

# **Project SGBD**

Nume Student: Atodiresei Irina

Grupa: 1053, Seria C

Conf. univ. dr. Alexandra Corbea

## **Descrierea bazei de date**

Proiectul are ca scop analizarea activității unei biblioteci.

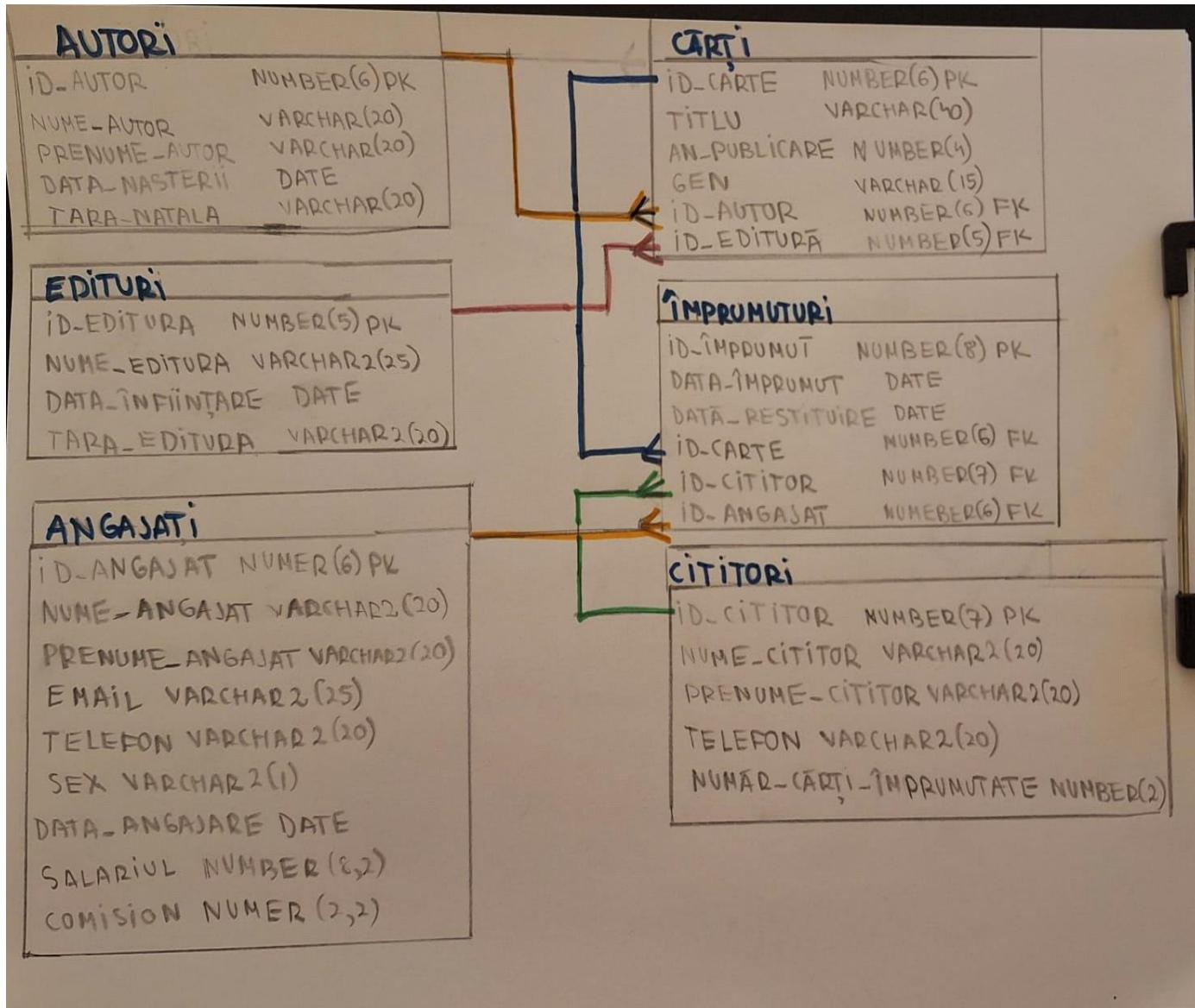
Tema aleasă pentru baza de date a unei biblioteci este concepută pentru a gestiona eficient informațiile legate de cărți, autori, edituri, cititori, angajați și împrumuturi în cadrul unei instituții bibliotecare. Fiecare tabelă din această bază de date contribuie la organizarea și accesarea ușoară a datelor.

- Tabela Cărți: Stochează detalii despre cărți, inclusiv titlul, autorul, editura, anul publicării și genul, oferind astfel informații esențiale despre fiecare carte din colecție.(fiecare carte avand id-ul unic și cheile externe id\_autor și id\_editura)
- Tabela Scriitori: Conține informații despre autori, precum numele și prenumele acestora, data nașterii și țara natală, permitând gestionarea și căutarea simplă a autorilor. (fiecare scriitor avand id-ul unic)
- Tabela Edituri: Păstrează detalii despre edituri, inclusiv numele, anul înființării și țara de proveniență, pentru a urmări cărțile publicate de diferite edituri. .(fiecare editura avand id-ul unic)
- Tabela Cititori: Menține informații despre cititori, precum nume, prenume, numarul de telefon, numărul de cărți împrumutate, permitând bibliotecii să țină evidența cititorilor săi. .(fiecare cititor avand id-ul unic)
- Tabela Angajati: Gestionează informații despre angajați, precum nume, prenume, email, telefon, sex, data angajării, salariul, comisionul și id-ul manager-ului căruia sunt subordonați, pentru a ține evidența personalului bibliotecii. (fiecare angajat avand id-ul unic)
- Tabela Împrumuturi: Monitorizează tranzacțiile de împrumut, inclusiv cărțile împrumutate, cititorii implicați, angajații implicați, precum și datele de împrumut și restituire. .(fiecare împrumut avand id-ul unic și cheile externe id\_carte, id\_cititor, id\_angajat)

Prin relațiile dintre aceste tabele, sistemul permite bibliotecii să gestioneze eficient colecția de cărți, să țină evidența autorilor și editurilor asociate, să monitorizeze cititorii și angajații și să urmărească tranzacțiile de împrumut.

## 2.Schema

(în SQL DEVELOPER tabelele au denumirea cu atodireseiirina\_ în față)



# TEMA 1

-- 1. Afişarea numărului total de cărți disponibile (cursor implicit+structura de control if..then..else)

DECLARE

```
v_total_carti NUMBER;
```

BEGIN

```
    SELECT COUNT(*) INTO v_total_carti FROM atodireseiirina_carti;
```

```
    IF v_total_carti > 100 THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Biblioteca are peste 100 de carti.');
```

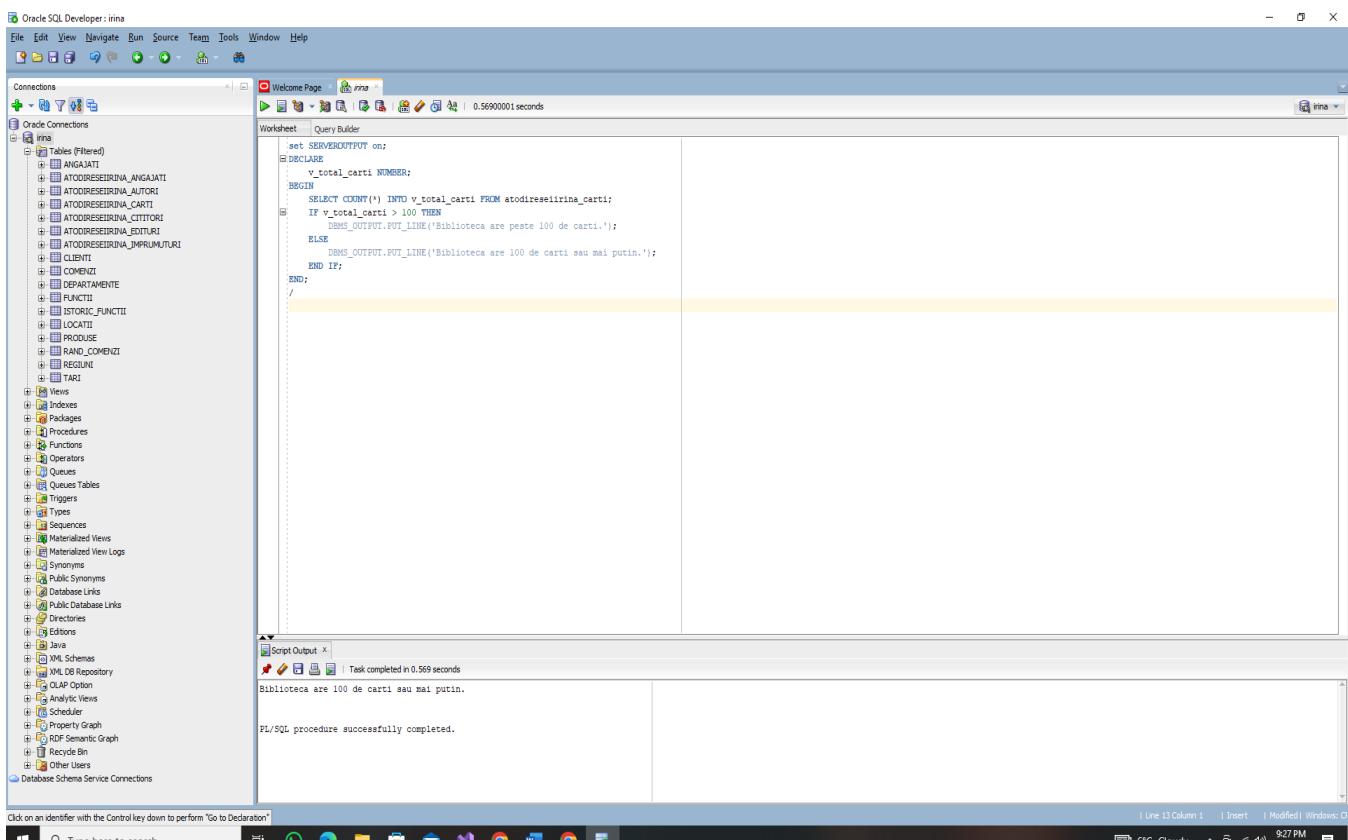
```
    ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('Biblioteca are 100 de carti sau mai putin.');
```

```
    END IF;
```

```
END;
```

```
/
```



The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** A tree view showing the database connection "irina" and its schema objects like ANGAJATI, ATODIRESEIRINA\_CARTI, etc.
- Worksheet:** The main workspace where the PL/SQL code is written. The code is:

```
set SERVEROUTPUT on;
DECLARE
    v_total_carti NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_total_carti FROM atodireseiirina_carti;
    IF v_total_carti > 100 THEN
        DBMS_OUTPUT.PUT_LINE('Biblioteca are peste 100 de carti.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Biblioteca are 100 de carti sau mai putin.');
    END IF;
/

```
- Script Output:** A panel at the bottom showing the execution results:

```
Biblioteca are 100 de carti sau mai putin.

PL/SQL procedure successfully completed.
```
- Status Bar:** Shows the message "Task completed in 0.569 seconds".
- Bottom Bar:** Shows the date and time as "9:27 PM".

-- 2. Creșterea salariului angajaților cu vechime mai mare de 5 ani (CASE+CURSOR EXPLICIT)

DECLARE

CURSOR c\_angajati IS

SELECT id\_angajat, salariu, data\_angajare FROM atodireseirina\_angajati;

v\_ani NUMBER;

BEGIN

FOR v\_angajat IN c\_angajati LOOP

v\_ani := TRUNC(MONTHS\_BETWEEN(SYSDATE, v\_angajat.data\_angajare) / 12);

CASE

WHEN v\_ani > 5 THEN

UPDATE atodireseirina\_angajati

SET salariu = v\_angajat.salariu \* 1.10

WHERE id\_angajat = v\_angajat.id\_angajat;

DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_angajat.id\_angajat || ' are vechime de ' || v\_ani || ' ani și a primit +10%.');

WHEN v\_ani <= 5 THEN

UPDATE atodireseirina\_angajati

SET salariu = v\_angajat.salariu \* 1.05

WHERE id\_angajat = v\_angajat.id\_angajat;

DBMS\_OUTPUT.PUT\_LINE('Angajatul ' || v\_angajat.id\_angajat || ' are vechime de ' || v\_ani || ' ani și a primit +5%.');

END CASE;

END LOOP;

END;

/

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The Connections sidebar lists 'irina' as the current connection, which contains tables like 'Views', 'Indexes', 'Procedures', and 'Functions'. The main area has tabs for 'Worksheet' and 'Query Builder'. The 'Worksheet' tab displays a PL/SQL script that updates salaries based on years of service. The 'Script Output' tab shows the results of the execution, indicating salary increases for various employees.

```

END;
/
DECLARE
    CURSOR c_angajati IS
        SELECT id_angajat, salariu, data_angajare FROM atodireseiirina_angajati;

    v_ani NUMBER;
BEGIN
    FOR v_angajat IN c_angajati LOOP
        v_ani := TRUNC(MONTHS_BETWEEN(SYSDATE, v_angajat.data_angajare) / 12);

        CASE
            WHEN v_ani > 5 THEN
                UPDATE atodireseiirina_angajati
                SET salariu = v_angajat.salariu * 1.10
                WHERE id_angajat = v_angajat.id_angajat;

                DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || ' are vechime de ' || v_ani || ' ani și a primit +10%');

            WHEN v_ani <= 5 THEN
                UPDATE atodireseiirina_angajati
                SET salariu = v_angajat.salariu * 1.05
                WHERE id_angajat = v_angajat.id_angajat;

                DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || ' are vechime de ' || v_ani || ' ani și a primit +5%');
        END CASE;
    END LOOP;
END;
/

```

Script Output:

```

Angajatul 1 are vechime de 15 ani și a primit +10%.
Angajatul 2 are vechime de 11 ani și a primit +10%.
Angajatul 3 are vechime de 6 ani și a primit +10%.
Angajatul 4 are vechime de 4 ani și a primit +5%.
Angajatul 5 are vechime de 13 ani și a primit +10%.
Angajatul 6 are vechime de 14 ani și a primit +10%.
Angajatul 8 are vechime de 9 ani și a primit +10%.
Angajatul 10 are vechime de 5 ani și a primit +5%.

PL/SQL procedure successfully completed.

```

-- 3. actualizarea comisionului angajaților în funcție de sex

```

BEGIN

    FOR v_angajat IN (
        SELECT id_angajat, sex FROM atodireseiirina_angajati
    ) LOOP

        CASE v_angajat.sex

            WHEN 'F' THEN

                UPDATE atodireseiirina_angajati
                SET comision = 0.12
                WHERE id_angajat = v_angajat.id_angajat;
        END IF;
    END LOOP;

```

```
DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'femeie comision 0.12');
```

```
WHEN 'M' THEN
```

```
    UPDATE atodireseiirina_angajati
```

```
    SET comision = 0.10
```

```
    WHERE id_angajat = v_angajat.id_angajat;
```

```
DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'barbat comision 0.10');
```

```
END CASE;
```

```
END LOOP;
```

```
END;
```

```
/
```

The screenshot shows the Oracle SQL Developer interface. The top half displays the PL/SQL code in the 'Script Editor' window. The bottom half shows the 'Script Output' window where the procedure has been successfully executed, displaying 10 rows of output.

```
BEGIN
  FOR v_angajat IN (
    SELECT id_angajat, sex FROM atodireseiirina_angajati
  ) LOOP
    CASE v_angajat.sex
      WHEN 'F' THEN
        UPDATE atodireseiirina_angajati
        SET comision = 0.12
        WHERE id_angajat = v_angajat.id_angajat;
        DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'femeie comision 0.12');

      WHEN 'M' THEN
        UPDATE atodireseiirina_angajati
        SET comision = 0.10
        WHERE id_angajat = v_angajat.id_angajat;
        DBMS_OUTPUT.PUT_LINE('Angajatul ' || v_angajat.id_angajat || 'barbat comision 0.10');

    END CASE;
  END LOOP;
END;
```

Script Output | Task completed in 0.228 seconds

PL/SQL procedure successfully completed.

```
Angajatul 1femeie comision 0.12
Angajatul 2barbat comision 0.10
Angajatul 3femeie comision 0.12
Angajatul 4femeie comision 0.12
Angajatul 5femeie comision 0.12
Angajatul 6barbat comision 0.10
Angajatul 7femeie comision 0.12
Angajatul 8femeie comision 0.12
Angajatul 10femeie comision 0.12
```

PL/SQL procedure successfully completed.

-- 4. Actualizarea numărului de cărți împrumutate pentru cititor (cursor explicit + UPDATE)

DECLARE

CURSOR c\_cititori IS

SELECT id\_cititor FROM atodireseiirina\_cititori;

v\_nr\_imprumuturi NUMBER;

BEGIN

FOR v\_cititor IN c\_cititori LOOP

SELECT COUNT(\*) INTO v\_nr\_imprumuturi

FROM atodireseiirina\_imprumuturi

WHERE id\_cititor = v\_cititor.id\_cititor

AND data\_restituire IS NULL;

UPDATE atodireseiirina\_cititori

SET numar\_carti\_imprumutate = v\_nr\_imprumuturi

WHERE id\_cititor = v\_cititor.id\_cititor;

DBMS\_OUTPUT.PUT\_LINE('Cititorul cu ID ' || v\_cititor.id\_cititor ||

' are ' || v\_nr\_imprumuturi || ' cărți împrumutate (nerestituite).');

END LOOP;

COMMIT;

END;

/

```

DECLARE
    CURSOR c_cititori IS
        SELECT id_cititor FROM atodireselirina_cititori;

    v_nr_imprumuturi NUMBER;
BEGIN
    FOR v_cititor IN c_cititori LOOP
        SELECT COUNT(*) INTO v_nr_imprumuturi
        FROM atodireselirina_imprumuturi
        WHERE id_cititor = v_cititor.id_cititor
            AND data_restituire IS NULL;

        UPDATE atodireselirina_cititori
        SET numar_carte_imprumutate = v_nr_imprumuturi
        WHERE id_cititor = v_cititor.id_cititor;

        DBMS_OUTPUT.PUT_LINE('Cititorul cu ID ' || v_cititor.id_cititor ||
            ' are ' || v_nr_imprumuturi || ' cărți împrumutate (nerestituire).');
    END LOOP;

    COMMIT;
END;
/

```

Script Output | Task completed in 0.374 seconds

Angajatul 5 are vechime de 13 ani și a primit +10%.  
 Angajatul 6 are vechime de 14 ani și a primit +10%.  
 Angajatul 8 are vechime de 9 ani și a primit +10%.  
 Angajatul 10 are vechime de 5 ani și a primit +5%.

PL/SQL procedure successfully completed.

Cititorul cu ID 1 are 0 cărți împrumutate (nerestituire).  
 Cititorul cu ID 1000 are 1 cărți împrumutate (nerestituire).  
 Cititorul cu ID 1001 are 0 cărți împrumutate (nerestituire).  
 Cititorul cu ID 1002 are 0 cărți împrumutate (nerestituire).  
 Cititorul cu ID 1003 are 0 cărți împrumutate (nerestituire).  
 Cititorul cu ID 1004 are 0 cărți împrumutate (nerestituire).  
 Cititorul cu ID 1005 are 0 cărți împrumutate (nerestituire).

-- 5. Stergerea cititori care nu mai au împrumuturi înregistrate

BEGIN

FOR v\_cititor IN (

```

        SELECT id_cititor FROM atodireselirina_cititori
    ) LOOP

```

DECLARE

v\_nr\_imprumuturi NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_nr\_imprumuturi

FROM atodireselirina\_imprumuturi

WHERE id\_cititor = v\_cititor.id\_cititor;

```

IF v_nr_imprumuturi = 0 THEN
    DELETE FROM atodireseiirina_cititor
    WHERE id_cititor = v_cititor.id_cititor;

    DBMS_OUTPUT.PUT_LINE('Cititorul ' || v_cititor.id_cititor || ' a fost șters (fără
împrumuturi).');

END IF;

END;

END LOOP;

ROLLBACK;

```

END;

```

END LOOP;

COMMIT;
END;
/

BEGIN
FOR v_cititor IN (
    SELECT id_cititor FROM atodireseiirina_cititor
) LOOP
DECLARE
    v_nr_imprumuturi NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_nr_imprumuturi
    FROM atodireseiirina_imprumuturi
    WHERE id_cititor = v_cititor.id_cititor;

    IF v_nr_imprumuturi = 0 THEN
        DELETE FROM atodireseiirina_cititor
        WHERE id_cititor = v_cititor.id_cititor;

        DBMS_OUTPUT.PUT_LINE('Cititorul ' || v_cititor.id_cititor || ' a fost șters (fără imprumuturi).');
    END IF;
END;
END LOOP;

ROLLBACK; -- sau COMMIT, dacă vrei să păstrezi ștergerile
END;

```

Script Output

| Task completed in 0.648 seconds

Cititorul 1015 a fost șters (fără imprumuturi).  
Cititorul 1016 a fost șters (fără imprumuturi).  
Cititorul 1017 a fost șters (fără imprumuturi).  
Cititorul 1018 a fost șters (fără imprumuturi).  
Cititorul 1019 a fost șters (fără imprumuturi).  
Cititorul 1020 a fost șters (fără imprumuturi).  
Cititorul 1021 a fost șters (fără imprumuturi).  
Cititorul 1022 a fost șters (fără imprumuturi).  
Cititorul 1023 a fost șters (fără imprumuturi).  
Cititorul 1024 a fost șters (fără imprumuturi).

PL/SQL procedure successfully completed.

## Tema 2

### 1. Afişarea cărţilor publicate înainte de anul 1900

(Excepție implicită: NO\_DATA\_FOUND, excepție explicită definită de utilizator dacă cartea e prea veche)

```
set serveroutput on;
```

```
DECLARE
```

```
    CURSOR c_carti_vechi IS
```

```
        SELECT id_carte, titlu, an_publicare FROM atodireseiirina_carte WHERE an_publicare < 1900;  
        v_carte c_carti_vechi%ROWTYPE;
```

```
BEGIN
```

```
    OPEN c_carti_vechi;
```

```
LOOP
```

```
    FETCH c_carti_vechi INTO v_carte;
```

```
    EXIT WHEN c_carti_vechi%NOTFOUND;
```

```
    IF v_carte.an_publicare < 1800 THEN
```

```
        RAISE_APPLICATION_ERROR(-20001, 'Cartea ' || v_carte.titlu || ' este prea veche pentru a  
        fi păstrată în bibliotecă.');
```

```
    END IF;
```

```
    DBMS_OUTPUT.PUT_LINE('Carte veche: ' || v_carte.titlu || '(' || v_carte.an_publicare || ')');
```

```
END LOOP;
```

```
CLOSE c_carti_vechi;
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Nu există cărți vechi în bibliotecă.');
```

```
    WHEN OTHERS THEN
```

```

DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);

END;

/

```

```

set serveroutput on;
DECLARE
    CURSOR c_carte_vechi IS
        SELECT id_carte, titlu, an_publicare FROM atodireseirina_carti WHERE an_publicare < 1900;
    v_carte c_carte_vechi%ROWTYPE;
BEGIN
    OPEN c_carte_vechi;
    LOOP
        FETCH c_carte_vechi INTO v_carte;
        EXIT WHEN c_carte_vechi%NOTFOUND;

        IF v_carte.an_publicare < 1800 THEN
            RAISE_APPLICATION_ERROR(-20001, 'Cartea ' || v_carte.titlu || ' este prea veche pentru a fi păstrată în bibliotecă.');
        END IF;
    END LOOP;
    CLOSE c_carte_vechi;
END;

```

Script Output:

```

Carte veche: Amintiri din Copilarie (1892)
Carte veche: Povestea lui Harap-Alb (1877)
Carte veche: Ivan Turbinice (1878)
Carte veche: Chirita în Iasi (1850)
Carte veche: Sanziana și Pepelea (1882)
Carte veche: Luceafărul (1883)
Carte veche: Floare albastra (1873)
Carte veche: Cartea Hamlet este prea veche pentru a fi păstrată în bibliotecă.

Eroare: ORA-20001: Cartea Hamlet este prea veche pentru a fi păstrată în bibliotecă.

PL/SQL procedure successfully completed.

```

## 2. Calculul mediei salariilor (cu tratament pentru împărțire la 0)

(Excepție implicită: ZERO\_DIVIDE, excepție explicită: salară inacceptabilă)

```
set serveroutput on;
```

```
DECLARE
```

```
    v_total_salariu NUMBER := 0;
```

```
    v_nr_angajati NUMBER := 0;
```

```
    v_medie NUMBER;
```

```
BEGIN
```

```
    SELECT SUM(salariu), COUNT(*) INTO v_total_salariu, v_nr_angajati FROM
atodireseirina_angajati;
```

```
    IF v_total_salariu > 10000000 THEN
```

```
        RAISE_APPLICATION_ERROR(-20002, 'Totalul salariilor este exagerat!');
```

```
    END IF;
```

```
v_medic := v_total_salariu / v_nr_angajati;
```

```
DBMS_OUTPUT.PUT_LINE('Media salariilor este: ' || TO_CHAR(v_medic, '9999.99'));
```

EXCEPTION

```
WHEN ZERO_DIVIDE THEN
```

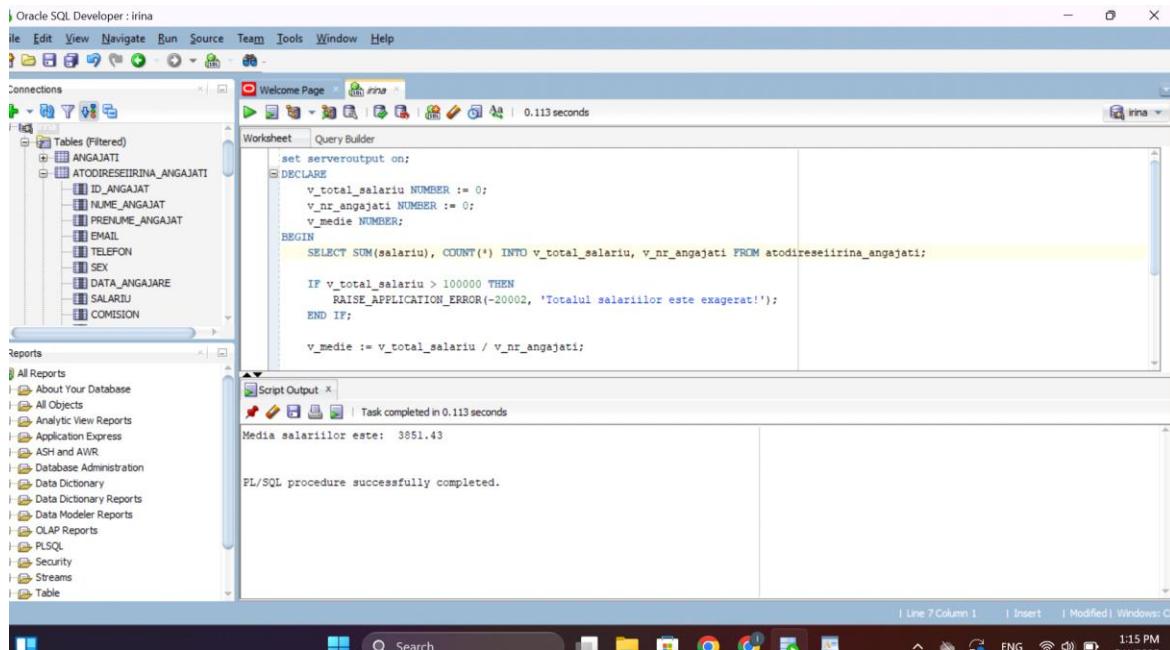
```
    DBMS_OUTPUT.PUT_LINE('Nu există angajați în baza de date.');
```

```
WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
```

```
END;
```

```
/
```



3. Găsirea unui autor unic după nume (cu TOO\_MANY\_ROWS)

(Excepție implicită: TOO\_MANY\_ROWS, excepție explicită: autor inexistent)

```
DECLARE
```

```
    v_id_autor NUMBER;
```

```
    v_nume_autor VARCHAR2(100) := 'Eminescu';
```

```
BEGIN
```

```
    SELECT id_autor INTO v_id_autor
```

```
    FROM atodireseiirina_autori
```

```

WHERE nume_autor = v_nume_autor;

DBMS_OUTPUT.PUT_LINE('ID autor: ' || v_id_autor);

EXCEPTION
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Autorul ' || v_nume_autor || ' nu a fost găsit.');
WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Există mai mulți autori cu acest nume. Specificați prenumele.');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
END;
/

```

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the PL/SQL code provided above. The 'Script Output' tab shows the execution results:

```

ID autor: 102
PL/SQL procedure successfully completed.

```

#### 4. Încercarea de a adăuga un împrumut pentru o carte deja împrumutată (excepție explicită)

(Folosire de cursor + RAISE\_APPLICATION\_ERROR)

DECLARE

CURSOR c\_imprumuturi IS

SELECT id\_carte FROM atodireseiirina\_Imprumuturi WHERE data\_restituire IS NULL;

v\_carte\_imprumutata c\_imprumuturi%ROWTYPE;

v\_id\_carte NUMBER := 112;

v\_gasit BOOLEAN := FALSE;

BEGIN

OPEN c\_imprumuturi;

LOOP

FETCH c\_imprumuturi INTO v\_carte\_imprumutata;

EXIT WHEN c\_imprumuturi%NOTFOUND;

IF v\_carte\_imprumutata.id\_carte = v\_id\_carte THEN

v\_gasit := TRUE;

EXIT;

END IF;

END LOOP;

CLOSE c\_imprumuturi;

IF v\_gasit THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Cartea cu ID-ul ' || v\_id\_carte || ' este deja împrumutată.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Cartea este disponibilă pentru împrumut.');

END IF;

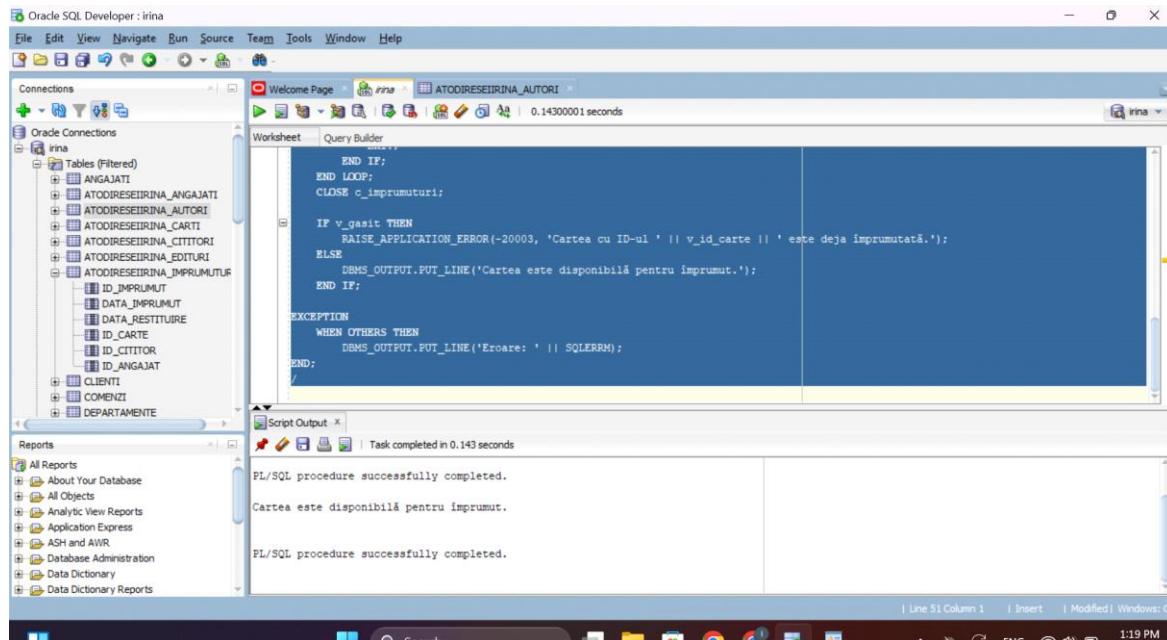
EXCEPTION

WHEN OTHERS THEN

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

END;

/



## Tema 3

### Pachet 1 Pachet\_Biblioteca: 1 procedura, 3 functii:

```
CREATE OR REPLACE PACKAGE Pachet_Biblioteca AS
```

-- Proceduri

```
PROCEDURE Lista_Imprumuturi_Cititor(id_cititor NUMBER);
```

-- Functii

```
FUNCTION Titlu_Exista(titlu_carte VARCHAR2) RETURN NUMBER;
```

```
FUNCTION Numar_Imprumuturi(id_cititor NUMBER) RETURN NUMBER;
```

```
FUNCTION Carti_Editura(id_editura NUMBER) RETURN NUMBER;
```

```
END;
```

```
CREATE OR REPLACE PACKAGE BODY Pachet_Biblioteca AS
```

```
PROCEDURE Imprumuta_Carte(id_carte NUMBER, id_cititor NUMBER, id_angajat NUMBER)
IS
```

```
    new_id NUMBER;
```

```
BEGIN
```

```

    SELECT NVL(MAX(ID_IMPRUMUT), 0) + 1 INTO new_id FROM
    ATODIRESEIIRINA_IMPRUMUTURI;

    INSERT INTO ATODIRESEIIRINA_IMPRUMUTURI VALUES (new_id, SYSDATE, NULL,
    id_carte, id_cititor, id_angajat);

    END;

```

```

PROCEDURE Lista_Imprumuturi_Cititor(id_cititor NUMBER) IS
    CURSOR imprumuturi IS
        SELECT ID_CARTE, DATA_IMPRUMUT, DATA_RESTITUIRE
        FROM ATODIRESEIIRINA_IMPRUMUTURI
        WHERE ID_CITITOR = id_cititor;

    BEGIN
        FOR rec IN imprumuturi LOOP
            DBMS_OUTPUT.PUT_LINE('Carte: ' || rec.ID_CARTE || ' | Imprumut: ' ||
            rec.DATA_IMPRUMUT || ' | Restituire: ' || rec.DATA_RESTITUIRE);
        END LOOP;
    END;

```

```

FUNCTION Titlu_Exista(titlu_carte VARCHAR2) RETURN NUMBER IS
    nr NUMBER;
BEGIN
    SELECT COUNT(*) INTO nr
    FROM ATODIRESEIIRINA_CARTI
    WHERE TITLU = titlu_carte;
    RETURN nr;
END;

```

```

FUNCTION Numar_Imprumuturi(id_cititor NUMBER) RETURN NUMBER IS
    nr NUMBER;
BEGIN
    SELECT COUNT(*) INTO nr
    FROM ATODIRESEIIRINA_IMPRUMUTURI

```

```

        WHERE ID_CITITOR = id_cititor;
        RETURN nr;
    END;

FUNCTION Carti_Editura(id_editura NUMBER) RETURN NUMBER IS
    total NUMBER;
BEGIN
    SELECT COUNT(*) INTO total FROM ATODIRESEIIRINA_CARTI WHERE ID_EDITURA =
id_editura;
    RETURN total;
END;

END;

--apeluri pachet biblioteca
-- Lista_Imprumuturi_Cititor
BEGIN
    Pachet_Biblioteca.Lista_Imprumuturi_Cititor(2);
END;
/
-- Titlul_Exista
BEGIN
    IF Pachet_Biblioteca.Titlul_Exista('Baltagul') > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Cartea exista.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Cartea nu exista.');
    END IF;
END;
/
-- Numar_Imprumuturi
DECLARE

```

```

nr NUMBER;

BEGIN
    nr := Pachet_Biblioteca.Numar_Imprumuturi(2);
    DBMS_OUTPUT.PUT_LINE('Imprumuturi: ' || nr);
END;
/

```

```

-- Carti>Editura

DECLARE
    nr NUMBER;

BEGIN
    nr := Pachet_Biblioteca.Carti>Editura(1);
    DBMS_OUTPUT.PUT_LINE('Carti publicate de editura: ' || nr);
END;
/

```

```

-- Titul_Există
BEGIN
    IF Pachet_Biblioteca.Titul_Există('Belaagul') > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Cartea există.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Cartea nu există.');
    END IF;
END;

-- Numar_Imprumuturi
DECLARE
    nr NUMBER;
BEGIN
    nr := Pachet_Biblioteca.Numar_Imprumuturi(2);
    DBMS_OUTPUT.PUT_LINE('Imprumuturi: ' || nr);
END;

-- Carti>Editura
DECLARE
    nr NUMBER;
BEGIN
    nr := Pachet_Biblioteca.Carti>Editura();
    DBMS_OUTPUT.PUT_LINE('Carti publicate de editura: ' || nr);
END;

```

Script Output X | Query Result X

| Carte | Imprumut  | Restituire |
|-------|-----------|------------|
| 2001  | 03-SEP-23 | 20-JUN-23  |
| 2000  | 05-SEP-23 | 30-SEP-23  |
| 2002  | 10-MAR-23 | 20-MAR-23  |
| 2003  | 11-MAY-23 | 28-FEB-23  |
| 2004  | 03-APR-23 | 17-APR-23  |
| 2005  | 04-APR-23 | 21-APR-23  |
| 2006  | 06-MAY-23 | 15-MAY-23  |
| 2004  | 27-JUL-23 | 03-AUG-23  |
| 2007  | 27-AUG-23 | 04-SEP-23  |
| 2003  | 09-JAN-23 | 25-JAN-23  |

PL/SQL procedure successfully completed.

Cartea nu există.

```

CREATE OR REPLACE PACKAGE PACHET_ANGAJATI AS
    PROCEDURE ADAUGA_CITITOR(nume VARCHAR2, prenume VARCHAR2, telefon VARCHAR2);
    PROCEDURE SCHIMBA_SALARIU(id_angajat NUMBER, procent NUMBER);
END;
/

```

```

PROCEDURE ADAUGA_CITITOR IS
    BEGIN
        INSERT INTO ATODIRESEIIRINA_CITITOR (ID_CITITOR, NUME_CITITOR, PRENUME_CITITOR, TELEFON_CITITOR)
        VALUES (NULL, nume, prenume, telefon);
        DBMS_OUTPUT.PUT_LINE('Cititor adaugat cu ID: ' || ID_CITITOR);
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Eroare la adaugarea cititorului');
    END;
END;
/

```

```

PROCEDURE SCHIMBA_SALARIU IS
    CURSOR c1 IS
        SELECT * FROM ANGAJAT WHERE ID_ANGAJAT = id_angajat;
    BEGIN
        FOR r1 IN c1 LOOP
            r1.SALARIU := r1.SALARIU * (1 + procent / 100);
            UPDATE ANGAJAT SET SALARIU = r1.SALARIU WHERE ID_ANGAJAT = id_angajat;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('Salariul a fost schimbat pentru angajatul cu ID: ' || id_angajat);
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Eroare la actualizarea salariului');
    END;
END;
/

```

## Pachet 2 Pachet\_Angajati: 2 proceduri, 2 functii:

CREATE OR REPLACE PACKAGE PACHET\_ANGAJATI AS

-- Proceduri

PROCEDURE Adauga\_Cititor(nume VARCHAR2, prenume VARCHAR2, telefon VARCHAR2);

PROCEDURE Schimba\_Salariu(id\_angajat NUMBER, procent NUMBER);

-- Functii

FUNCTION Cititori\_Activi RETURN NUMBER;

FUNCTION Cel\_Mai\_Productiv\_Angajat RETURN VARCHAR2;

END;

CREATE OR REPLACE PACKAGE BODY PACHET\_ANGAJATI AS

PROCEDURE Adauga\_Cititor(nume VARCHAR2, prenume VARCHAR2, telefon VARCHAR2) IS

new\_id NUMBER;

BEGIN

SELECT NVL(MAX(ID\_CITITOR), 0) + 1 INTO new\_id FROM ATODIRESEIIRINA\_CITITOR;

INSERT INTO ATODIRESEIIRINA\_CITITOR VALUES (new\_id, nume, prenume, telefon, 0);

END;

```
PROCEDURE Schimba_Salariu(id_angajat NUMBER, procent NUMBER) IS
BEGIN
    UPDATE ATODIRESEIIRINA_ANGAJATI
    SET SALARIU = SALARIU + SALARIU * procent / 100
    WHERE ID_ANGAJAT = id_angajat;
END;
```

```
FUNCTION Cititori_Activi RETURN NUMBER IS
nr NUMBER;
BEGIN
    SELECT COUNT(DISTINCT ID_CITITOR)
    INTO nr
    FROM ATODIRESEIIRINA_IMPRUMUTURI
    WHERE DATA_RESTITUIRE IS NULL;
    RETURN nr;
END;
```

```
FUNCTION Cel_Mai_Productiv_Angajat RETURN VARCHAR2 IS
angajat VARCHAR2(100);
BEGIN
    SELECT NUME_ANGAJAT || ' ' || PRENUME_ANGAJAT
    INTO angajat
    FROM ATODIRESEIIRINA_ANGAJATI
    WHERE ID_ANGAJAT =
        (SELECT ID_ANGAJAT
        FROM (
            SELECT ID_ANGAJAT, COUNT(*) AS nr
            FROM ATODIRESEIIRINA_IMPRUMUTURI
            GROUP BY ID_ANGAJAT
            ORDER BY nr DESC
        )
    )
```

```

        WHERE ROWNUM = 1
    );
    RETURN angajat;
END;

END;
/

--apelare
-- Adauga_Cititor
BEGIN
    Pachet_Angajati.Adauga_Cititor('Marin', 'Alina', '0734123456');
END;
/

-- Schimba_Salariu
BEGIN
    PACHET_ANGAJATI.Schimba_Salariu(200, 10);
END;
/

-- Cititori_Activi
DECLARE
    nr NUMBER;
BEGIN
    nr := Pachet_Angajati.Cititori_Activi;
    DBMS_OUTPUT.PUT_LINE('Cititori activi: ' || nr);
END;
/

```

```
-- Cel_Mai_Productiv_Angajat

DECLARE
    angajat VARCHAR2(100);

BEGIN
    angajat := Pachet_Angajati.Cel_Mai_Productiv_Angajat;
    DBMS_OUTPUT.PUT_LINE('Cel mai productiv angajat: ' || angajat);
END;

/

```

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Shows a connection named "Inna".
- Worksheet:** Contains the PL/SQL code for the package body.
- Script Output:** Shows the results of the execution, including the compilation message, the successful execution of the procedure, and the output line "Cel mai productiv angajat: Popescu Elena".
- Status Bar:** Displays "Task completed in 0.144 seconds".
- System Tray:** Shows icons for battery, signal strength, and time (10:56 PM).

```

-- Cel_Mai_Productiv_Angajat

DECLARE
    angajat VARCHAR2(100);

BEGIN
    angajat := Pachet_Angajati.Cel_Mai_Productiv_Angajat;
    DBMS_OUTPUT.PUT_LINE('Cel mai productiv angajat: ' || angajat);
END;

/

```

```

Package Body PACHET_ANGAJATI compiled
PL/SQL procedure successfully completed.
Cititori activi: 1
PL/SQL procedure successfully completed.
Cel mai productiv angajat: Popescu Elena
PL/SQL procedure successfully completed.

```

## **Procedura Arhiveaza\_Carti\_Inactive**

Creăm o tabelă auxiliară atodireseiirina\_carti\_inactive, care va conține cărțile ce nu au fost niciodată împrumutate.

```
set serveroutput on;
```

```
CREATE TABLE atodireseiirina_carti_inactive AS
```

```
SELECT * FROM atodireseiirina_carti WHERE 1=0;
```

```
CREATE OR REPLACE PROCEDURE Arhiveaza_Carti_Inactive IS
```

```
CURSOR c_carti IS
```

```
SELECT * FROM atodireseiirina_carti c
```

```
WHERE NOT EXISTS (
```

```
    SELECT 1 FROM atodireseiirina_imprumuturi i
```

```
    WHERE i.id_carte = c.id_carte
```

```
);
```

```
v_carte c_carti%ROWTYPE;
```

```
BEGIN
```

```
OPEN c_carti;
```

```
LOOP
```

```
    FETCH c_carti INTO v_carte;
```

```
    EXIT WHEN c_carti%NOTFOUND;
```

```
-- Inserare în tabela de arhivă
```

```
INSERT INTO atodireseiirina_carti_inactive
```

```
VALUES v_carte;
```

```
-- Ștergere din tabela principală
```

```
DELETE FROM atodireseiirina_carti
```

```
WHERE id_carte = v_carte.id_carte;
```

```
DBMS_OUTPUT.PUT_LINE('Cartea "' || v_carte.titlu || '" a fost arhivată (nu a fost niciodată  
împrumutată).');
```

```
END LOOP;
```

```
CLOSE c_carti;
```

```
COMMIT;
```

```
END;
```

```
/
```

```
BEGIN
```

```
Arhiveaza_Carti_Inactive;
```

```
END;
```

```
/
```

The screenshot shows the Oracle SQL Developer interface with a PL/SQL script running in the Worksheet window and its output displayed in the Script Output window.

**Worksheet Window:**

```
OPEN c_carti;
LOOP
  FETCH c_carti INTO v_carte;
  EXIT WHEN c_carti%NOTFOUND;

  -- Inserare in tabela de arhivă
  INSERT INTO stocireserina_carti_inactive
  VALUES v_carte;

  -- Stergere din tabela principală
  DELETE FROM stocireserina_carti
  WHERE id_carte = v_carte.id_carte;

  DBMS_OUTPUT.PUT_LINE('Cartea "' || v_carte.titlu || '" a fost arhivată (nu a fost niciodată împrumutată).');

END LOOP;
CLOSE c_carti;

COMMIT;
END;
/
BEGIN
  Arhiveaza_Carti_Inactive;
END;
/
```

**Script Output Window:**

```
Cartea "Copiii capitánului Grant" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Mihail Strogoff" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Metamorfoza" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Scrisoarea catre tata" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Cartea de la Moartea lui Iosef" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Fiecle de sagii" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Capodopera neconunscută" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Patul lui Procüst" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Jocul ielilor" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Mandrie și prejudecata" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Persuasivine" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Hoarte pe care le-a fost arhivate" (nu a fost niciodată împrumutată).
Cartea "Cărțile din Orient Express" a fost arhivată (nu a fost niciodată împrumutată).
Cartea "Cetățile infiorate lamaie" a fost arhivată (nu a fost niciodată împrumutată).

PL/SQL procedure successfully completed.
```

# Triggeri

## Setare comision implicit în funcție de sex

```
CREATE OR REPLACE TRIGGER trg_set_comision_default
BEFORE INSERT ON atodireseiirina_angajati
FOR EACH ROW
BEGIN
IF :NEW.sex = 'F' AND :NEW.comision IS NULL THEN
:NEW.comision := 0.12;
ELSIF :NEW.sex = 'M' AND :NEW.comision IS NULL THEN
:NEW.comision := 0.10;
END IF;
END;
```

/

```
SELECT id_angajat, nume_angajat, prenume_angajat, comision
FROM atodireseiirina_angajati
WHERE id_angajat = 998;
```

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections pane displays the 'ina' connection with various schema objects like ANGAJATI, ATODIRESEIIRINA\_ANGAJATI, and others. The central area is the Worksheet tab, which contains the SQL code for creating the trigger. The bottom right shows the Script Output tab with the message 'Trigger TRG\_SET\_COMISION\_DEFAULT compiled'. A table output shows a single row with ID\_ANGAJAT 998, NUME\_ANGAJAT Ionescu, PRENUME\_ANGAJAT Maria, and COMISION .12.

| ID_ANGAJAT | NUME_ANGAJAT | PRENUME_ANGAJAT | COMISION |
|------------|--------------|-----------------|----------|
| 998        | Ionescu      | Maria           | .12      |

## Verificare salariu minim (nu poate fi sub 2000)

BEGIN

```
INSERT INTO atodireseiirina_angajati (
    id_angajat, nume, prenume, email, telefon, sex, data_angajare, salariu, comision, id_manager
) VALUES (
    999, 'Popescu', 'Ion', 'ion.popescu@example.com', '0712345678', 'M',
    TO_DATE('2023-01-01', 'YYYY-MM-DD'), 1800, NULL, NULL
);
```

END;

/

BEGIN

```
INSERT INTO atodireseiirina_angajati (
    id_angajat, nume_angajat, prenume_angajat, email, telefon, sex, data_angajare, salariu, comision,
    id_manager
) VALUES (
    999, 'Popescu', 'Ion', 'ion.popescu@example.com', '0712345678', 'M',
    TO_DATE('2023-01-01', 'YYYY-MM-DD'), 1800, NULL, NULL
);
```

END;

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** Oracle Connections, connected to 'irina'.
- Tables (Filtered):** ANGAJATI, ATODIRESEIRINA\_ANGAJATI, ATODIRESEIRINA\_AUTORI, ATODIRESEIRINA\_CARTI, ATODIRESEIRINA\_CITITORI, ATODIRESEIRINA\_EDITURI, ATODIRESEIRINA\_IMPRUMUTURI, COMENZI, COMENZI\_CITATORI, DEPARTAMENTE, FUNCTII, ISTORIC\_FUNCTII, LOCATII, PRODUS, RAND\_COMENZI, REGUNI, TARI.
- Worksheet:** Query Builder window containing the trigger creation script and the insert statement.

```
CREATE OR REPLACE TRIGGER trg_check_salariu_minim
BEFORE INSERT OR UPDATE ON atodireseiirina_angajati
FOR EACH ROW
BEGIN
    IF :NEW.salariu < 2000 THEN
        RAISE_APPLICATION_ERROR(-20010, 'Salariul nu poate fi sub 2000 RON.');
    END IF;
END;

BEGIN
    INSERT INTO atodireseiirina_angajati (
        id_angajat, nume_angajat, prenume_angajat, email, telefon, sex, data_angajare, salariu, comision, id_manager
    ) VALUES (
        999, 'Popescu', 'Ion', 'ion.popescu@example.com', '0712345678', 'M',
        TO_DATE('2023-01-01', 'YYYY-MM-DD'), 1800, NULL, NULL
    );
END;
/
```

- Script Output:** Task completed in 0.148 seconds.

```
ERROR at line 1:
ORA-20010: Salariul nu poate fi sub 2000 RON.
ORA-06512: at "ATODIRESEII_53.TRG_CHECK_SALARIU_MINIM", line 3
ORA-04088: error during execution of trigger 'ATODIRESEII_53.TRG_CHECK_SALARIU_MINIM'
ORA-06512: at line 2
```

<https://docs.oracle.com/error-help/db/ora-20010/>

## Validare dată restituire: nu poate fi anterioară datei împrumutului

```
CREATE OR REPLACE TRIGGER trg_verificare_data_restituire
BEFORE INSERT OR UPDATE ON atodireseiirina_imprumuturi
FOR EACH ROW
BEGIN
    IF :NEW.data_restituire IS NOT NULL AND :NEW.data_restituire < :NEW.data_imprumut THEN
        RAISE_APPLICATION_ERROR(-20005, 'Data restituirii nu poate fi anterioară datei de
        împrumut.');
    END IF;
END;
```

/

-- Incerc să inserez o restituire în trecut

```
INSERT INTO atodireseiirina_imprumuturi VALUES (
    1502, TO_DATE('2025-05-20', 'YYYY-MM-DD'),
    TO_DATE('2025-05-19', 'YYYY-MM-DD'), -- invalid
    102, 2, 998
);
```

The screenshot shows the Oracle SQL Developer interface with the following details:

- Connections:** A connection named "irina" is selected.
- Tables:** The schema contains several tables, including ANGAJATI, ATODIRESEIIRINA\_ANGAJATI, ATODIRESEIIRINA\_CARTI, ATODIRESEIIRINA\_CITITORI, ATODIRESEIIRINA\_EDITURI, ATODIRESEIIRINA\_IMPRUMUTURI, CLIENTI, COMENZI, DEPARTAMENTE, FUNCTII, ISTORIC\_FUNCTII, LOCATII, PROULICE, RAND\_COMENZI, REGUNI, and TARI.
- Worksheet:** The code for the trigger is pasted into the worksheet area. The trigger checks if the new restitute date is less than the loan date and raises an application error if true.
- Script Output:** The output shows the execution of the trigger and the resulting error message: "ORA-20005: Data restituirii nu poate fi anterioară datei de împrumut." This indicates that the insertion attempt failed due to the validation rule.

