarsit)

OR (data\_sfarsit BETWEEN DataInceput AND DataSfarsit));

IF v\_count > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Camera nu este disponibila in acea perioada.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Camera este disponibilă în acea perioadă.');

END IF;

END;

/

--Apel

DECLARE

v\_id\_camera NUMBER := 1; -- ID-ul camerei pentru verificare

v\_data\_inceput DATE := TO\_DATE('2023-05-01', 'YYYY-MM-DD'); -- Data de început a perioadei de verificare

v\_data\_sfarsit DATE := TO\_DATE('2023-05-05', 'YYYY-MM-DD'); -- Data de sfârșit a perioadei de verificare

BEGIN

VerificaDisponibilitateCamera1(v\_id\_camera, v\_data\_inceput, v\_data\_sfarsit); -- Apelarea procedurii pentru a verifica disponibilitatea camerei

DBMS\_OUTPUT.PUT\_LINE('Verificarea disponibilitatii camerei pentru ID ' || v\_id\_camera); -- Afișarea rezultatului verificării

END;

/

A close up of a screen

Description automatically generated

PACHETE

-- Setarea SERVEROUTPUT la ON

SET SERVEROUTPUT ON;

-- Specificația pachetului

CREATE OR REPLACE PACKAGE actualizare\_clienti IS

PROCEDURE adauga\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE,

p\_nume CLIENTI.Nume%TYPE,

p\_prenume CLIENTI.Prenume%TYPE,

p\_email CLIENTI.Email%TYPE,

p\_telefon CLIENTI.Telefon%TYPE

);

PROCEDURE modifica\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE,

p\_nume CLIENTI.Nume%TYPE,

p\_prenume CLIENTI.Prenume%TYPE,

p\_email CLIENTI.Email%TYPE,

p\_telefon CLIENTI.Telefon%TYPE

);

PROCEDURE modifica\_client\_email(

p\_id\_client CLIENTI.ID\_Client%TYPE,

p\_email CLIENTI.Email%TYPE

);

PROCEDURE sterge\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE

);

FUNCTION exista\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE

) RETURN BOOLEAN;

exceptie\_client EXCEPTION;

END actualizare\_clienti;

/

-- Corpul pachetului

CREATE OR REPLACE PACKAGE BODY actualizare\_clienti IS

PROCEDURE adauga\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE,

p\_nume CLIENTI.Nume%TYPE,

p\_prenume CLIENTI.Prenume%TYPE,

p\_email CLIENTI.Email%TYPE,

p\_telefon CLIENTI.Telefon%TYPE

) IS

BEGIN

IF exista\_client(p\_id\_client) THEN

RAISE exceptie\_client;

ELSE

INSERT INTO CLIENTI (ID\_Client, Nume, Prenume, Email, Telefon)

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

END IF;

EXCEPTION

WHEN exceptie\_client THEN

DBMS\_OUTPUT.PUT\_LINE('Client existent!');

END adauga\_client;

PROCEDURE modifica\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE,

p\_nume CLIENTI.Nume%TYPE,

p\_prenume CLIENTI.Prenume%TYPE,

p\_email CLIENTI.Email%TYPE,

p\_telefon CLIENTI.Telefon%TYPE

) IS

BEGIN

IF exista\_client(p\_id\_client) THEN

UPDATE CLIENTI

SET Nume = p\_nume, Prenume = p\_prenume, Email = p\_email, Telefon = p\_telefon

WHERE ID\_Client = p\_id\_client;

ELSE

RAISE exceptie\_client;

END IF;

EXCEPTION

WHEN exceptie\_client THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu acest ID nu exista!');

END modifica\_client;

PROCEDURE modifica\_client\_email(

p\_id\_client CLIENTI.ID\_Client%TYPE,

p\_email CLIENTI.Email%TYPE

) IS

BEGIN

IF exista\_client(p\_id\_client) THEN

UPDATE CLIENTI

SET Email = p\_email

WHERE ID\_Client = p\_id\_client;

ELSE

RAISE exceptie\_client;

END IF;

EXCEPTION

WHEN exceptie\_client THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu acest ID nu exista!');

END modifica\_client\_email;

PROCEDURE sterge\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE

) IS

BEGIN

IF exista\_client(p\_id\_client) THEN

DELETE FROM CLIENTI

WHERE ID\_Client = p\_id\_client;

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

ELSE

RAISE exceptie\_client;

END IF;

EXCEPTION

WHEN exceptie\_client THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu acest ID nu exista!');

END sterge\_client;

FUNCTION exista\_client(

p\_id\_client CLIENTI.ID\_Client%TYPE

) RETURN BOOLEAN IS

v\_unu NUMBER;

BEGIN

SELECT 1 INTO v\_unu

FROM CLIENTI

WHERE ID\_Client = p\_id\_client;

RETURN TRUE;

EXCEPTION

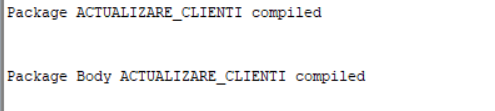
WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END exista\_client;

END actualizare\_clienti;

/



-- Adăugarea unui nou client

EXECUTE actualizare\_clienti.adauga\_client(1001, 'Popescu', 'Ion', 'ion.popescu@gmail.com', '0712345678');

-- Verificarea inserării

SELECT \* FROM CLIENTI WHERE ID\_Client = 1001;

A screenshot of a computer

Description automatically generated

-- Modificarea datelor unui client

EXECUTE actualizare\_clienti.modifica\_client(1001, 'Popescu', 'Ioan', 'ioan.popescu@gmail.com', '0712345679');

-- Verificarea modificării

SELECT \* FROM CLIENTI WHERE ID\_Client = 1001;

A screenshot of a computer

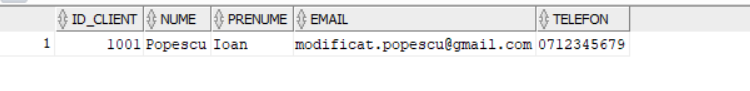
Description automatically generated

-- Modificarea doar a email-ului unui client (suprascriere)

EXECUTE actualizare\_clienti.modifica\_client\_email(1001, 'modificat.popescu@gmail.com');

-- Verificarea modificării

SELECT \* FROM CLIENTI WHERE ID\_Client = 1001;



BEGIN

actualizare\_clienti.sterge\_client(1001);

END;

/

-- Verificarea ștergerii

SELECT \* FROM CLIENTI WHERE ID\_Client = 1001;

A screenshot of a computer

Description automatically generated

PACHET 2

CREATE OR REPLACE PACKAGE actualizare\_camere IS

PROCEDURE adauga\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE,

p\_numarcamera CAMERE.NumarCamera%TYPE,

p\_tip CAMERE.Tip%TYPE,

p\_pretnopate CAMERE.PretNoapte%TYPE,

p\_etaj CAMERE.Etaj%TYPE,

p\_capacitate CAMERE.Capacitate%TYPE

);

PROCEDURE modifica\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE,

p\_numarcamera CAMERE.NumarCamera%TYPE,

p\_tip CAMERE.Tip%TYPE,

p\_pretnopate CAMERE.PretNoapte%TYPE,

p\_etaj CAMERE.Etaj%TYPE,

p\_capacitate CAMERE.Capacitate%TYPE

);

PROCEDURE modifica\_camera\_pret(

p\_id\_camera CAMERE.ID\_Camera%TYPE,

p\_pretnopate CAMERE.PretNoapte%TYPE

);

PROCEDURE sterge\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE

);

FUNCTION exista\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE

) RETURN BOOLEAN;

exceptie\_camera EXCEPTION;

END actualizare\_camere;

/

CREATE OR REPLACE PACKAGE BODY actualizare\_camere IS

PROCEDURE adauga\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE,

p\_numarcamera CAMERE.NumarCamera%TYPE,

p\_tip CAMERE.Tip%TYPE,

p\_pretnopate CAMERE.PretNoapte%TYPE,

p\_etaj CAMERE.Etaj%TYPE,

p\_capacitate CAMERE.Capacitate%TYPE

) IS

BEGIN

IF exista\_camera(p\_id\_camera) THEN

RAISE exceptie\_camera;

ELSE

INSERT INTO CAMERE (ID\_Camera, NumarCamera, Tip, PretNoapte, Etaj, Capacitate)

VALUES (p\_id\_camera, p\_numarcamera, p\_tip, p\_pretnopate, p\_etaj, p\_capacitate);

END IF;

EXCEPTION

WHEN exceptie\_camera THEN

DBMS\_OUTPUT.PUT\_LINE('Camera existenta!');

END adauga\_camera;

PROCEDURE modifica\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE,

p\_numarcamera CAMERE.NumarCamera%TYPE,

p\_tip CAMERE.Tip%TYPE,

p\_pretnopate CAMERE.PretNoapte%TYPE,

p\_etaj CAMERE.Etaj%TYPE,

p\_capacitate CAMERE.Capacitate%TYPE

) IS

BEGIN

IF exista\_camera(p\_id\_camera) THEN

UPDATE CAMERE

SET NumarCamera = p\_numarcamera, Tip = p\_tip, PretNoapte = p\_pretnopate, Etaj = p\_etaj, Capacitate = p\_capacitate

WHERE ID\_Camera = p\_id\_camera;

ELSE

RAISE exceptie\_camera;

END IF;

EXCEPTION

WHEN exceptie\_camera THEN

DBMS\_OUTPUT.PUT\_LINE('Camera cu acest ID nu exista!');

END modifica\_camera;

-- Supraîncarcare a procedurii modifica\_camera

PROCEDURE modifica\_camera\_pret(

p\_id\_camera CAMERE.ID\_Camera%TYPE,

p\_pretnopate CAMERE.PretNoapte%TYPE

) IS

BEGIN

IF exista\_camera(p\_id\_camera) THEN

UPDATE CAMERE

SET PretNoapte = p\_pretnopate

WHERE ID\_Camera = p\_id\_camera;

ELSE

RAISE exceptie\_camera;

END IF;

EXCEPTION

WHEN exceptie\_camera THEN

DBMS\_OUTPUT.PUT\_LINE('Camera cu acest ID nu exista!');

END modifica\_camera\_pret;

PROCEDURE sterge\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE

) IS

BEGIN

IF exista\_camera(p\_id\_camera) THEN

DELETE FROM CAMERE

WHERE ID\_Camera = p\_id\_camera;

DBMS\_OUTPUT.PUT\_LINE('Camera cu ID-ul ' || p\_id\_camera || ' a fost stearsa!');

ELSE

RAISE exceptie\_camera;

END IF;

EXCEPTION

WHEN exceptie\_camera THEN

DBMS\_OUTPUT.PUT\_LINE('Camera cu acest ID nu exista!');

END sterge\_camera;

FUNCTION exista\_camera(

p\_id\_camera CAMERE.ID\_Camera%TYPE

) RETURN BOOLEAN IS

v\_unu NUMBER;

BEGIN

SELECT 1 INTO v\_unu

FROM CAMERE

WHERE ID\_Camera = p\_id\_camera;

RETURN TRUE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN FALSE;

END exista\_camera;

END actualizare\_camere;

/

A close up of words

Description automatically generated

-- Adăugarea unei noi camere

EXECUTE actualizare\_camere.adauga\_camera(20, '305', 'Dubla Standard', 300.00, 3, 2);

-- Verificarea inserării

SELECT \* FROM CAMERE WHERE ID\_Camera = 20;

A screenshot of a computer

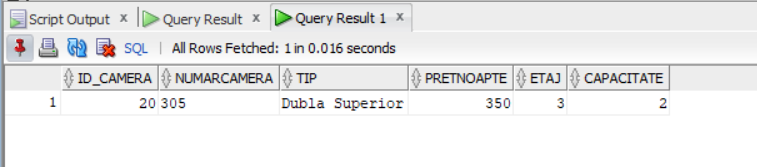
Description automatically generated

-- Modificarea datelor unei camere

EXECUTE actualizare\_camere.modifica\_camera(20, '305', 'Dubla Superior', 350.00, 3, 2);

-- Verificarea modificării

SELECT \* FROM CAMERE WHERE ID\_Camera = 20;

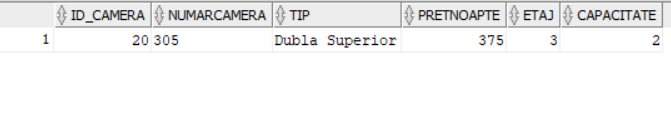


-- Modificarea doar a prețului unei camere

EXECUTE actualizare\_camere.modifica\_camera\_pret(20, 375.00);

-- Verificarea modificării

SELECT \* FROM CAMERE WHERE ID\_Camera = 20;



-- Ștergerea unei camere folosind CALL

CALL actualizare\_camere.sterge\_camera(20);

-- Verificarea ștergerii

SELECT \* FROM CAMERE WHERE ID\_Camera = 20;

A close-up of a sign

Description automatically generated

-- Apelarea dintr-un bloc anonim

BEGIN

actualizare\_camere.sterge\_camera(20);

END;

/

A close up of a text

Description automatically generated

DECLANSATORI

### 1 **. Declanșator BEFORE INSERT: Prevenirea suprapunerii rezervărilor**

CREATE OR REPLACE TRIGGER prevenire\_suprapunere\_rezervari

BEFORE INSERT ON REZERVARI

FOR EACH ROW

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM REZERVARI

WHERE ID\_Camera = :NEW.ID\_Camera

AND (:NEW.DataInceput BETWEEN DataInceput AND DataSfarsit

OR :NEW.DataSfarsit BETWEEN DataInceput AND DataSfarsit

OR :NEW.DataInceput <= DataInceput AND :NEW.DataSfarsit >= DataSfarsit);

IF v\_count > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Suprapunere de rezervări detectată pentru camera ' || :NEW.ID\_Camera);

END IF;

END;

/



### 3. Declanșator BEFORE DELETE: Salvarea într-o tabelă de audit

Pentru acest declanșator, întâi trebuie să creezi o tabelă de audit care să stocheze detaliile rezervărilor șterse. Apoi, vom crea declanșatorul propriu-zis.

CREATE TABLE AUDIT\_REZERVARI (

ID\_Rezervare INT,

ID\_Client INT,

ID\_Camera INT,

DataInceput DATE,

DataSfarsit DATE,

Status VARCHAR2(50),

Data\_Stergere DATE

);



A close-up of a white background

Description automatically generated

CREATE OR REPLACE TRIGGER audit\_stergere\_rezervare

BEFORE DELETE ON REZERVARI

FOR EACH ROW

BEGIN

INSERT INTO AUDIT\_REZERVARI (ID\_Rezervare, ID\_Client, ID\_Camera, DataInceput, DataSfarsit, Status, Data\_Stergere)

VALUES (:OLD.ID\_Rezervare, :OLD.ID\_Client, :OLD.ID\_Camera, :OLD.DataInceput, :OLD.DataSfarsit, :OLD.Status, SYSDATE);

END;

/

### Declanșator AFTER INSERT: Actualizarea statusului camerei

CREATE OR REPLACE TRIGGER actualizare\_status\_camera

AFTER INSERT ON REZERVARI

FOR EACH ROW

BEGIN

UPDATE CAMERE

SET Tip = 'Rezervată'

WHERE ID\_Camera = :NEW.ID\_Camera;

END;

/

A black text on a white background

Description automatically generated