PREZENTARE PROIECT

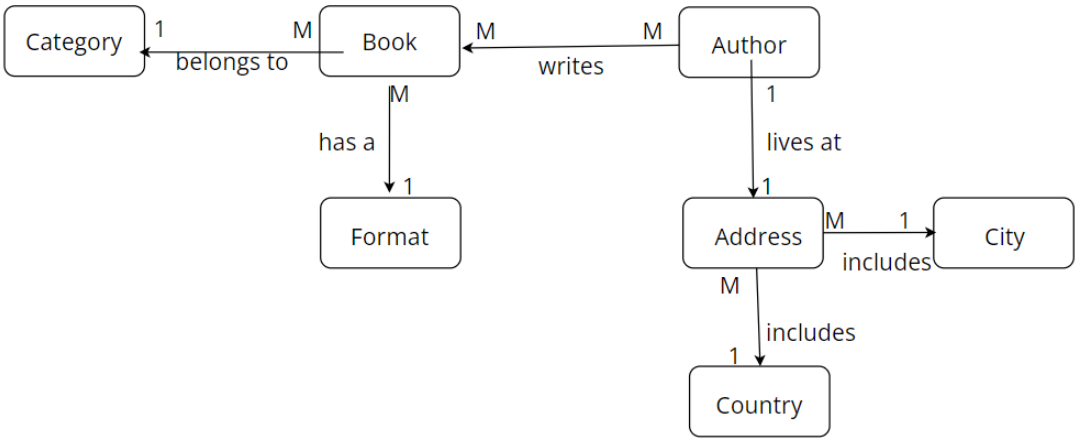
Chitimus Mara Ioana, grupa 405

a) Baza de date contine urmatoarele entitati :

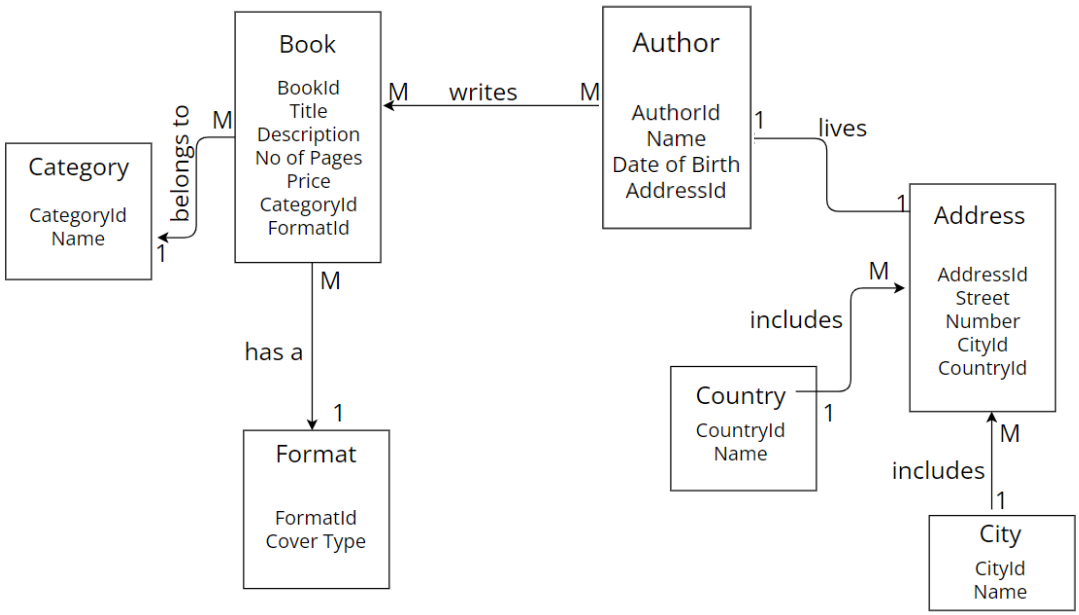
i. Carte (Book)
ii. Categorie (Category)
iii. Format (Format)
iv. Autor (Author)
v. Address (Adresa)
vi. Tara (Country)
vii. Oras (City)

O carte apartine unei categorii si are un anumit format. Unul sau mai multi autori pot scrie una sau mai multe carti. Un autor locuieste la o singura adresa, iar o adresa include o tara si un oras.

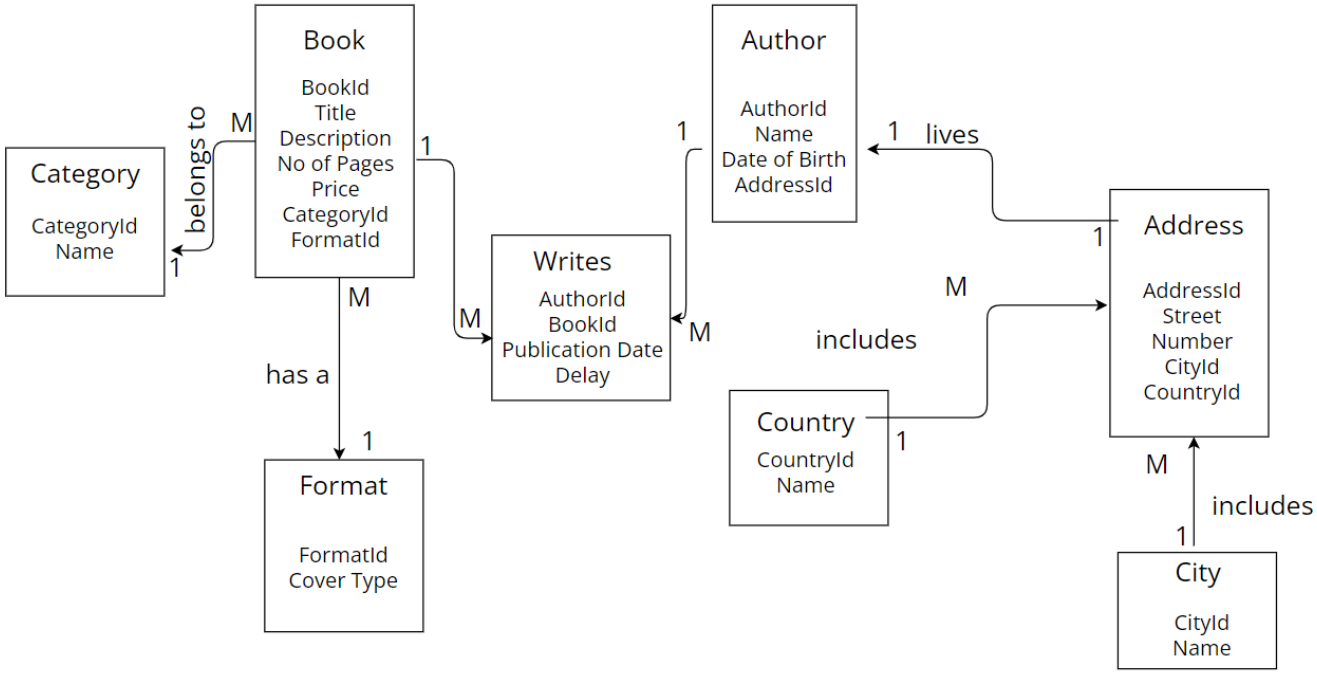
b) Diagrama Entitate Relatie

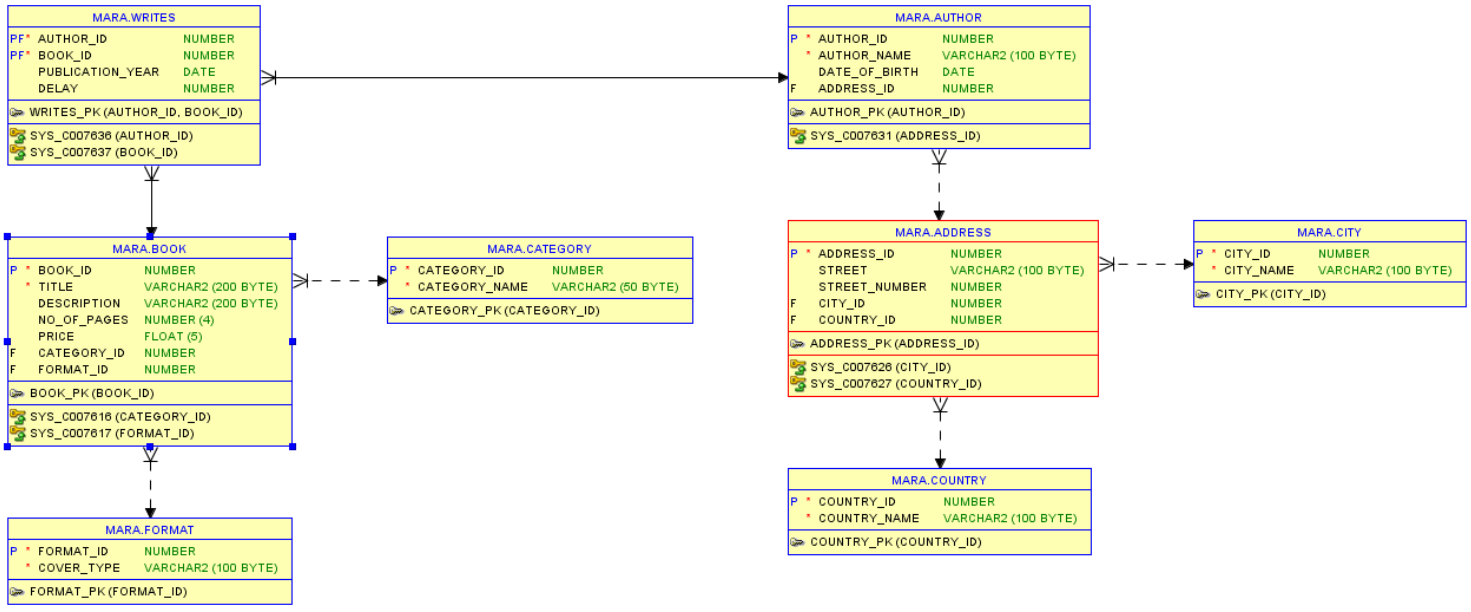


c) Diagrama conceptuala



d) Design logic





e) Am adus deja schema de mai sus in FN3, dar voi aduce exemple si pentru urmatoarele situatii:

- Atribut repetitv
  Sa consideram tabelul Inventar_Magazine cu doua coloane : magazin_id si inventar. Atunci atributele din coloane inventar vor fi considerate atribute repetitive, deoarece au mai multe valori pentru o singura linie din tabel.

MAGAZIN_ID	INVENTAR
Cora01	120 kg faina, 170 kg rosii, 140 kg faina
Carrefour05	130 kg castraveti, 130 kg cirese
Auchan07	250 kg faina, 250 kg zahar
MegaImage15	150 kg zahar, 160 kg faina

Pentru a scapa de atributul repetitiv, vom crea doua coloane in loc de cea e Inventar. Astfel, tabelul va fi in FN1.

MAGAZIN_ID	PRODUS	CANTITATE (KG)
Cora01	rosii	150
Carrefour05	zahar	270
Carrefour05	rosii	170

- Tabel in FN1 dar nu in FN2
  Tabelul de mai sus este in FN1. Dar daca am vrea sa mai adaugam o coloana, Oras, tabelul va fi in FN1 in continuare. Magazin_Id si Produs reprezinta cheia primara a tabelului.

MAGAZIN_ID	PRODUS	CANTITATE (KG)	Oras
Cora01	rosii	150	Bucuresti
Carrefour05	zahar	270	Cluj
Carrefour05	rosii	170	Cluj

Dar un va fi in FN2, deoarece exista atribute care un depind de toata cheia primara. Oras, de exemplu, depende doar de id-ul magazinului, un si de produs. Solutia pentru a-l aduce in FN2 este sa separam tabelul iin doua.

MAGAZIN_ID	Oras
Cora01	Bucuresti
Carrefour05	Cluj

MAGAZIN_ID	PRODUS	CANTITATE (KG)
Cora0	rosii	150
Carrefour05	zahar	270
Carrefour05	rosii	170

Acum vom avea tabelul magazin (cu magazin_id PK) si tabelul Inventar (cu magazin_id si produs ca PK compus). Ambele se afla in FN2.

- Tabel in FN2 dar nu in FN3

MAGAZIN_ID	Oras	Cod Postal
Cora01	Bucuresti	**023**
Carrefour05	Cluj	**566**

Daca la tabelul Magazin adaugam si cod postal, acesta va fi in continuare in FN2, dar nu si in FN3, deoarece atribútele care un sunt PK un depind DOAR de cheia primara. Codul

postal, de exemplu, un depende doar de id-ul magazinului, ci si de oras. (nu poti ave adoua orase cu acelasi cod postal, dar te astepti sa vezi acelasi cod postal pentru acelasi oras in orice inregistrare din tabel). Solutia pentru a il aduce in FN3 este sa il separam in doua tabele.

MAGAZIN_ID	Oras
Cora01	Bucuresti
Carrefour05	Cluj

Oras	Cod Postal
Bucuresti	**023**
Cluj	**566**

Asa avem doua tabele, ambele in FN3. Oras este cheie primara pentru tabelul Adresa, iar tabelul Magazin are magazin_id drept cheie primara si Oras drept cheie straina.

f) ---creating the tables

```
CREATE TABLE Category(
Category_Id NUMBER GENERATED BY DEFAULT AS IDENTITY,
Category_Name VARCHAR2(50) NOT NULL,
PRIMARY KEY(Category_Id)
);

CREATE TABLE Format(
Format_Id NUMBER GENERATED BY DEFAULT AS IDENTITY,
Cover_Type VARCHAR2(100) NOT NULL,
PRIMARY KEY(Format_Id)
);

CREATE TABLE Book(
Book_Id NUMBER GENERATED BY DEFAULT AS IDENTITY,
PRIMARY KEY(Book_Id),
Title VARCHAR2(200) NOT NULL,
Description VARCHAR2(200),
No_Of_Pages NUMBER(4),
Price FLOAT(5),
Category_Id NUMBER,
Format_Id NUMBER,
FOREIGN KEY (Category_Id) REFERENCES Category(Category_Id),
FOREIGN KEY (Format_Id) REFERENCES Format(Format_Id)
);

CREATE TABLE City(
City_Id NUMBER GENERATED BY DEFAULT AS IDENTITY,
```

```
PRIMARY KEY(City_Id),
City_Name VARCHAR(100) NOT NULL
);

CREATE TABLE Country(
Country_Id NUMBER GENERATED BY DEFAULT AS IDENTITY,
PRIMARY KEY(Country_Id),
Country_Name VARCHAR(100) NOT NULL
);

CREATE TABLE Address(
Address_Id NUMBER GENERATED BY DEFAULT AS IDENTITY,
PRIMARY KEY(Address_Id),
Street VARCHAR(100),
Street_Number NUMBER,
City_Id NUMBER,
FOREIGN KEY (City_Id) REFERENCES City(City_Id),
Country_Id NUMBER,
FOREIGN KEY (Country_Id) REFERENCES Country(Country_Id)
);

CREATE TABLE Author(
Author_Id NUMBER GENERATED BY DEFAULT AS IDENTITY,
PRIMARY KEY(Author_Id),
Author_Name VARCHAR(100) NOT NULL,
Date_of_Birth DATE,
Address_Id NUMBER,
FOREIGN KEY (Address_Id) REFERENCES Address(Address_Id)
);

CREATE TABLE Writes(
Author_Id NUMBER,
FOREIGN KEY (Author_Id) REFERENCES Author(Author_Id),
Book_Id NUMBER,
FOREIGN KEY (Book_Id) REFERENCES Book(Book_Id),
PRIMARY KEY (Author_Id, Book_Id),
Publication_Year DATE,
Delay NUMBER
);

--- inserting into tables
-- category
INSERT INTO Category(Category_Name)
VALUES ('Fantasy');
INSERT INTO Category(Category_Name)
VALUES ('Romance');
INSERT INTO Category(Category_Name)
VALUES('Comedy');
```

```
INSERT INTO Category(Category_Name)
VALUES('Personal Development');
INSERT INTO Category(Category_Name)
VALUES('History');

--format
INSERT INTO Format(Cover_Type)
VALUES ('Paperback');
INSERT INTO Format(Cover_Type)
VALUES ('Hardback');
INSERT INTO Format(Cover_Type)
VALUES ('Collector Edition');
INSERT INTO Format(Cover_Type)
VALUES ('Leather Bound');
INSERT INTO Format(Cover_Type)
VALUES ('Digital');

--book
INSERT INTO Book(Title, Description, No_Of_Pages, Price, Category_Id, Format_Id )
VALUES ('Harry Potter','Harry Potter goes to Hogwarts', '256', '54', '1', '3');
INSERT INTO Book(Title, Description, No_Of_Pages, Price, Category_Id, Format_Id )
VALUES ('Me Before You','Romantic description for a romantic book', '457', '43', '2', '2');
INSERT INTO Book(Title, Description, No_Of_Pages, Price, Category_Id, Format_Id )
VALUES ('Best Jokes','Jokes for everyone', '133', '23', '3', '5');
INSERT INTO Book(Title, Description, No_Of_Pages, Price, Category_Id, Format_Id )
VALUES ('Good Habbits','Good habbits for a good life', '121', '32', '4', '1');
INSERT INTO Book(Title, Description, No_Of_Pages, Price, Category_Id, Format_Id )
VALUES ('WW2','WW2 - A short history', '501', '134', '5', '4');
INSERT INTO Book(Title, Description, No_Of_Pages, Price, Category_Id, Format_Id )
VALUES ('WW1','WW1 - A short history', NULL, '150', '5', '4');
INSERT INTO Book(Title, Description, No_Of_Pages, Price, Category_Id, Format_Id )
VALUES ('Gripa Spaniola',' A short history', 89, 89, 5, 4);

--city
INSERT INTO City(City_Name)
VALUES ('London');
INSERT INTO City(City_Name)
VALUES ('Paris');
INSERT INTO City(City_Name)
VALUES ('Bucharest');
INSERT INTO City(City_Name)
VALUES ('Berlin');
INSERT INTO City(City_Name)
VALUES ('Tokyo');

--country
INSERT INTO Country(Country_Name)
VALUES ('United Kingdom');
```

```
INSERT INTO Country(Country_Name)
VALUES ('France');
INSERT INTO Country(Country_Name)
VALUES ('Romania');
INSERT INTO Country(Country_Name)
VALUES ('Germany');
INSERT INTO Country(Country_Name)
VALUES ('Japan');

--address
INSERT INTO Address(Street, Street_Number, City_Id, Country_Id)
VALUES ('Elm','12','1','1');
INSERT INTO Address(Street, Street_Number, City_Id, Country_Id)
VALUES ('Rue Belle','121','2','2');
INSERT INTO Address(Street, Street_Number, City_Id, Country_Id)
VALUES ('Plopilor','25','3','3');
INSERT INTO Address(Street, Street_Number, City_Id, Country_Id)
VALUES ('Schk','53','4','4');
INSERT INTO Address(Street, Street_Number, City_Id, Country_Id)
VALUES ('Kong','156','5','5');

--author
INSERT INTO Author(Author_Name, Date_Of_Birth, Address_Id)
VALUES ('JK Rowling','20-jul-1977',1);
INSERT INTO Author(Author_Name, Date_Of_Birth, Address_Id)
VALUES ('Suzzie K','18-jan-1957',4);
INSERT INTO Author(Author_Name, Date_Of_Birth, Address_Id)
VALUES ('Harper JW','15-jul-1975',3);
INSERT INTO Author(Author_Name, Date_Of_Birth, Address_Id)
VALUES ('Harry Bay','20-jun-1986',2);
INSERT INTO Author(Author_Name, Date_Of_Birth, Address_Id)
VALUES ('Karl F','12-aug-1967',5);

--writes
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (1,1,'12-aug-1997');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (1,3,'15-jul-2013');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (3,3,'15-jul-2013');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (4,3,'15-jul-2013');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (2,2,'18-jan-1999');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (5,5,'24-nov-2018');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (4,5,'24-nov-2018');
```

```
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (3,4,'10-feb-1999');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (2,4,'10-feb-1999');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year)
VALUES (1,4,'10-feb-1999');
INSERT INTO Writes(Author_Id, Book_Id, Publication_Year, Delay)
VALUES (4,6,'24-dec-2016',3);

g) --- QUERIES
-- 1. Selecteaza toti autorii care au scris cel putin doua carti.
SELECT a.author_id, a.author_name
FROM author a
WHERE a.author_id IN ( SELECT w.author_id
                       FROM writes w
                       GROUP BY w.author_id
                       HAVING COUNT(w.author_id) >= 2 );

-- 2. Afiseaza pretul mediu al cartilor publicate intre anii 2000 si 2020.
SELECT AVG(b.price) AS Average_price
FROM book b
WHERE b.book_id IN (SELECT w.book_id
                    FROM writes w
                    WHERE w.publication_year between to_date( '01-jan-2000 00:00:00', 'dd-mon-
yyyy hh24:mi:ss' )
                                        and to_date( '01-jan-2020 00:00:00', 'dd-mon-yyyy hh24:mi:ss' ));

-- 3. Afiseaza toate cartile care sunt history sau sunt editie de colectie, cu tipul de coperta si
categoria in litere mici.
SELECT c.category_name, b.title, f.cover_type
FROM book b RIGHT JOIN category c ON c.category_id = b.category_id
          LEFT JOIN format f ON b.format_id = f.format_id
WHERE lower(f.cover_type) = to_char('collector edition') OR lower(c.category_name) =
'history';

-- 4. Returneaza cartile care au pretul sub 70 de lei si afiseaza in romana formatul (in loc de
cel in engleza).
SELECT b.title, b.price,
   DECODE(f.format_id, 1, 'Coperta de hartie',
                       2, 'Coperta cartonata',
                       3, 'Editie de colectie',
                       4, 'Editie de piele',
                       5, 'Digital',
                         'Non domestic')  "Tip coperta"
   FROM book b INNER JOIN format f ON b.format_id = f.format_id
   WHERE b.price < 70;
```

```
--5. Afiseaza titlurile cartilor (cu upper letters) care au cel putin 200 de pagini si nu sunt digitale, ordonate dupa pret, crescator.
SELECT UPPER(b.title), b.no_of_pages, b.price, f.cover_type
FROM book b FULL JOIN format f ON f.format_id = b.format_id
WHERE b.no_of_pages >= 200 AND upper(f.cover_type) <> TO_CHAR('DIGITAL')
ORDER BY b.price;

-- 6. Afiseaza suma preturilor cartilor scrise de JK Rowling.
SELECT SUM(b.price) AS SUMA_PRETURILOR
FROM book b INNER JOIN writes w ON b.book_id = w.book_id INNER JOIN author a ON a.author_id = w.author_id
WHERE UPPER(a.author_name) = UPPER(TO_CHAR('JK Rowling'));

-- 7. Afiseaza toate cartile de istorie, cu titlu, descriere, numar de pagini si pret. Daca nu se stie numarul de pagini, se va afisa 'Necunoscut'.
SELECT b.title, b.description, NVL(TO_CHAR(b.no_of_pages), 'Necunoscut'), b.price
FROM book b FULL JOIN category c ON c.category_id = b.category_id
WHERE lower(c.category_name) = 'history';

--8. Afiseaza toate cartile cu titlu, descriere, pret si numar de pagini. Daca au acelasi pret cu numarul de pagini, se va afisa in coloana de pret
--'Acelasi ca pretul'.
SELECT
  book_id, title, description, price,
  COALESCE(TO_CHAR(NULLIF(no_of_pages, price)), 'Acelasi ca pretul')numar_pagini
FROM
  book
WHERE
  no_of_pages IS NOT NULL;

-- 9.Afiseaza toate cartile de istorie (titlu, pret, numar de pagini). Daca au acelasi pret cu numarul de pagini, se va afisa in coloana de pret
--'Acelasi ca pretul'.

SELECT
  b.title, b.price,
  CASE
    WHEN b.no_of_pages = b.price
    THEN 'Acelasi ca pretul'
    ELSE to_char(b.no_of_pages)
  END
FROM
  book b INNER JOIN category c ON C.category_id = b.category_id
WHERE
  no_of_pages IS NOT NULL AND lower(c.category_name) = 'history';

-- 10. Afiseaza pentru fiecare autor si carte, diferenta de luni intre data publicarii cartii si data nasterii autorului.
```

```
SELECT a.author_name, b.title, MONTHS_BETWEEN
(TO_DATE(w.publication_year),TO_DATE(a.date_of_birth) ) "Months"
FROM author a INNER JOIN writes w ON w.author_id = a.author_id INNER JOIN book b ON
w.book_id = b.book_id
ORDER BY (a.author_name);

--11. Afiseaza adevarata data de publicare pentru cartile care au avut delay.
SELECT b.title, TO_CHAR( ADD_MONTHS(w.publication_year,w.delay),'DD-MON-YYYY') "Real
publication date"
FROM book b INNER JOIN writes w ON b.book_id = w.book_id;

--12. Afiseaza titlurile prescurtate la primele 3 caractere pentru toate cartile.
SELECT SUBSTR(to_char(b.title),1,3) "Prescurtare"
FROM book b;

-- 13. Afiseaza pretul maxim si pretul minim dintre toate cartile care nu au avut intarzieri la
publicare.
SELECT MAX(price) as maximum_price, MIN(price) as minimum_price
FROM book b
WHERE b.book_id NOT IN (SELECT w.book_id
                FROM writes w
                WHERE w.delay = null);

-- 14. Afiseaza pentru fiecare titlu pozitia pe care apare prima data secventa 'ww'
SELECT INSTR(lower(title),'ww', 1, 1) "Instring"
FROM book;

-- 15. Afiseaza autorii care locuiesc in Romania sau in UK.
SELECT a.author_name, c.country_name
FROM author a INNER JOIN address ad ON a.address_id = ad.address_id INNER JOIN country
c ON c.country_id = ad.country_id
WHERE lower(c.country_name) = 'united kingdom' OR lower(c.country_name)='romania';

h) -- Crearea tabelului de mesaje:

CREATE TABLE MESSAGES(
   MESSAGE_ID NUMBER ,
   PRIMARY KEY(MESSAGE_ID),
   MESSAGE VARCHAR2(255),
   MESSAGE_TYPE VARCHAR2(1) CHECK (MESSAGE_TYPE IN ('E','W','I')),
   CREATED_BY VARCHAR2(40) NOT NULL,
   CREATED_AT DATE NOT NULL
);

j) --creaza o secventa pentru a genera id-ul pentru tabelul de mesaje

CREATE SEQUENCE message_seq
 START WITH     1
```

```
 INCREMENT BY   1
 NOCACHE
 NOCYCLE
 NOMAXVALUE;

i)
-- 1. Subprogram stocat care sa foloseasca doua tipuri de colectii
-- Sa se afiseze pentru fiecare autor suma tuturor cartilor publicate

CREATE OR REPLACE PROCEDURE SUMA_CARTI
IS
BEGIN
   DECLARE
   TYPE autori_arr IS TABLE OF author.author_id%TYPE
   INDEX BY BINARY_INTEGER;
   TYPE suma_preturi IS VARRAY(100) OF NUMBER (10);
   v_suma_preturi suma_preturi;
   v_autori_arr autori_arr := autori_arr();
   v_suma NUMBER;
   BEGIN
   SELECT author_id
   BULK COLLECT INTO v_autori_arr
   FROM author;
   FOR i IN v_autori_arr.FIRST..v_autori_arr.LAST LOOP
      v_suma := 0;
      SELECT b.price
      BULK COLLECT INTO v_suma_preturi
      FROM book b INNER JOIN writes w ON b.book_id = w.book_id
               INNER JOIN author a ON w.author_id = a.author_id
      WHERE a.author_id = v_autori_arr(i);

      FOR j IN v_suma_preturi.FIRST..v_suma_preturi.LAST LOOP
         v_suma := v_suma + v_suma_preturi(j);
      END LOOP;
      DBMS_OUTPUT.PUT_LINE('Autorul CU ID-ul  ' || v_autori_arr(i) || ' are suma preturilor
cartilor egala cu ' || v_suma);
   END LOOP;
   END;
END;
/

BEGIN
SUMA_CARTI;
END;
/

– 2. Subprogram stocat cu un cursor
-- Afseaza pentru fiecare categorie existenta numarul de carti ce ii apartin.
```

```
CREATE OR REPLACE PROCEDURE CATEGORY_BOOK IS
BEGIN
DECLARE
   CURSOR c1 IS
   SELECT category_name category, COUNT(book_id) nr_carti
   FROM category c, book b
   WHERE c.category_id = b.category_id
   GROUP BY category_name;
   BEGIN
      FOR i IN c1 LOOP
      IF i.nr_carti = 0 THEN
         DBMS_OUTPUT.PUT_LINE ('Categoria ' || i.category || ' nu contine carti.');
      ELSIF i.nr_carti = 1  THEN
         DBMS_OUTPUT.PUT_LINE ('Categoria ' || i.category || ' contine o carte.');
      ELSE
         DBMS_OUTPUT.PUT_LINE ('Categoria ' || i.category || ' contine ' || i.nr_carti || '
carti.');
      END IF;
      END LOOP;
   END;
END CATEGORY_BOOK;
/

SET SERVEROUTPUT ON;

BEGIN
CATEGORY_BOOK;
END;
/

-- 3. Subprogram stocat de tip functie in care se folosesc 3 tabele
-- sa se afiseze cartea scrisa de un autor al carui nume este dat de user

CREATE OR REPLACE FUNCTION BOOK_AUTHOR (nume_autor author.author_name%TYPE)
RETURN book.title%TYPE
AS
   titlu varchar(200);
BEGIN
   SELECT title INTO titlu
   FROM book b INNER JOIN writes w ON b.book_id = w. book_id
            INNER JOIN author a ON a.author_id = w.author_id
   WHERE lower(a.author_name) = lower(nume_autor);
   RETURN titlu;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      INSERT INTO
MESSAGES(MESSAGE_ID,MESSAGE,MESSAGE_TYPE,CREATED_BY,CREATED_AT) VALUES
(message_seq.nextval, 'Nu exista autor cu acest nume.', 'E', 'BOOK_AUTHOR',SYSDATE);
```

```
      --RAISE_APPLICATION_ERROR(-20201, 'Nu exista autor cu acest nume.');
      RETURN NULL;
   WHEN TOO_MANY_ROWS THEN
      INSERT INTO
MESSAGES(MESSAGE_ID,MESSAGE,MESSAGE_TYPE,CREATED_BY,CREATED_AT) VALUES
(message_seq.nextval,'Exista mai multe carti scrise de acest autor.', 'E',
'BOOK_AUTHOR',SYSDATE);
      --RAISE_APPLICATION_ERROR(-20201, 'Exista mai multe carti scrise de acest autor.');
      RETURN NULL;
END BOOK_AUTHOR;
/

-- cazul functional
BEGIN
   DBMS_OUTPUT.PUT_LINE('Autorul a scris cartea: ' || BOOK_AUTHOR('karl f'));
END;
/

-- eroare 'Nu exista autor cu acest nume.' (va fi inserata in tabel, nu afisata pee cran)
BEGIN
   DBMS_OUTPUT.PUT_LINE('Autorul a scris cartea: ' || BOOK_AUTHOR('carl f'));
END;
/
SELECT * FROM MESSAGES;

-- eroare 'Exista mai multe carti scrise de acest autor.' (va fi inserata in tabel, nu afisata pee
cran)
BEGIN
   DBMS_OUTPUT.PUT_LINE('Autorul a scris cartea: ' || BOOK_AUTHOR('jk rowling'));
END;
/
SELECT * FROM MESSAGES;

--4. Sa se creeze un trigger care se asigura ca nu se pot face modificari pe tabelul book in
weekend.
CREATE OR REPLACE TRIGGER BOOK_TRIGGER
BEFORE INSERT OR UPDATE OR DELETE ON book
BEGIN
   IF (TO_CHAR(SYSDATE,'dy') IN ('sat','sun'))
   THEN
      RAISE_APPLICATION_ERROR(-20500,'In weekend nu se pot face modificari pe tabelul
book');
   ELSE
      INSERT INTO MESSAGES(MESSAGE_ID,
MESSAGE,MESSAGE_TYPE,CREATED_BY,CREATED_AT) VALUES (message_seq.nextval, 'Book
a fost modificat', 'I', 'BOOK_TRIGGER',SYSDATE);
   END IF;
END;
```

```
/

-- declansare trigger
UPDATE book SET title = 'trigger_test' WHERE book_id = 5;
SELECT * FROM MESSAGES;

-- 5. Creeaza un trigger care sa se asigure ca orice valoare a pretului si a numarului de pagini
din books este pozitiva.
CREATE OR REPLACE TRIGGER PRICE_PAGES_TRIGGER
BEFORE INSERT OR UPDATE ON book
FOR EACH ROW
BEGIN
    IF(:NEW.price <= 0 OR :NEW.no_of_pages <=0) THEN
      RAISE_APPLICATION_ERROR (-20201, 'Paginile si pretul trebuie sa fie mai mari ca 0.');
   END IF;
END;
/

-- declansare trigger
-- cu eroare
UPDATE book SET price = -1 WHERE book_id = 1;
-- fara eroare
UPDATE book SET price = 205 WHERE book_id = 5;

-- 6. Creeaza un trigger care sa salveze erorile in tabelul de mesaje dupa fiecare eroare de
server
CREATE OR REPLACE TRIGGER SAVE_ERRORS
AFTER SERVERERROR ON DATABASE
BEGIN
   INSERT INTO
MESSAGES(MESSAGE_ID,MESSAGE,MESSAGE_TYPE,CREATED_BY,CREATED_AT) VALUES
(message_seq.nextval, SYS.server_error_msg(1), 'E', 'SAVE_ERRORS',SYSDATE);
END;


-- Declansare trigger: provocare eroare server prin impartirea la 0

SELECT 1/0 FROM DUAL;
SELECT * FROM MESSAGES;

--7. Pachet

CREATE OR REPLACE PACKAGE TOATE_PROCEDURILE IS
   PROCEDURE SUMA_CARTI;
   PROCEDURE CATEGORY_BOOK;
   FUNCTION BOOK_AUTHOR (nume_autor author.author_name%TYPE)
         RETURN book.title%TYPE;

   TYPE autori_arr IS TABLE OF author.author_id%TYPE
```

```
   INDEX BY BINARY_INTEGER;
   TYPE suma_preturi IS VARRAY(100) OF NUMBER (10);

   CURSOR c1 IS
   SELECT category_name category, COUNT(book_id) nr_carti
   FROM category c, book b
   WHERE c.category_id = b.category_id
   GROUP BY category_name;


END TOATE_PROCEDURILE;
/
```