**Temă la PSGBD**

**Ciobanu Sebastian – Adrian, 2A5**

**Băetu Ciprian, 2A5**

Aplicația aleasă este cea a unei librării online cu cărți gratuite (în genul proiectului Gutenberg: <https://www.gutenberg.org/>). De aceea, vom avea nevoie de o bază de date care să conțină informații legate de autori, opere, cărți, genuri, subgenuri.

Fiecărui autor i se vor asocia:

* numele
* prenumele

Fiecărei opere i se vor asocia:

* un număr unic de identificare
* titlul
* unul sau mai mulți autori
* numărul volumului, în caz că există
* unul sau mai multe genuri în care se încadrează
* unul sau mai multe subgenuri în care se încadrează

Fiecărei cărți i se vor asocia:

* opera
* ISBN-ul care identifică unic cartea (1)
* editura la care a apărut
* anul apariției
* ratingul dat de vizitatorii librăriei, în caz că există
* calea către fișierul stocat pe diskul serverului sau stocat pe Web, ce servește la descărcarea cărții

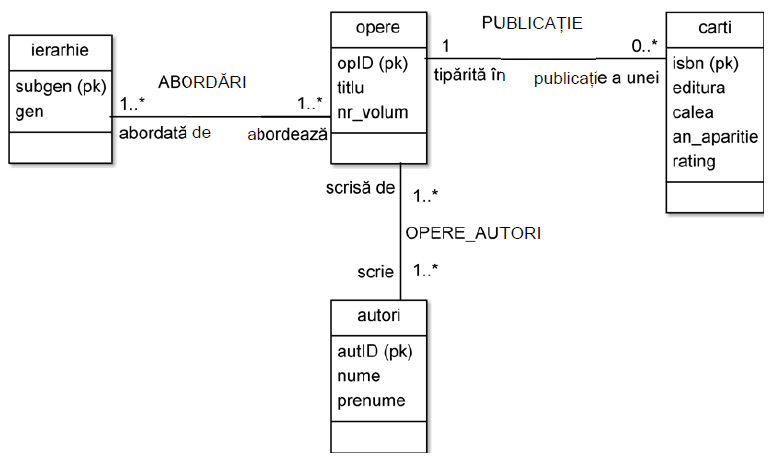
Va exista o ierarhie de genuri și subgenuri.

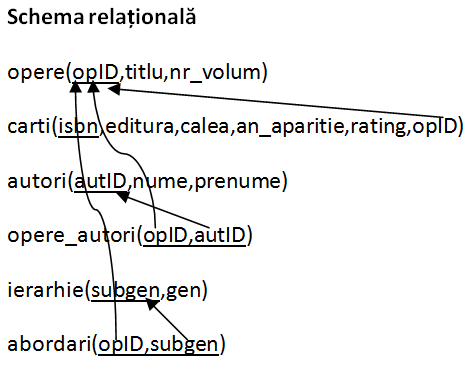
Un subgen va aparține sigur unui singur gen. (2)

Fiecare gen poate avea asociat unul sau mai multe subgenuri.

Dacă o carte are asociată o operă cu un anumit autor / un anumit gen / un anumit subgen, atunci toate cărțile care au asociate acea operă vor avea asociat acel autor/ acel gen/ acel subgen. (3)

**Diagrama ERD în notație UML:**



****

Am adăugat atributul autID pentru entitatea autori cu scopul de a avea o cheie formată dintr-un singur atribut (pentru ca entitățile ce vor prelua cheia primară drept cheie străină – opere\_autori – să preia doar un atribut).

Tipurile de date ale atributelor sunt:

* opID – număr întreg mai mare decât 0
* titlu – șir de maxim 100 de caractere
* nr\_volum – număr întreg mai mare decât 0
* ISBN – șir de maxim 13 caractere (de lungime 10 sau 13)
* editura – șir de maxim 50 de caractere
* calea – șir de maxim 100 de caractere
* an\_apariție – număr întreg mai mare decât 0
* rating - număr întreg mai mare decât 0 (mai mic decât sau egal cu 10)
* autID – număr întreg mai mare decât 0
* nume – șir de maxim 50 de caractere
* prenume – șir de maxim 50 de caractere
* subgen – șir de maxim 50 de caractere
* gen – șir de maxim 50 de caractere

În plus, pentru gestionarea userilor, a recomandărilor pentru aceștia (o carte sau 2) și a unor citate random ce vor apărea pe pagina principală, avem următoarele tabele:

* utilizator\_parola(utilizator,parola)
* citate(id,text,autor)
* distante\_utilizatori(u1,u2,distanta)
* utilizatori\_carte\_rating(utilizator,isbn,rating)

Pachetele folosite sunt:

* excepții: o serie de excepții predefinite –excepție,cod,mesaj- pentru a fi utilizate global la nivelul aplicației (evităm astfel duplicarea codului)
* CRUD: procedurile necesare pentru operații CRUD
  + insereaza\_carte
  + insereaza\_autor
  + insereaza\_opera
  + insereaza\_subgen
  + sterge\_carte
  + sterge\_autor
  + sterge\_opera
  + sterge\_subgen
* import\_export: pentru importul și exportul BD în format csv
  + export\_database
  + import\_BD
* gestioneaza\_carte: pentru gestionarea cărților
  + carte\_random
  + rate
  + afiseaza\_carte
  + afiseaza\_opera
* logica\_aplicației
  + login
  + citat\_random
* recomandare
  + recomandă cărți
* paginare
  + paginare\_carti\_next
  + paginare\_carti\_back
  + paginare\_opere\_next
  + paginare\_opere\_back
  + paginare\_autori\_next
  + paginare\_autori\_back

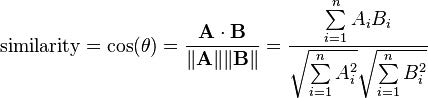
Pentru eficientizarea interogărilor și în special a paginărilor am folosit următorii **indecși** denumiți sugestiv:

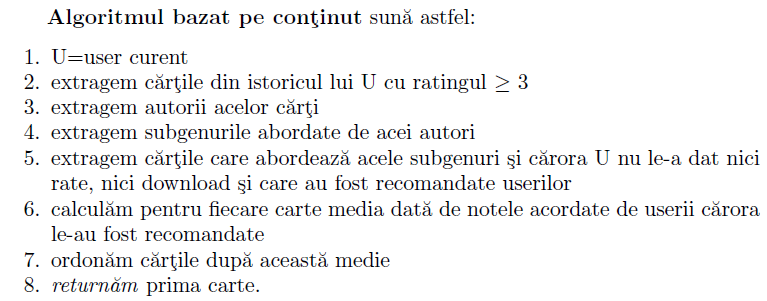
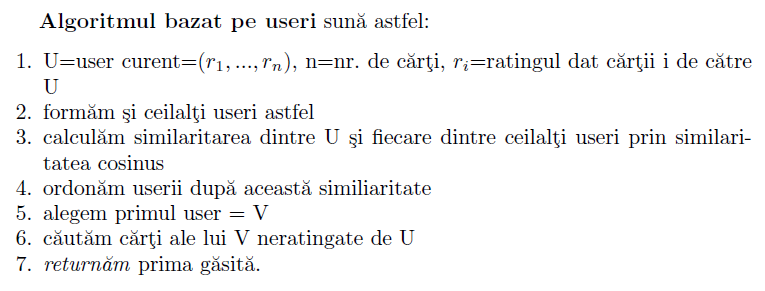
* nume\_prenume\_autor
* titlu\_volum
* rating\_isbn
* isbn\_opID

**Triggerele** folosite au fost:

* actualizeaza\_distante: actualizează distanțele între useri după notarea unei cărți
* insereaza\_autor\_distante: insereaă userul în tabela distanțe după inserarea unui utilizator
* medie\_rating: recalculează nota unei cărți după ce i-a fost dat un nou rating.

Sistemul de recomandări este descris în continuare.





Codul utilizat este următorul:

* Crearea tabelelor:

DROP TABLE opere\_autori;

/

DROP TABLE abordari;

/

DROP TABLE utilizator\_carte\_rating;

/

DROP TABLE carti;

/

DROP TABLE ierarhie;

/

DROP TABLE autori;

/

DROP TABLE opere;

/

DROP TABLE distante\_utilizatori;

/

DROP TABLE utilizator\_parola;

/

DROP TABLE citate;

/

CREATE TABLE opere (

opID INTEGER PRIMARY KEY CHECK(opID > 0),

titlu VARCHAR2(1000) NOT NULL,

volum INTEGER CHECK(volum > 0)

);

/

CREATE TABLE carti (

ISBN NUMBER PRIMARY KEY,

editura VARCHAR2(50) NOT NULL,

calea VARCHAR2(200),

an\_aparitie INTEGER NOT NULL,

rating NUMBER CHECK(rating >= 1 AND rating <= 10),

nr\_rating NUMBER CHECK(nr\_rating >= 0),

opID INTEGER REFERENCES opere(opID) ON DELETE CASCADE NOT NULL

);

/

CREATE TABLE autori (

autID INTEGER PRIMARY KEY,

nume VARCHAR2(200),

prenume VARCHAR2(200)

);

/

CREATE TABLE opere\_autori (

opID INTEGER REFERENCES opere(opID) ON DELETE CASCADE NOT NULL,

autID INTEGER REFERENCES autori(autID) ON DELETE CASCADE NOT NULL,

PRIMARY KEY (opID, autID)

);

/

CREATE TABLE ierarhie (

subgen VARCHAR2(50) PRIMARY KEY,

gen VARCHAR2(50) NOT NULL

);

/

CREATE TABLE abordari (

opID INTEGER REFERENCES opere(opID) ON DELETE CASCADE NOT NULL,

subgen VARCHAR2(50) REFERENCES ierarhie(subgen) ON DELETE CASCADE NOT NULL,

PRIMARY KEY (opID, subgen)

);

/

CREATE TABLE utilizator\_parola (

utilizator VARCHAR2(100) PRIMARY KEY,

parola VARCHAR2(100)

);

/

CREATE TABLE citate

( id NUMBER PRIMARY KEY,

text VARCHAR2(4000),

autor varchar2(1000)

);

CREATE TABLE distante\_utilizatori (

u1 VARCHAR2(100) REFERENCES utilizator\_parola(utilizator) ON DELETE CASCADE,

u2 VARCHAR2(100) REFERENCES utilizator\_parola(utilizator) ON DELETE CASCADE,

distanta NUMBER,

PRIMARY KEY (u1, u2)

);

/

create table utilizator\_carte\_rating (

utilizator varchar2(100) references utilizator\_parola(utilizator),

isbn number references carti(isbn),

rating number check(rating>=1 and rating<=10) not null,

primary key(utilizator,isbn)

);

/

* Pachetul exceptii:

CREATE OR REPLACE PACKAGE exceptii IS

constrangere EXCEPTION;

constrangere\_nr number := -20020;

constrangere\_text varchar2(1000) :='Eroare! V? rug?m s? verifica?i valoarea introdus? pentru ';

PRAGMA EXCEPTION\_INIT(constrangere, -2290);

cheie\_straina\_gresita EXCEPTION;

cheie\_straina\_gresita\_nr number := -20019;

pragma exception\_init(cheie\_straina\_gresita, -2291);

subgen\_deja\_existent EXCEPTION;

subgen\_deja\_existent\_nr number := -20006;

subgen\_deja\_existent\_text varchar2(1000) :='Eroare la introducere subgen! Subgenul exist? deja.';

PRAGMA EXCEPTION\_INIT(subgen\_deja\_existent, -20006);

isbn\_inexistent EXCEPTION;

isbn\_inexistent\_nr number := -20009;

isbn\_inexistent\_text varchar2(1000) := 'Eroare! V? rug?m s? verifica?i ISBN-ul.';

PRAGMA EXCEPTION\_INIT(isbn\_inexistent, -20009);

autor\_inexistent EXCEPTION;

autor\_inexistent\_nr number := -20011;

autor\_inexistent\_text varchar2(1000) :='Eroare la ?tergere autor! V? rug?m s? verifica?i ID-ul autorului.';

PRAGMA EXCEPTION\_INIT(autor\_inexistent, -20011);

opera\_inexistenta EXCEPTION;

opera\_inexistenta\_nr number := -20013;

opera\_inexistenta\_text varchar2(1000) := 'Eroare la ?tergere oper?! V? rug?m s? verifica?i ID-ul operei.';

PRAGMA EXCEPTION\_INIT(opera\_inexistenta, -20013);

subgen\_inexistent EXCEPTION;

subgen\_inexistent\_nr number := -20015;

subgen\_inexistent\_text varchar2(1000) :='Eroare la ?tergere subgen! V? rug?m s? verifica?i subgenul.';

PRAGMA EXCEPTION\_INIT(subgen\_inexistent, -20015);

END;

/

* Pachetul CRUD:

CREATE OR REPLACE PACKAGE CRUD IS

PROCEDURE insereaza\_carte(

p\_titlu IN opere.titlu%TYPE,

p\_volum IN opere.volum%TYPE,

p\_isbn IN carti.isbn%TYPE,

p\_editura IN carti.editura%TYPE,

p\_calea IN carti.calea%TYPE,

p\_an\_aparitie IN carti.an\_aparitie%TYPE,

p\_subgen IN ierarhie.subgen%TYPE,

p\_nume IN autori.nume%type,

p\_prenume IN autori.prenume%type);

PROCEDURE insereaza\_autor

(

p\_nume IN autori.nume%type,

p\_prenume IN autori.prenume%type

);

PROCEDURE insereaza\_subgen

(

p\_subgen IN ierarhie.subgen%type,

p\_gen IN ierarhie.gen%type

);

PROCEDURE insereaza\_opera

(

p\_titlu opere.titlu%type,

p\_volum opere.volum%type,

p\_nume autori.nume%type,

p\_prenume autori.prenume%type

);

PROCEDURE sterge\_carte

(

p\_isbn carti.isbn%type

);

PROCEDURE sterge\_autor(

p\_autID autori.autID%type);

PROCEDURE sterge\_opera(

p\_opID opere.opID%type);

PROCEDURE sterge\_subgen(

p\_subgen ierarhie.subgen%type,

p\_mesaj\_succes out varchar2);

END CRUD;

/

---------------------------- INSERTURI ----------------------------------------

CREATE OR REPLACE PACKAGE BODY CRUD AS

PROCEDURE insereaza\_carte(

p\_titlu IN opere.titlu%TYPE,

p\_volum IN opere.volum%TYPE,

p\_isbn IN carti.isbn%TYPE,

p\_editura IN carti.editura%TYPE,

p\_calea IN carti.calea%TYPE,

p\_an\_aparitie IN carti.an\_aparitie%TYPE,

p\_subgen IN ierarhie.subgen%TYPE,

p\_nume IN autori.nume%type,

p\_prenume IN autori.prenume%type)

IS

v\_opID opere.opID%TYPE;

v\_autID autori.autID%TYPE;

v\_gen ierarhie.gen%type;

BEGIN

SAVEPOINT inainte;

SELECT MAX(opID)+1 INTO v\_opID FROM opere;

SELECT MAX(autID)+1 INTO v\_autID FROM autori;

SELECT gen INTO v\_gen FROM ierarhie WHERE subgen=p\_subgen;

INSERT INTO opere VALUES

(v\_opID,p\_titlu,p\_volum

);

INSERT INTO autori VALUES

(v\_autID,p\_nume,p\_prenume

);

INSERT INTO opere\_autori VALUES

(v\_opID,v\_autID

);

INSERT

INTO carti VALUES

(

p\_isbn,

p\_editura,

p\_calea,

p\_an\_aparitie,

NULL,

0,

v\_opID

);

INSERT INTO abordari VALUES

(v\_opID,p\_subgen

);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

ROLLBACK TO inainte;

raise\_application\_error(-20001,'Eroare la introducere carte! V? rug?m verifica?i ISBN-ul.');

WHEN exceptii.cheie\_straina\_gresita THEN

ROLLBACK TO inainte;

raise\_application\_error(exceptii.cheie\_straina\_gresita\_nr,'Eroare la introducere carte! V? rug?m verifica?i subgenul.');

WHEN NO\_DATA\_FOUND THEN

ROLLBACK TO inainte;

raise\_application\_error(-20003,'Eroare la introducere carte! V? rug?m verifica?i subgenul.');

WHEN OTHERS THEN

ROLLBACK TO inainte;

raise\_application\_error(-20004,'Eroare la introducere carte! V? rug?m reîncerca?i.');

END insereaza\_carte;

PROCEDURE insereaza\_autor

(

p\_nume IN autori.nume%type,

p\_prenume IN autori.prenume%type

)

IS

v\_autID autori.autID%type;

BEGIN

SELECT MAX(autID)+1 INTO v\_autID FROM autori;

INSERT INTO autori VALUES

(v\_autID,p\_nume,p\_prenume

);

EXCEPTION

WHEN OTHERS THEN

raise\_application\_error(-20005,'Eroare la introducere autor! V? rug?m verifica?i autorul.');

END insereaza\_autor;

PROCEDURE insereaza\_subgen

(

p\_subgen IN ierarhie.subgen%type,

p\_gen IN ierarhie.gen%type

)

IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM ierarhie WHERE subgen=p\_subgen;

IF (v\_count != 0) THEN

raise exceptii.subgen\_deja\_existent;

END IF;

INSERT INTO ierarhie VALUES

(p\_subgen,p\_gen

);

EXCEPTION

WHEN exceptii.subgen\_deja\_existent THEN

raise\_application\_error(exceptii.subgen\_deja\_existent\_nr,exceptii.subgen\_deja\_existent\_text);

WHEN OTHERS THEN

raise\_application\_error(-20007,'Eroare la introducere subgen! V? rug?m verifica?i (sub)genul.');

END insereaza\_subgen;

PROCEDURE insereaza\_opera

(

p\_titlu opere.titlu%type,

p\_volum opere.volum%type,

p\_nume autori.nume%type,

p\_prenume autori.prenume%type

)

IS

v\_opID opere.opID%type;

v\_autID autori.autID%type;

BEGIN

SELECT MAX(opID)+1 INTO v\_opID FROM opere;

SELECT MAX(autID)+1 INTO v\_autID FROM autori;

INSERT INTO opere VALUES

(v\_opID,p\_titlu,p\_volum

);

INSERT INTO autori VALUES

(v\_autID,p\_nume,p\_prenume

);

INSERT INTO opere\_autori VALUES

(v\_opID,v\_autID

);

EXCEPTION

WHEN OTHERS THEN

raise\_application\_error(-20008,'Eroare la introducere opera! V? rug?m încerca?i din nou.');

END insereaza\_opera;

---------------------------- DELETE-URI ----------------------------------------

PROCEDURE sterge\_carte

(

p\_isbn carti.isbn%type

)

IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM carti WHERE p\_isbn=isbn;

IF v\_count != 1 THEN

raise exceptii.isbn\_inexistent;

END IF;

DELETE FROM carti WHERE p\_isbn=isbn;

EXCEPTION

WHEN exceptii.isbn\_inexistent THEN

raise\_application\_error(exceptii.isbn\_inexistent\_nr,exceptii.isbn\_inexistent\_text);

WHEN OTHERS THEN

raise\_application\_error(-20010,'Eroare la ?tergere carte! V? rug?m s? încerca?i din nou.');

END sterge\_carte;

PROCEDURE sterge\_autor(

p\_autID autori.autID%type)

IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM autori WHERE p\_autID=autID;

IF v\_count != 1 THEN

raise exceptii.autor\_inexistent;

END IF;

DELETE FROM autori WHERE p\_autID=autID;

EXCEPTION

WHEN exceptii.autor\_inexistent THEN

raise\_application\_error(exceptii.autor\_inexistent\_nr,exceptii.autor\_inexistent\_text);

WHEN OTHERS THEN

raise\_application\_error(-20012,'Eroare la ?tergere autor! V? rug?m s? încerca?i din nou.');

END sterge\_autor;

PROCEDURE sterge\_opera(

p\_opID opere.opID%type)

IS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM opere WHERE p\_opID=opID;

IF v\_count != 1 THEN

raise exceptii.opera\_inexistenta;

END IF;

DELETE FROM opere WHERE p\_opID=opID;

EXCEPTION

WHEN exceptii.opera\_inexistenta THEN

raise\_application\_error(exceptii.opera\_inexistenta\_nr,exceptii.opera\_inexistenta\_text);

WHEN OTHERS THEN

raise\_application\_error(-20014,'Eroare la ?tergere oper?! V? rug?m s? încerca?i din nou.');

END sterge\_opera;

PROCEDURE sterge\_subgen(

p\_subgen ierarhie.subgen%type,

p\_mesaj\_succes out varchar2)

IS

v\_count NUMBER;

v\_count\_carti number;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM ierarhie WHERE p\_subgen=subgen;

IF v\_count != 1 THEN

raise exceptii.subgen\_inexistent;

END IF;

select count(\*) into v\_count\_carti

from opere natural join abordari natural join ierarhie

where subgen=p\_subgen;

DELETE FROM ierarhie WHERE p\_subgen=subgen;

p\_mesaj\_succes := 'Au fost ?terse '||v\_count\_carti||' .';

EXCEPTION

WHEN exceptii.subgen\_inexistent THEN

raise\_application\_error(exceptii.subgen\_inexistent\_nr,exceptii.subgen\_inexistent\_text);

WHEN OTHERS THEN

raise\_application\_error(-20016,'Eroare la ?tergere oper?! V? rug?m s? încerca?i din nou.');

END sterge\_subgen;

END CRUD;

/

* Pachetul import\_export:

SET SERVEROUTPUT ON;

CREATE OR REPLACE PACKAGE import\_export

IS

PROCEDURE export\_database;

PROCEDURE import\_BD;

END import\_export;

/

CREATE OR REPLACE PACKAGE BODY import\_export

IS

g\_fisierID\_import utl\_file.file\_type;

g\_linie\_citita VARCHAR2(1000);

PROCEDURE export\_abordari

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'abordariExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'opID,subgen');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM abordari

)

LOOP

Utl\_File.Put\_Line(out\_file , '' || c\_linie.opID || ',' || c\_linie.subgen);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_abordari;

PROCEDURE export\_opere

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'opereExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'opID,titlu,volum');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM opere

)

LOOP

Utl\_File.Put\_Line(out\_file , '' || c\_linie.opID || ',' || c\_linie.titlu || ',' || c\_linie.volum);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_opere;

PROCEDURE export\_carti

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'cartiExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'ISBN,editura,calea,an\_aparitie,rating,opID');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM carti

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.ISBN || ',' || c\_linie.editura || ',' || c\_linie.calea || ',' || c\_linie.an\_aparitie || ',' || c\_linie.rating || ',' || c\_linie.opID);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_carti;

PROCEDURE export\_autori

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'autoriExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'autID,nume,prenume');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM autori

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.autID || ',' || c\_linie.nume || ',' || c\_linie.prenume);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_autori;

PROCEDURE export\_opere\_autori

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'opere\_autoriExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'opID,autID');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM opere\_autori

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.opID || ',' || c\_linie.autID);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_opere\_autori;

PROCEDURE export\_ierarhie

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'ierarhieExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'subgen,gen');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM ierarhie

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.subgen || ',' || c\_linie.gen);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_ierarhie;

PROCEDURE export\_utilizatori\_parola

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'utilizator\_parolaExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'utilizator,parola');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM utilizator\_parola

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.utilizator || ',' || c\_linie.parola);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_utilizatori\_parola;

PROCEDURE export\_citate

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'citateExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'id,text,autor');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM citate

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.id || ',' || c\_linie.text || ',' || c\_linie.autor);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_citate;

PROCEDURE export\_distante\_utilizatori

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'distante\_utilizatoriExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'u1,u2,distanta');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM distante\_utilizatori

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.u1 || ',' || c\_linie.u2 || ',' || c\_linie.distanta);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_distante\_utilizatori;

PROCEDURE export\_utilizator\_carte

IS

out\_file UTL\_FILE.FILE\_TYPE;

BEGIN

out\_file := Utl\_File.FOpen('EXPORT', 'utilizator\_carte\_ratingExport.csv' , 'W');

Utl\_File.Put\_Line(out\_file , 'utilizator,ISBN,rating');

UTL\_FILE.FFLUSH (out\_file);

FOR c\_linie IN

(SELECT \* FROM utilizator\_carte\_rating

)

LOOP

Utl\_File.Put\_Line(out\_file, '' || c\_linie.utilizator || ',' || c\_linie.ISBN || ',' || c\_linie.rating);

UTL\_FILE.FFLUSH (out\_file);

END LOOP;

UTL\_FILE.FCLOSE(out\_file);

END export\_utilizator\_carte;

PROCEDURE export\_database

IS

BEGIN

export\_abordari();

export\_opere();

export\_carti();

export\_autori();

export\_opere\_autori();

export\_ierarhie();

export\_utilizatori\_parola();

export\_citate();

export\_distante\_utilizatori();

export\_utilizator\_carte();

END export\_database;

PROCEDURE import\_BD\_tabel(

p\_fisier VARCHAR2,

p\_nume\_tabel VARCHAR2)

AS

v\_insert VARCHAR2(2500);

v\_antet VARCHAR2(1000);

BEGIN

g\_fisierID\_import := utl\_file.fopen ('EXPORT', p\_fisier, 'R');

utl\_file.get\_line(g\_fisierID\_import,v\_antet);

LOOP

utl\_file.get\_line(g\_fisierID\_import,g\_linie\_citita);

g\_linie\_citita := '''' || REPLACE(REPLACE(g\_linie\_citita,'''',''''''),',',''',''') || '''';

v\_insert := 'insert into ' || p\_nume\_tabel || '(' || v\_antet || ') values(' || g\_linie\_citita || ')';

-- dbms\_output.put\_line(v\_insert);

EXECUTE immediate v\_insert;

EXECUTE immediate 'commit';

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

utl\_file.fclose(g\_fisierID\_import);

WHEN OTHERS THEN

dbms\_output.put\_line(SQLERRM);

dbms\_output.put\_line(v\_insert);

utl\_file.fclose(g\_fisierID\_import);

END import\_BD\_tabel;

PROCEDURE import\_BD

AS

BEGIN

import\_BD\_tabel('opereExport.csv','opere');

import\_BD\_tabel('cartiExport.csv','carti');

import\_BD\_tabel('autoriExport.csv','autori');

import\_BD\_tabel('opere\_autoriExport.csv','opere\_autori');

import\_BD\_tabel('ierarhieExport.csv','ierarhie');

import\_BD\_tabel('abordariExport.csv','abordari');

import\_BD\_tabel('utilizator\_parolaExport.csv', 'utilizator\_parola');

import\_BD\_tabel('citateExport.csv','citate');

-- import\_BD\_tabel('distante\_utilizatoriExport.csv','distante\_utilizatori');

import\_BD\_tabel('utilizator\_carte\_ratingExport.csv','utilizator\_carte\_rating');

END import\_BD;

END import\_export;

/

BEGIN

import\_export.import\_BD();

END;

/

* Pachetul gestioneaza\_carte:

CREATE OR REPLACE PACKAGE gestioneaza\_carte

IS

FUNCTION carte\_random(

p\_HTML IN NUMBER)

RETURN VARCHAR2;

PROCEDURE rate(

p\_utilizator utilizator\_carte\_rating.utilizator%type,

p\_isbn utilizator\_carte\_rating.isbn%type,

p\_rating utilizator\_carte\_rating.rating%type,

p\_mesaj\_succes OUT VARCHAR2);

FUNCTION afiseaza\_carte(

p\_ISBN IN carti.ISBN%TYPE,

p\_HTML IN NUMBER)

RETURN VARCHAR2;

FUNCTION afiseaza\_opera(

p\_opID IN opere.opID%TYPE,

p\_HTML IN NUMBER)

RETURN VARCHAR2;

END gestioneaza\_carte;

/

CREATE OR REPLACE PACKAGE BODY gestioneaza\_carte IS

function carte\_random(p\_HTML IN NUMBER) return VARCHAR2

as

v\_max\_isbn number;

v\_isbn number;

v\_count number;

begin

select max(isbn) into v\_max\_isbn from carti;

loop

v\_isbn:=trunc(DBMS\_RANDOM.VALUE(1,v\_max\_isbn+1));

select count(isbn) into v\_count from carti where isbn=v\_isbn;

if v\_count = 1 then

return afiseaza\_carte(v\_isbn, p\_HTML);

end if;

end loop;

end carte\_random;

procedure rate (p\_utilizator utilizator\_carte\_rating.utilizator%type,

p\_isbn utilizator\_carte\_rating.isbn%type, p\_rating utilizator\_carte\_rating.rating%type,

p\_mesaj\_succes out varchar2)

as

v\_count number;

begin

select count(\*) into v\_count from utilizator\_carte\_rating

where p\_utilizator=utilizator and isbn=p\_isbn;

if v\_count = 0 then

insert into utilizator\_carte\_rating values(p\_utilizator,p\_isbn,p\_rating);

p\_mesaj\_succes := 'Felicit?ri! Primul dvs rating la aceast? carte a fost înregistrat!';

elsif v\_count = 1 then

update utilizator\_carte\_rating set rating=p\_rating

where utilizator=p\_utilizator and isbn=p\_isbn;

p\_mesaj\_succes := 'Felicit?ri! Ratingul dvs pentru aceast? carte a fost actualizat!';

end if;

exception

when exceptii.cheie\_straina\_gresita then

raise\_application\_error(exceptii.cheie\_straina\_gresita\_nr,'Eroare la introducere rating! V? rug?m s? verifica?i utilizatorul ?i isbn-ul s? fie valide.');

when exceptii.constrangere then

raise\_application\_error(exceptii.constrangere\_nr,exceptii.constrangere\_text || '''rating''.');

when others then

raise\_application\_error(-20017,'Eroare la introducere rating! V? rug?m încerca?i din nou!');

end rate;

FUNCTION marcheaza\_continut(

p\_HTML IN NUMBER,

p\_tag\_inceput IN VARCHAR2,

p\_tag\_final IN VARCHAR2,

p\_continut IN VARCHAR2)

RETURN VARCHAR2

AS

BEGIN

IF (p\_HTML > 0) THEN

RETURN p\_tag\_inceput || p\_continut || p\_tag\_final;

ELSE

RETURN p\_continut || ', ';

END IF;

END marcheaza\_continut;

FUNCTION afiseaza\_carte(

p\_ISBN IN carti.ISBN%TYPE,

p\_HTML IN NUMBER)

RETURN VARCHAR2

AS

v\_rezultat VARCHAR2(32767);

v\_titlu opere.titlu%TYPE;

v\_volum opere.volum%TYPE;

v\_editura carti.editura%TYPE;

v\_an\_aparitie carti.an\_aparitie%TYPE;

v\_calea carti.calea%TYPE;

v\_rating carti.rating%TYPE;

v\_opID carti.opID%TYPE;

v\_autori VARCHAR2(32767);

v\_subgenuri\_abordate VARCHAR2(32767);

BEGIN

SELECT editura, an\_aparitie, calea, rating, opID

INTO v\_editura, v\_an\_aparitie, v\_calea, v\_rating, v\_opID

FROM carti

WHERE ISBN = p\_ISBN;

SELECT titlu, volum

INTO v\_titlu, v\_volum

FROM opere

WHERE opID = v\_opID;

v\_rezultat := marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Opera: ' || v\_titlu);

v\_rezultat := v\_rezultat || marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Volum: ' || v\_volum);

v\_rezultat := v\_rezultat || marcheaza\_continut(p\_HTML, '<p>', '</p>', 'ISBN: ' || p\_ISBN);

v\_rezultat := v\_rezultat || marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Editura: ' || v\_editura);

v\_rezultat := v\_rezultat || marcheaza\_continut(p\_HTML, '<p>', '</p>', 'An aparitie: ' || v\_an\_aparitie);

FOR c\_linie IN (SELECT nume, prenume FROM autori NATURAL JOIN opere\_autori WHERE opID = v\_opID) LOOP

v\_autori := v\_autori || marcheaza\_continut(p\_HTML, '<li>', '</li>', c\_linie.nume || ' ' || c\_linie.prenume);

END LOOP;

v\_autori := marcheaza\_continut(p\_HTML, '<ol>', '</ol>', v\_autori);

v\_autori := marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Autori: ' || v\_autori);

v\_rezultat := v\_rezultat || v\_autori;

FOR c\_linie IN (SELECT subgen, gen FROM abordari NATURAL JOIN ierarhie WHERE opID = v\_opID) LOOP

v\_subgenuri\_abordate := v\_subgenuri\_abordate || marcheaza\_continut(p\_HTML, '<li>', '</li>', c\_linie.subgen || ' (' || c\_linie.gen || ')');

END LOOP;

v\_subgenuri\_abordate := marcheaza\_continut(p\_HTML, '<ol>', '</ol>', v\_subgenuri\_abordate);

v\_subgenuri\_abordate := marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Subgenuri abordate: ' || v\_subgenuri\_abordate);

v\_rezultat := v\_rezultat || v\_subgenuri\_abordate;

v\_rezultat := v\_rezultat || marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Rating: ' || v\_rating);

v\_rezultat := v\_rezultat || marcheaza\_continut(p\_HTML, '<a href="', '"> Calea </a>', v\_calea);

v\_rezultat := marcheaza\_continut(p\_HTML, '<div style="border:1px solid black; margin: 10px;padding:10px">', '</div>', v\_rezultat);

RETURN v\_rezultat;

END afiseaza\_carte;

FUNCTION afiseaza\_opera(p\_opID IN opere.opID%TYPE, p\_HTML IN NUMBER) RETURN VARCHAR2

AS

v\_rezultat VARCHAR2(32767);

v\_titlu opere.titlu%TYPE;

v\_volum opere.volum%TYPE;

v\_autori VARCHAR2(32767);

BEGIN

SELECT titlu, volum INTO v\_titlu, v\_volum FROM opere WHERE opID = p\_opID;

v\_rezultat := marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Opera: ' || v\_titlu);

v\_rezultat := v\_rezultat || marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Volum: ' || v\_volum);

FOR c\_linie IN (SELECT nume, prenume FROM autori NATURAL JOIN opere\_autori WHERE opID = p\_opID) LOOP

v\_autori := v\_autori || marcheaza\_continut(p\_HTML, '<li>', '</li>', c\_linie.nume || ' ' || c\_linie.prenume);

END LOOP;

v\_autori := marcheaza\_continut(p\_HTML, '<ol>', '</ol>', v\_autori);

v\_autori := marcheaza\_continut(p\_HTML, '<p>', '</p>', 'Autori: ' || v\_autori);

v\_rezultat := v\_rezultat || v\_autori;

v\_rezultat := marcheaza\_continut(p\_HTML, '<div style="border:1px solid black; margin: 10px; padding: 10px">', '</div>', v\_rezultat);

RETURN v\_rezultat;

END afiseaza\_opera;

END gestioneaza\_carte;

/

* Pachetul logica\_aplicatiei:

CREATE OR REPLACE PACKAGE logica\_aplicatiei AS

FUNCTION login(

p\_utilizator IN utilizator\_parola.utilizator%TYPE,

p\_parola IN utilizator\_parola.parola%TYPE)

RETURN VARCHAR2;

procedure citat\_random(v\_text out varchar2, v\_autor out varchar2);

END logica\_aplicatiei;

/

CREATE OR REPLACE PACKAGE BODY logica\_aplicatiei AS

FUNCTION login(

p\_utilizator IN utilizator\_parola.utilizator%TYPE,

p\_parola IN utilizator\_parola.parola%TYPE)

RETURN VARCHAR2

IS

v\_corect NUMBER(1);

BEGIN

SELECT COUNT(\*) INTO v\_corect FROM utilizator\_parola WHERE utilizator = p\_utilizator AND parola = p\_parola;

IF (v\_corect = 0) THEN

RETURN 'F';

ELSE

RETURN 'T';

END IF;

END login;

procedure citat\_random(v\_text out varchar2, v\_autor out varchar2)

as

v\_max\_id number;

v\_id number;

v\_count number;

begin

select max(id) into v\_max\_id from citate;

loop

v\_id:=trunc(DBMS\_RANDOM.VALUE(1,v\_max\_id+1));

select count(id) into v\_count from citate where id=v\_id;

if v\_count = 1 then

select text,autor into v\_text,v\_autor from citate where id=v\_id;

return;

end if;

end loop;

end citat\_random;

END logica\_aplicatiei;

/

* Pachetul recomandare:

create or replace type vector\_number is table of number;

/

create or replace type vector\_varchar2 is table of varchar2(50);

/

CREATE OR REPLACE PACKAGE recomandare

IS

FUNCTION recomanda\_carti(

p\_utilizator IN utilizator\_parola.utilizator%TYPE,

p\_HTML IN NUMBER)

RETURN VARCHAR2;

END recomandare;

/

CREATE OR REPLACE PACKAGE BODY recomandare

IS

FUNCTION recomanda\_bazat\_pe\_utilizatori(

p\_utilizator IN utilizator\_parola.utilizator%TYPE)

RETURN NUMBER

IS

v\_rezultat VARCHAR2(32767);

v\_similaritate\_maxima NUMBER;

v\_numar\_ut\_optimi NUMBER;

v\_utilizator\_optim utilizator\_parola.utilizator%TYPE;

v\_ISBN carti.ISBN%TYPE;

BEGIN

SELECT MAX(distanta)

INTO v\_similaritate\_maxima

FROM distante\_utilizatori

WHERE u1 = p\_utilizator

OR u2 = p\_utilizator;

SELECT COUNT('1')

INTO v\_numar\_ut\_optimi

FROM distante\_utilizatori

WHERE u1 = p\_utilizator

AND distanta = v\_similaritate\_maxima;

IF (v\_numar\_ut\_optimi > 0) THEN

SELECT u2

INTO v\_utilizator\_optim

FROM distante\_utilizatori

WHERE u1 = p\_utilizator

AND distanta = v\_similaritate\_maxima;

ELSE

SELECT u1

INTO v\_utilizator\_optim

FROM distante\_utilizatori

WHERE u2 = p\_utilizator

AND distanta = v\_similaritate\_maxima;

END IF;

-- Avem userul cu cea mai mare similaritate. Urmeaza sa luam cartea la care a dat cel mai mare rating, dar userul initial nu a dat (adica nu a citit-o).

SELECT ISBN

INTO v\_ISBN

FROM

(SELECT ISBN,

rating

FROM utilizator\_carte\_rating

WHERE utilizator = v\_utilizator\_optim

MINUS

SELECT ISBN,

rating

FROM utilizator\_Carte\_rating

WHERE utilizator = p\_utilizator

)

WHERE ROWNUM <= 1

ORDER BY rating DESC;

RETURN v\_isbn;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN '';

WHEN OTHERS THEN

RETURN NULL;

END recomanda\_bazat\_pe\_utilizatori;

FUNCTION recomandare\_bazata\_pe\_continut(

p\_utilizator utilizator\_carte\_rating.utilizator%type)

RETURN NUMBER

AS

v\_autori vector\_number;

v\_subgenuri vector\_varchar2;

v\_recom carti.isbn%type;

BEGIN

SELECT DISTINCT autID bulk collect

INTO v\_autori

FROM

(SELECT isbn

FROM utilizator\_carte\_rating

WHERE utilizator=p\_utilizator

AND rating >= 6

) NATURAL

JOIN carti NATURAL

JOIN opere NATURAL

JOIN opere\_autori NATURAL

JOIN autori;

SELECT DISTINCT(subgen) bulk collect

INTO v\_subgenuri

FROM opere\_autori NATURAL

JOIN opere NATURAL

JOIN abordari

WHERE autId IN

(SELECT \* FROM TABLE(v\_autori)

);

SELECT isbn

INTO v\_recom

FROM

(SELECT isbn

FROM abordari NATURAL

JOIN opere NATURAL

JOIN carti NATURAL

JOIN

(SELECT isbn, utilizator, rating rating\_personal FROM utilizator\_carte\_rating

)

WHERE subgen IN

(SELECT \* FROM TABLE(v\_subgenuri)

)

AND utilizator!=p\_utilizator

ORDER BY rating\_personal DESC

)

WHERE rownum = 1;

RETURN v\_recom;

EXCEPTION

WHEN no\_data\_found THEN

RETURN NULL;

WHEN OTHERS THEN

RETURN NULL;

END recomandare\_bazata\_pe\_continut;

FUNCTION recomanda\_carti(

p\_utilizator IN utilizator\_parola.utilizator%TYPE,

p\_HTML IN NUMBER)

RETURN VARCHAR2

IS

v\_isbn1 carti.isbn%type;

v\_isbn2 carti.isbn%type;

v\_rezultat VARCHAR2(30000);

BEGIN

v\_isbn1 := recomanda\_bazat\_pe\_utilizatori(p\_utilizator);

v\_isbn2 := recomandare\_bazata\_pe\_continut(p\_utilizator);

IF v\_isbn1 IS NOT NULL THEN

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(v\_isbn1,p\_HTML);

END IF;

IF v\_isbn2 IS NOT NULL THEN

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(v\_isbn2,p\_HTML);

END IF;

IF v\_rezultat IS NULL THEN

v\_rezultat := 'Nu avem c?r?i de recomandat pentru dvs. V? rug?m da?i-v? cu p?rerea despre anumite c?r?i ?i reveni?i pe aceast? pagin?.';

END IF;

RETURN v\_rezultat;

END recomanda\_carti;

END recomandare;

/

* Pachetul paginare:

SELECT \*

FROM

(SELECT \*

FROM carti

WHERE rating <= :1

AND NOT (rating = :1

AND isbn >= :2)

ORDER BY rating DESC,

isbn DESC

)

WHERE rownum <=5;

SELECT \*

FROM

(SELECT isbn,

rating

FROM carti

WHERE rating >= :1

AND NOT (rating = :1

AND isbn <= :2)

ORDER BY rating asc,

isbn asc

)

WHERE rownum <=5 order by rating desc,isbn desc;

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere

WHERE upper(titlu) LIKE :1

AND isbn >=:2

ORDER BY isbn

)

WHERE rownum <=5;

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere

WHERE upper(titlu) LIKE :1

AND isbn <:2

ORDER BY isbn desc

)

WHERE rownum <=5

order by isbn asc;

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere NATURAL

JOIN opere\_autori NATURAL

JOIN autori

WHERE upper(nume) LIKE :1

AND upper(prenume) LIKE :2

AND isbn >=:3

ORDER BY isbn

)

WHERE rownum <=5;

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere NATURAL

JOIN opere\_autori NATURAL

JOIN autori

WHERE upper(nume) LIKE :1

AND upper(prenume) LIKE :2

AND isbn <:3

ORDER BY isbn desc

)

WHERE rownum <=5 order by isbn asc;

set serveroutput on;

declare

p\_rating\_out carti.rating%type;

p\_isbn\_out carti.isbn%type;

begin

dbms\_output.put\_line(paginare.paginare\_carti\_back(10, 51994, 1, p\_rating\_out, p\_isbn\_out));

end;

/

CREATE OR REPLACE PACKAGE paginare

AS

FUNCTION paginare\_carti\_next(

p\_rating\_in carti.rating%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_rating\_out OUT carti.rating%type,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2;

FUNCTION paginare\_carti\_back(

p\_rating\_in carti.rating%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_rating\_out OUT carti.rating%type,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2;

FUNCTION paginare\_opere\_next(

p\_format VARCHAR2,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2;

FUNCTION paginare\_opere\_back(

p\_format VARCHAR2,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2;

FUNCTION paginare\_autori\_next(

p\_nume\_in autori.nume%type,

p\_prenume\_in autori.prenume%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2;

FUNCTION paginare\_autori\_back(

p\_nume\_in autori.nume%type,

p\_prenume\_in autori.prenume%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2;

END paginare;

/

CREATE OR REPLACE PACKAGE body paginare

IS

FUNCTION paginare\_carti\_next(

p\_rating\_in carti.rating%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_rating\_out OUT carti.rating%type,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2

AS

CURSOR pagina (p\_rating\_in carti.rating%type, p\_isbn\_in carti.isbn%type)

IS

SELECT \*

FROM

(SELECT isbn,

rating

FROM carti

WHERE rating <= p\_rating\_in

AND NOT (rating = p\_rating\_in

AND isbn >= p\_isbn\_in)

ORDER BY rating DESC,

isbn DESC

)

WHERE rownum <=5;

v\_rezultat VARCHAR2(32000);

v\_rating\_limita NUMBER := p\_rating\_in;

v\_isbn\_limita NUMBER := p\_isbn\_in;

BEGIN

IF v\_rating\_limita IS NULL THEN

SELECT MAX(rating) INTO v\_rating\_limita FROM carti;

END IF;

IF v\_isbn\_limita IS NULL THEN

SELECT MAX(isbn)+1 INTO v\_isbn\_limita FROM carti;

END IF;

FOR carte IN pagina(v\_rating\_limita,v\_isbn\_limita)

LOOP

p\_rating\_out := carte.rating;

p\_isbn\_out := carte.isbn;

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(carte.isbn,p\_HTML);

END LOOP;

RETURN v\_rezultat;

END paginare\_carti\_next;

FUNCTION paginare\_carti\_back(

p\_rating\_in carti.rating%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_rating\_out OUT carti.rating%type,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2

AS

CURSOR pagina (p\_rating\_in carti.rating%type, p\_isbn\_in carti.isbn%type)

IS

SELECT \*

FROM

(SELECT isbn,

rating

FROM carti

WHERE rating >= p\_rating\_in

AND NOT (rating = p\_rating\_in

AND isbn < p\_isbn\_in)

ORDER BY rating ASC,

isbn ASC

)

WHERE rownum <=5 ORDER BY rating DESC, ISBN DESC;

v\_rezultat VARCHAR2(32000);

v\_rating\_limita NUMBER := p\_rating\_in;

v\_isbn\_limita NUMBER := p\_isbn\_in;

v\_count number :=0;

BEGIN

FOR carte IN pagina(v\_rating\_limita,v\_isbn\_limita)

LOOP

v\_count := v\_count+1;

if v\_count = 1 then

p\_rating\_out := carte.rating;

p\_isbn\_out := carte.isbn+1;

end if;

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(carte.isbn,p\_HTML);

END LOOP;

RETURN v\_rezultat;

END paginare\_carti\_back;

FUNCTION paginare\_opere\_next(

p\_format VARCHAR2,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2

AS

CURSOR pagina (p\_isbn\_in carti.isbn%type)

IS

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere

WHERE upper(titlu) LIKE upper(p\_format)

AND isbn >=p\_isbn\_in

ORDER BY isbn

)

WHERE rownum <=5;

v\_rezultat VARCHAR2(32000);

v\_isbn\_limita NUMBER := p\_isbn\_in;

BEGIN

IF v\_isbn\_limita IS NULL THEN

SELECT MIN(isbn) INTO v\_isbn\_limita FROM carti;

END IF;

FOR carte IN pagina(v\_isbn\_limita)

LOOP

p\_isbn\_out := carte.isbn;

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(carte.isbn,p\_HTML);

END LOOP;

p\_isbn\_out := p\_isbn\_out + 1;

RETURN v\_rezultat;

END paginare\_opere\_next;

FUNCTION paginare\_opere\_back(

p\_format VARCHAR2,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2

AS

CURSOR pagina (p\_isbn\_in carti.isbn%type)

IS

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere

WHERE upper(titlu) LIKE upper(p\_format)

AND isbn < p\_isbn\_in

ORDER BY isbn desc

)

WHERE rownum <=5 order by isbn asc;

v\_rezultat VARCHAR2(32000);

v\_isbn\_limita NUMBER := p\_isbn\_in;

v\_count NUMBER :=0;

BEGIN

FOR carte IN pagina(v\_isbn\_limita)

LOOP

v\_count :=v\_count+1;

IF v\_count = 1 THEN

p\_isbn\_out := carte.isbn;

END IF;

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(carte.isbn,p\_HTML);

END LOOP;

p\_isbn\_out := p\_isbn\_out - 1;

RETURN v\_rezultat;

END paginare\_opere\_back;

FUNCTION paginare\_autori\_next(

p\_nume\_in autori.nume%type,

p\_prenume\_in autori.prenume%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2

AS

CURSOR pagina (p\_isbn\_in carti.isbn%type)

IS

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere NATURAL

JOIN opere\_autori NATURAL

JOIN autori

WHERE upper(nume) LIKE upper(p\_nume\_in)

AND upper(prenume) LIKE upper(p\_prenume\_in)

AND isbn >=p\_isbn\_in

ORDER BY isbn

)

WHERE rownum <=5;

v\_rezultat VARCHAR2(32000);

v\_isbn\_limita NUMBER := p\_isbn\_in;

BEGIN

IF v\_isbn\_limita IS NULL THEN

SELECT MIN(isbn) INTO v\_isbn\_limita FROM carti;

END IF;

FOR carte IN pagina(v\_isbn\_limita)

LOOP

p\_isbn\_out := carte.isbn;

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(carte.isbn,p\_HTML);

END LOOP;

p\_isbn\_out := p\_isbn\_out + 1;

RETURN v\_rezultat;

END paginare\_autori\_next;

FUNCTION paginare\_autori\_back(

p\_nume\_in autori.nume%type,

p\_prenume\_in autori.prenume%type,

p\_isbn\_in carti.isbn%type,

p\_HTML NUMBER,

p\_isbn\_out OUT carti.isbn%type)

RETURN VARCHAR2

AS

CURSOR pagina (p\_isbn\_in carti.isbn%type)

IS

SELECT \*

FROM

(SELECT isbn

FROM carti NATURAL

JOIN opere NATURAL

JOIN opere\_autori NATURAL

JOIN autori

WHERE upper(nume) LIKE upper(p\_nume\_in)

AND upper(prenume) LIKE upper(p\_prenume\_in)

AND isbn <p\_isbn\_in

ORDER BY isbn desc

)

WHERE rownum <=5 order by isbn asc;

v\_rezultat VARCHAR2(32000);

v\_isbn\_limita NUMBER := p\_isbn\_in;

v\_count NUMBER :=0;

BEGIN

FOR carte IN pagina(v\_isbn\_limita)

LOOP

v\_count :=v\_count+1;

IF v\_count = 1 THEN

p\_isbn\_out := carte.isbn;

END IF;

v\_rezultat := v\_rezultat || gestioneaza\_carte.afiseaza\_carte(carte.isbn,p\_HTML);

END LOOP;

RETURN v\_rezultat;

END paginare\_autori\_back;

END paginare;

/

* Triggerele utilizate:

-- Trigger care recalculeaza distantele intre utilizatori in momentul in care un anumit utilizator isi modifica ratingul acordat unei carti sau

-- da un rating nou

ALTER SESSION SET PLSCOPE\_SETTINGS = 'IDENTIFIERS:NONE';

CREATE OR REPLACE TYPE carte IS OBJECT(

ISBN NUMBER

);

/

CREATE OR REPLACE TYPE carti\_cotate IS TABLE OF carte;

/

CREATE OR REPLACE TYPE nume\_utilizator IS OBJECT (

utilizator VARCHAR2(100)

);

/

CREATE OR REPLACE TYPE utilizatori IS TABLE OF nume\_utilizator;

/

CREATE OR REPLACE TRIGGER actualizeaza\_distante

FOR INSERT OR UPDATE OF rating ON utilizator\_carte\_rating

COMPOUND TRIGGER

v\_count\_global NUMBER := 1;

utilizatori\_de\_modificat utilizatori := utilizatori();

FUNCTION calculeaza\_similaritate(

p\_u1 IN utilizator\_parola.utilizator%TYPE,

p\_u2 IN utilizator\_parola.utilizator%TYPE)

RETURN NUMBER

IS

carti\_u1 carti\_cotate := carti\_cotate(); -- cartile care au primit rating de la utilizatorul 1

carti\_u2 carti\_cotate := carti\_cotate(); -- cartile care au primit rating de la utilizatorul 2

v\_numarator NUMBER := 0;

v\_numitor1 NUMBER := 0;

v\_numitor2 NUMBER := 0;

v\_rating\_u1 NUMBER;

v\_rating\_u2 NUMBER;

v\_count NUMBER;

BEGIN

SELECT carte(ISBN) BULK COLLECT INTO carti\_u1 FROM utilizator\_carte\_rating

WHERE utilizator = p\_u1;

SELECT carte(ISBN) BULK COLLECT INTO carti\_u2 FROM utilizator\_carte\_rating

WHERE utilizator = p\_u2;

-- Calculam numaratorul din formula similaritatii

FOR c\_line IN (

SELECT t1.ISBN AS ISBN FROM TABLE(carti\_u1) t1

INTERSECT

SELECT t2.ISBN AS ISBN FROM TABLE(carti\_u2) t2)

LOOP

SELECT rating INTO v\_rating\_u1 FROM utilizator\_carte\_rating WHERE utilizator = p\_u1 AND ISBN = c\_line.ISBN;

SELECT rating INTO v\_rating\_u2 FROM utilizator\_carte\_rating WHERE utilizator = p\_u2 AND ISBN = c\_line.ISBN;

v\_numarator := v\_numarator + v\_rating\_u1 \* v\_rating\_u2;

END LOOP;

-- Calculam numitorul din formula similaritatii

FOR c\_line IN (

SELECT t1.ISBN AS ISBN FROM TABLE(carti\_u1) t1

UNION

SELECT t2.ISBN AS ISBN FROM TABLE(carti\_u2) t2)

LOOP

SELECT COUNT('1') INTO v\_count FROM utilizator\_carte\_rating WHERE utilizator = p\_u1 AND ISBN = c\_line.ISBN;

IF (v\_count > 0) THEN

SELECT rating INTO v\_rating\_u1 FROM utilizator\_carte\_rating WHERE utilizator = p\_u1 AND ISBN = c\_line.ISBN;

ELSE

v\_rating\_u1 := 0;

END IF;

SELECT count('1') INTO v\_count FROM utilizator\_carte\_rating WHERE utilizator = p\_u2 AND ISBN = c\_line.ISBN;

IF (v\_count > 0) THEN

SELECT rating INTO v\_rating\_u2 FROM utilizator\_carte\_rating WHERE utilizator = p\_u2 AND ISBN = c\_line.ISBN;

ELSE

v\_rating\_u2 := 0;

END IF;

v\_numitor1 := v\_numitor1 + (v\_rating\_u1 \* v\_rating\_u1);

v\_numitor2 := v\_numitor2 + (v\_rating\_u2 \* v\_rating\_u2);

END LOOP;

--Daca niciunul din cei 2 nu a dat rate la vreo carte => se returneaza 0 similaritatea

IF(v\_numitor1 + v\_numitor2 = 0) THEN

RETURN 0;

END IF;

RETURN (v\_numarator / (SQRT(v\_numitor1) \* SQRT(v\_numitor2)));

END calculeaza\_similaritate;

-- Procedura care pentru un anumit utilizator, actualizeaza distanta cu toti ceilalti utilizatori

PROCEDURE actualizeaza\_distanta(p\_utilizator IN utilizator\_parola.utilizator%TYPE) IS

v\_similaritate NUMBER;

BEGIN

FOR c\_line IN (SELECT DISTINCT(utilizator) AS utilizator FROM utilizator\_carte\_rating WHERE utilizator != p\_utilizator) LOOP

-- Pentru toti utilizatorii diferiti de cel curent, aflam similaritatea cu utilizatorul curent si facem update in tabela distante

v\_similaritate := calculeaza\_similaritate(p\_utilizator, c\_line.utilizator);

IF (p\_utilizator < c\_line.utilizator) THEN

UPDATE distante\_utilizatori SET distanta = v\_similaritate WHERE u1 = p\_utilizator AND u2 = c\_line.utilizator;

ELSE

UPDATE distante\_utilizatori SET distanta = v\_similaritate WHERE u1 = c\_line.utilizator AND u2 = p\_utilizator;

END IF;

END LOOP;

END actualizeaza\_distanta;

AFTER EACH ROW IS

BEGIN

utilizatori\_de\_modificat.extend();

utilizatori\_de\_modificat(v\_count\_global) := nume\_utilizator(:NEW.utilizator);

v\_count\_global := v\_count\_global + 1;

END AFTER EACH ROW;

AFTER STATEMENT IS

BEGIN

FOR c\_linie IN (SELECT \* FROM table(utilizatori\_de\_modificat)) LOOP

actualizeaza\_distanta(c\_linie.utilizator);

END LOOP;

END AFTER STATEMENT;

END actualizeaza\_distante;

/

-- Table care introduce un utilizator nou introdus si in tabela de distante, a.i. pt. orice pereche (u,v) din tabela distante,

-- avem: u < v.

CREATE OR REPLACE TRIGGER insereaza\_utilizator\_distante

FOR INSERT ON utilizator\_parola

COMPOUND TRIGGER

utilizatori\_vechi utilizatori;

BEFORE STATEMENT IS

BEGIN

SELECT nume\_utilizator(utilizator) BULK COLLECT INTO utilizatori\_vechi FROM utilizator\_parola;

END BEFORE STATEMENT;

AFTER EACH ROW IS

BEGIN

FOR c\_linie IN (SELECT tab.utilizator AS utilizator FROM TABLE(utilizatori\_vechi) tab) LOOP

IF (:NEW.utilizator < c\_linie.utilizator) THEN

INSERT INTO distante\_utilizatori(u1, u2, distanta)

VALUES (:NEW.utilizator, c\_linie.utilizator, null);

ELSE

INSERT INTO distante\_utilizatori(u1, u2, distanta)

VALUES (c\_linie.utilizator,:NEW.utilizator, null);

END IF;

END LOOP;

END AFTER EACH ROW;

END insereaza\_utilizator\_distante;

/

-- Trigger care actualizeaza media ratingurilor unei carti

create or replace trigger medie\_rating

after update of rating or insert on utilizator\_carte\_rating

for each row

declare

v\_isbn carti.isbn%type;

v\_nr\_rating number;

v\_rating number;

v\_medie number;

v\_suma number;

begin

v\_isbn := :new.isbn;

select nr\_rating,rating into v\_nr\_rating,v\_rating

from carti

where isbn = v\_isbn;

v\_suma := v\_rating \* v\_nr\_rating;

if inserting then

v\_suma := v\_suma + :new.rating;

v\_nr\_rating := v\_nr\_rating + 1;

v\_medie := v\_suma / v\_nr\_rating;

update carti set nr\_rating = v\_nr\_rating, rating = v\_medie where isbn=v\_isbn;

else

v\_suma := v\_suma - :old.rating + :new.rating;

v\_medie := v\_suma / v\_nr\_rating;

update carti set rating = v\_medie where isbn=v\_isbn;

end if;

end medie\_rating;

/

ALTER SESSION SET PLSCOPE\_SETTINGS = 'IDENTIFIERS:ALL';

* Indecsii utilizati:

DROP INDEX nume\_prenume\_autor;

/

DROP INDEX titlu\_volum;

/

DROP INDEX rating\_ISBN;

/

drop index isbn\_opID;

/

CREATE INDEX nume\_prenume\_autor ON autori

(UPPER(nume), UPPER(prenume)

);

/

CREATE INDEX titlu\_volum ON opere

(UPPER(titlu), volum

);

/

CREATE INDEX rating\_ISBN ON carti

(rating, ISBN

);

/

CREATE INDEX isbn\_opID ON carti

(isbn,opID

);

/

BEGIN

DBMS\_STATS.GATHER\_TABLE\_STATS(NULL, 'AUTORI', METHOD\_OPT=>'for all indexed columns', CASCADE => true);

DBMS\_STATS.GATHER\_TABLE\_STATS(NULL, 'OPERE', METHOD\_OPT=>'for all indexed columns', CASCADE => true);

DBMS\_STATS.GATHER\_TABLE\_STATS(NULL, 'CARTI', METHOD\_OPT=>'for all indexed columns', CASCADE => true);

DBMS\_STATS.GATHER\_TABLE\_STATS(NULL, 'OPERE\_AUTORI', METHOD\_OPT=>'for all indexed columns', CASCADE => true);

DBMS\_STATS.GATHER\_TABLE\_STATS(NULL, 'IERARHIE', METHOD\_OPT=>'for all indexed columns', CASCADE => true);

DBMS\_STATS.GATHER\_TABLE\_STATS(NULL, 'ABORDARI', METHOD\_OPT=>'for all indexed columns', CASCADE => true);

END;