

# **Gestionarea librărilor**

**Master An I, Semestrul I, 2023-2024**

**Denisa Predescu,**

**BDTS Grupa 405**

## Cuprins

Prezentarea concisă a bazei de date .....	3
Diagrama entitate-relație (ERD) .....	4
Diagramă conceptuală .....	6
Design logic .....	8
Design fizic .....	9
Sistemul este în FN3 .....	10
Exemplu de atribut repetitiv care face ca un tabel relațional să nu fie în FN1 .....	10
Exemplu de tabel relațional din diagramă care este în FN1, dar nu în FN2 .....	12
Exemplu de tabel relațional din diagramă care este în FN2, dar nu în FN3 .....	13
Implementarea tabelelor și adăugarea de informații .....	16
Implementarea tabelelor .....	16
Adăugare de înregistrări în fiecare tabel creat .....	20
Interogări .....	40
Creare tabel pentru mesaje .....	59
Probleme în PL/SQL .....	60
Înregistrarea excepțiilor apărute în tabelul MESAJE .....	102

## Prezentarea concisă a bazei de date

Baza de date gestionează datele unor librării fizice, reținând informații ce ajută la buna funcționare a unor librării în viața reală. Așadar se depozitează date referitoare la cărți (acestea urmează să fie cumpărate), categoriile, scriitorii și limbile în care se găsesc cărțile, seriile din care sunt compuse cărțile, angajați care lucrează în librării, bonurile emise în urma achiziționării, librării și locația acestora, precum și stocul de cărți care reține numărul de la fiecare librărie în parte.

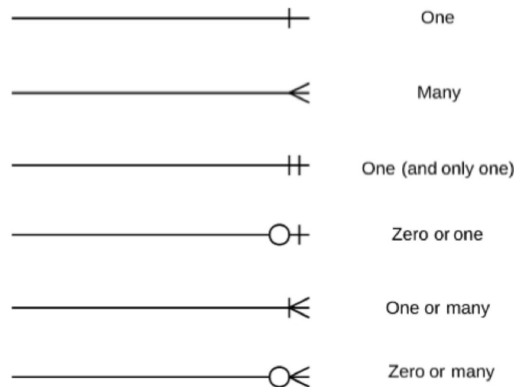
Cărțile sunt încadrate în diverse categorii și pot fi scrise de unul sau mai multi scriitori, precizându-se ca și aceștia din urma pot avea mai multe cărți scrise și acestea să fie incluse în diverse categorii. O carte poate fi inclusă în cel mult o serie, dar nu este obligatoriu să facă parte din una. În continuare, o carte poate să se găsească în una sau mai multe limbi în librării, dar trebuie să fie în minim una dacă cartea există pe stoc.

Angajații sunt de două tipuri: vânzatori și îngrijitori. Vânzatorii sunt cei care vând cărțile, iar bonurile emise după cumpărare sunt și ele reținute, făcând legătura dintre vânzător și cartea cumpărată. Se precizează că un angajat poate să vândă de mai multe ori aceeași carte, în aceeași zi, aceasta cât timp cartea respective se găsește la librăria de la care se dorește cumpărarea (dacă se găsește pe stoc) aceasta deoarece vinde exemplare diferite ale aceleiași cărți. Stocul trebuie actualizat după fiecare vânzare.

Un angajat trebuie să lucreze la o librărie pentru a fi considerat angajat, iar librăriile se găsesc la o locație stabilită dintr-un oraș.

## Diagrama entitate-relație (ERD)

Diagrama entitate-relație (ERD) prezintă entitățile necesare în baza de date, detalii informative ce se găsesc în fiecare entitate (atribute) și relațiile dintre acestea. Există 3 tipuri de relații (one-to-one, one-to-many, many-to-many), aratând prin conexiuni cardinalitatea minimă și maximă a relațiilor dintre entități.



### *Prezentarea cardinalității minime și maxime posibile dintre entități*

Printre atributele specifice unei entități se numără și cheia primară care este evidențiată în diagrama prin folosirea structurii *atribut (PK)*.

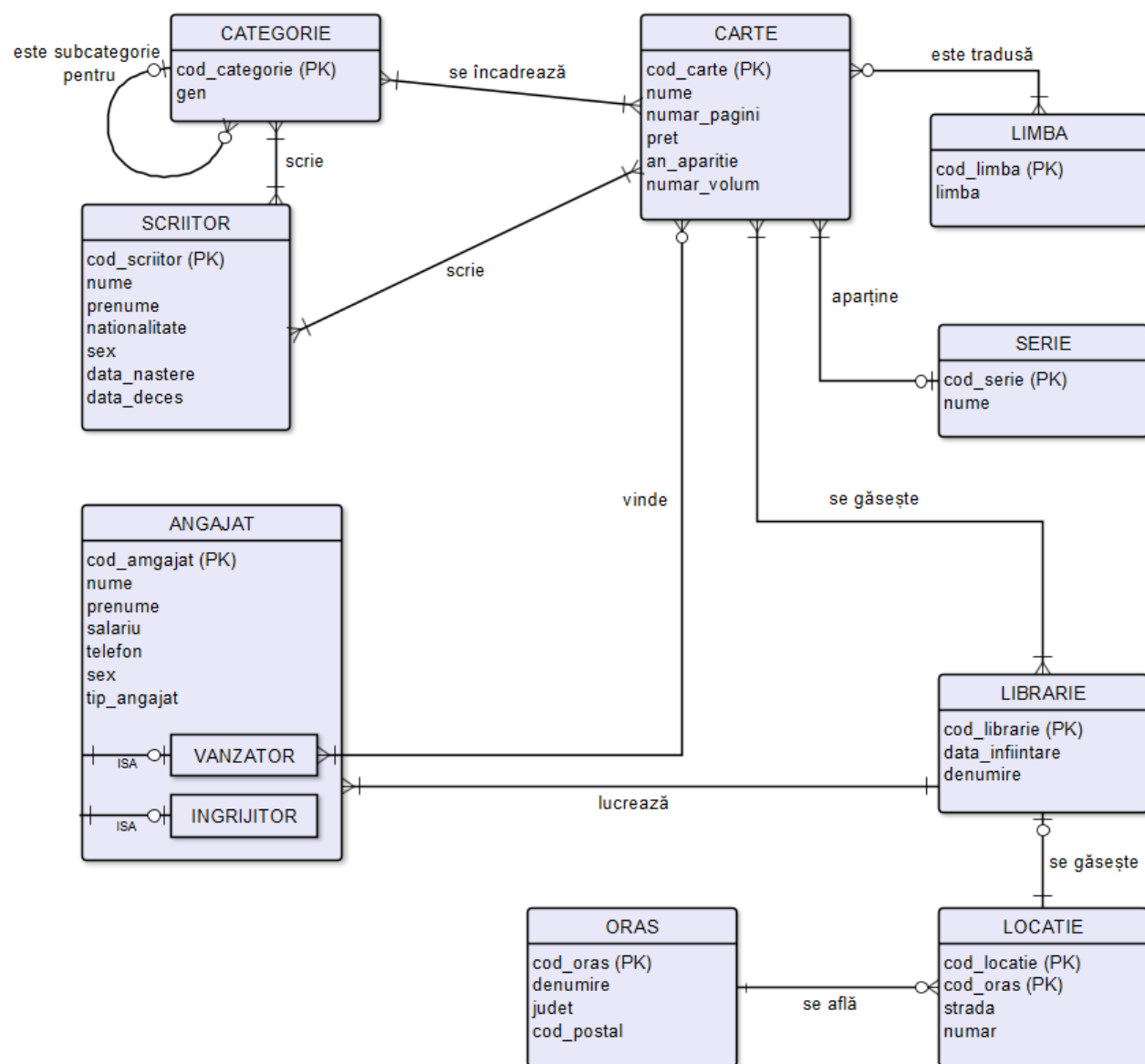


Diagrama ERD

## Diagramă conceptuală

Comparativ cu diagrama ERD care oferă detalii cu privire la entități, atribute și relații, diagrama conceptuală a bazei de date arată o prezentare mai detaliată a bazei de date, oferindu-se mai multe detalii despre cum este realizată aceasta.

Relațiile dintre entități devin chei externe ce sunt specificate prin prezența structurii *atribut (FK)*. În cazul unei relații 1:1, cheia externă este poziționată în tabelul cu mai puține linii, adică depinde de cardinalitatea minimă a numei relații. Un exemplu sugestiv este relația dintre *LIBRARIE* și *LOCAȚIE* care are cardinalitatea minimă 0:1, deci cheia externă va fi plasată în tabelul *LIBRARIE*. În cazul relației n:1, cheia externă este plasată în tabelul care se află de partea cu *n* (many) a relației. Un exemplu folosind baza de date creată este relație dintre tabelele *CARTE* și *SERIE* care are cardinalitatea maximă n:1, deci în tabelul *CARTE* va fi adăugată cheia externă din tabelul *SERIE*. În baza de date creată se găsesc și relații de tip m:n (many-to-many) care vor produce tabele asociative care vor conține cheile primare ale tabelelor în cauză, relațiile se sparg în relații de tip n:1, tabelele asociative fiind de partea cu *n* (many) cu fiecare dintre tabelele asociate. În acest mod s-au creat tabelele *ARE* (tabel asociativ de tip 3), *BON* și *STOC* (tabele asociative de tip 2).

În diagrama creată tabelele asociative sunt desenate punctat pentru a fi ușor de observat.

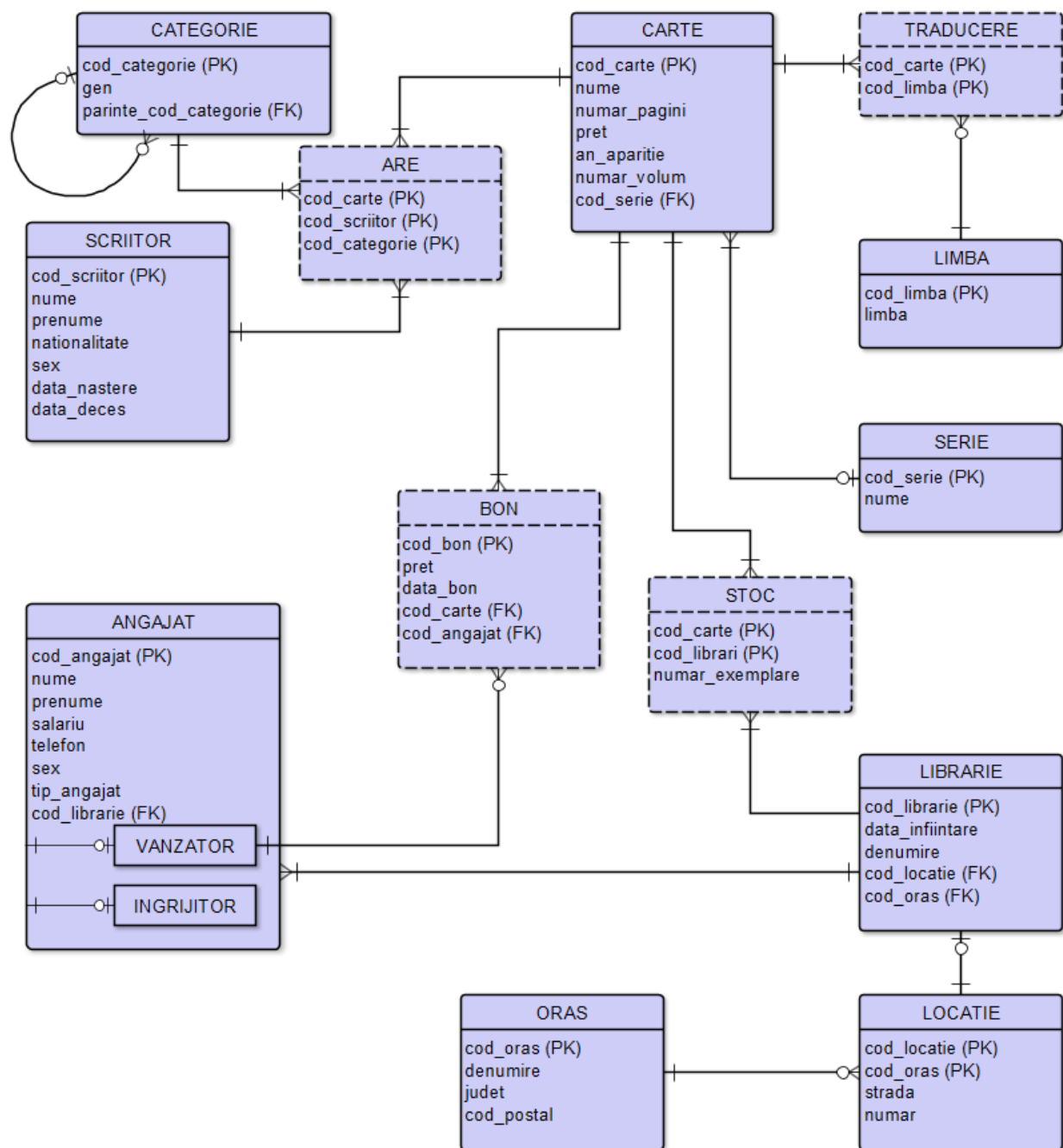


Diagrama conceptuală

## Design logic

*CATEGORIE* (cod\_categorie#, gen, parinte cod\_categorie)

*SCRIITOR* (cod\_scriitor#, nume, prenume, nationalitate, sex, data\_nastere, data\_deces)

*CARTE* (cod\_carte#, nume, numar\_pagini, pret, an\_aparitie, numar\_volum, cod\_serie)

*SERIE* (cod\_serie#, nume)

*LIMBA* (cod\_limba#, limba)

*ANGAJAT* (cod\_angajat#, nume, prenume, salariu, telefon, sex, tip\_angajat, cod\_librarie)

*VANZATOR* (cod\_angajat#, nume, prenume, salariu, telefon, sex, tip\_angajat, cod\_librarie)

*INGRIJITOR* (cod\_angajat#, nume, prenume, salariu, telefon, sex, tip\_angajat, cod\_librarie)

*LIBRARIE* (cod\_librarie#, data\_infiintare, denumire, cod\_locatie, cod\_oras)

*LOCATIE* (cod\_locatie#, cod\_oras#, strada, numar)

*ORAS* (cod\_oras#, denumire, judet, cod\_postal)

*ARE* (cod\_scriitor#, cod\_carte#, cod\_categorie#)

*TRADUCERE* (cod\_carte#, cod\_limba#)

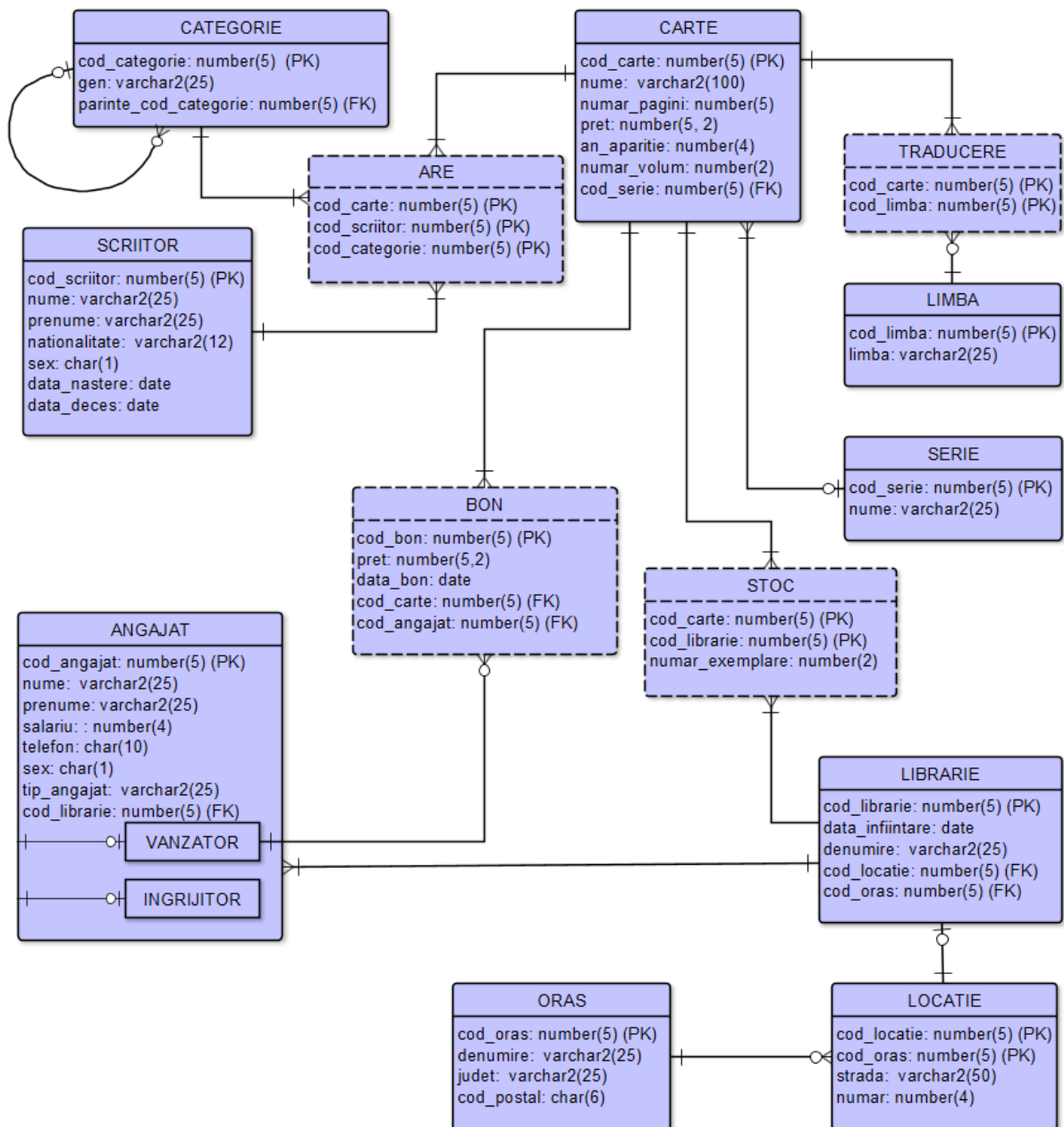
*BON* (cod\_bon #, cod\_carte, cod\_angajat, pret, data\_bon)

*STOC* (cod\_carte#, cod\_librarie#, nr\_exemplare)

*Notă:* Atributele urmate de diez (#) constituie cheile primare ale tabelelor, iar cele subliniate constituie chei externe.



## Design fizic



## Sistemul este în FN3

### Exemplu de atribut repetitiv care face ca un tabel relațional să nu fie în FN1

Se cere să se arate un exemplu de tabel din diagramă care nu este în FN1 și apoi să se aducă tabelul la FN1. Ca și definiție, o relație este în prima formă normală dacă fiecărui atribut care o compune îi corespunde o valoare indivizibilă (atomică).

#### Exemplu:

Se consideră că entitatea *CARTE* are ca atribute *cod\_carte*, *nume*, *număr\_pagini*, *preț*, *an\_apariție*, *număr\_volum*, *cod\_serie* și *limba*. O carte se poate găsi în librărie tradusă în una sau mai multe limbi.

cod_carte	nume	numar_pagini	pret	an_aparitie	numar_volum	cod_serie	limba
100	Invitatie la vals	336	30.6	1936	(null)	(null)	romana
101	Cu capul in nori	315	31	2010	(null)	(null)	romana
106	Locul de intalnire	318	27	1999	1	310	romana, engleza
116	Harry Potter si camera secretelor	400	48.48	1998	2	330	romana, engleza, franceza

Dacă se alege ca pentru fiecare limba să se creeze o nouă intrare cu toate celelalte atribute identice, ar însemna repetatea cheii primare care este unica (specifică pentru o carte anume). În același timp, nu se știe care este numărul maxim de limbi în care se poate traduce o carte. De aceea soluția este descompunerea tabelului în *CARTE* și *LIMBA* și crearea unui tabel asociativ *TRADUCERE* (cu cheie primară compusă): o carte poate fi tradusă în mai multe limbi și o limba poate fi folosită pentru a traduce mai multe cărți.

```
SELECT *
FROM carti
WHERE cod_carte IN (100, 101, 106, 116);
```

	COD_CARTE	NUME	NUMAR_PAGINI	PRET	AN_APARITIE	NUMAR_VOLUM	COD_SERIE
1	100	Invatatia la vals	336	30.6	1936	(null)	(null)
2	101	Cu capul in nori	315	31	2010	(null)	(null)
3	106	Locul de intalnire	318	27	1999	1	310
4	116	Harry Potter si camera secretelor	400	48.48	1998	2	330

*Datele date ca exemplu din tabela CARTE după ce aceasta a fost adusă la FN1*

```
SELECT *
FROM limbi;
```

	COD_LIMBA	LIMBA
1	10	romana
2	20	engleza
3	30	franceza
4	40	germana
5	50	rusa

*Datele din tabela LIMBA după ce aceasta a fost adusă la FN1*

```
SELECT *
FROM traduceri
WHERE cod_carte IN (100, 101, 106, 116);
```

	COD_CARTE	COD_LIMBA
1	100	10
2	101	10
3	106	10
4	106	20
5	116	10
6	116	20
7	116	30

*Datele date ca exemplu din tabela TRADUCERE*

### Exemplu de tabel relațional din diagramă care este în FN1, dar nu în FN2

O relație  $R$  este în a doua formă normală dacă și numai dacă:

- relația  $R$  este în FN1;
- fiecare atribut care nu este cheie (nu participă la cheia primară) este dependent de întreaga cheie primară.

Se ia ca exemplu tabelul *LOCATIE*:

cod_locatie	cod_oras	strada	numar	denumire	judet	cod_postal
90	1000	Strada Doamnei	20	Bucuresti	(null)	012581
100	1010	Strada Fierului	2	Oradea	Bihor	410001
110	1020	Strada Republicii	35	Cluj-Napoca	Cluj	010292
120	1030	Strada Avram Iancu	10	Iasi	Iasi	700028
130	1040	Strada Vasile Lupu	148	Drobeta Turnul Severin	Mehedinti	220036
140	1050	Strada Gheorghe Sincai	17	Voluntari	Ilfov	077191
150	1000	Strada Baltaretului	7	Bucuresti	(null)	012581

Fiecare atribut care nu este cheie (nu participă la cheia primară) este dependent de întreaga cheie primară - în cazul nostru atributele *stradă*, *număr*, *denumire*, *judet*, *cod\_poștal* nu sunt chei și trebuie să depindă direct de întreaga cheie primară *cod\_locatie#* și *cod\_oras#*. Aceste atribute nu depind direct de întreaga cheie primară deoarece se observa dependența directă dintre *cod\_oras#*, *denumire*, *judet* și *cod\_poștal*, însemnând ca *denumire*, *judet* și *cod\_poștal* depind direct doar de o parte a cheii primare, și anume, doar de *cod\_oras#*. Așadar relația nu se află în FN2.

Dependențe funcționale:

- $\{ \text{cod\_oras\#} \} \rightarrow \{ \text{denumire, judet, cod\_poștal} \}$   
(*cod\_oras* determină funcțional *denumire*, *judet*, *cod\_poștal*)

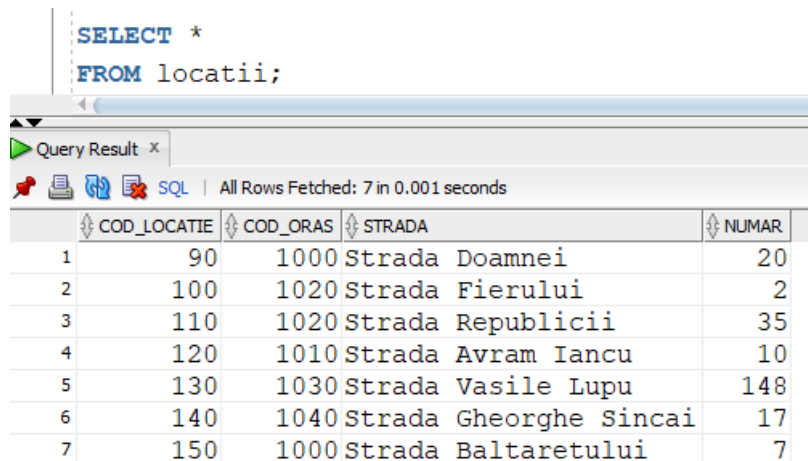
- $\{cod\_locație\#, cod\_oraș\# \} \rightarrow \{stradă, număr\}$

La final relația se va descompune în următoarele scheme relaționale:

*ORAS* (*cod\_oraș#*, *denumire*, *județ*, *cod\_poștal*)

*LOCATIE* (*cod\_locație#*, *cod\_oraș#*, *stradă*, *număr*)

```
SELECT *
FROM locatii;
```

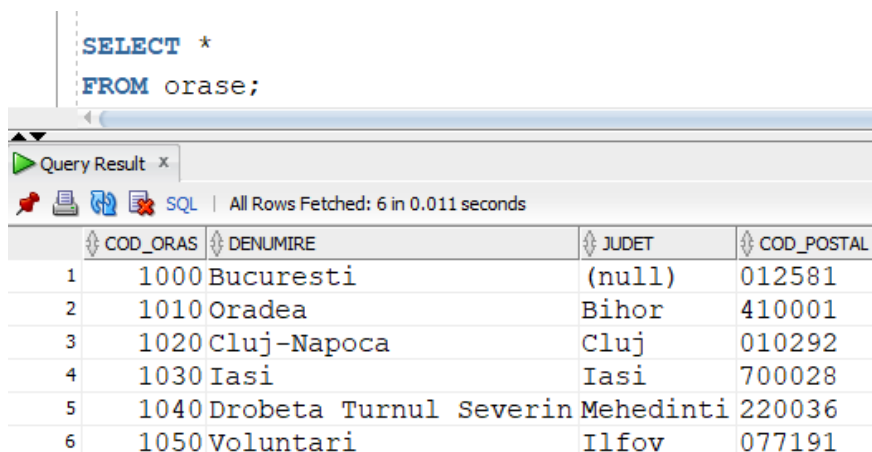


Query Result x  
All Rows Fetched: 7 in 0.001 seconds

	COD_LOCATIE	COD_ORAS	STRADA	NUMAR
1	90	1000	Strada Doamnei	20
2	100	1020	Strada Fierului	2
3	110	1020	Strada Republicii	35
4	120	1010	Strada Avram Iancu	10
5	130	1030	Strada Vasile Lupu	148
6	140	1040	Strada Gheorghe Sincai	17
7	150	1000	Strada Baltaretului	7

*Tabelul LOCATIE*

```
SELECT *
FROM orase;
```



Query Result x  
All Rows Fetched: 6 in 0.011 seconds

	COD_ORAS	DENUMIRE	JUDET	COD_POSTAL
1	1000	Bucuresti	(null)	012581
2	1010	Oradea	Bihor	410001
3	1020	Cluj-Napoca	Cluj	010292
4	1030	Iasi	Iasi	700028
5	1040	Drobeta Turnul Severin	Mehedinti	220036
6	1050	Voluntari	Ilfov	077191

*Tabelul ORAS*

**Exemplu de tabel relațional din diagramă care este în FN2, dar nu în FN3**

O relație *R* este în a treia formă normală dacă și numai dacă:

- relația  $R$  este în FN2;
- fiecare atribut care nu este cheie (nu participă la o cheie) depinde direct de cheia primară.

Se ia ca exemplu tabelul *ANGAJAT*:

cod_angajat	nume	prenume	salariu	telefon	sex	tip_angajat	cod_librarie	data_infiintare	denumire	cod_locatie	cod_oras
400	Popescu	Maria	1275	0761234567	f	vanzator	40	10-OCT-15	Libraria Iulius	90	1000
401	Popa	Marin	2450	0761236666	m	vanzator	60	04-JUN-17	Libraria Iulius	130	1030
402	Ionescu	Adiel	2725	0769999967	m	vanzator	70	10-AUG-10	Libraria Iulius	120	1010
403	Marinescu	Claudiu	1600	0761234000	m	vanzator	80	16-MAY-08	Libraria Iulius	110	1020
404	Ionescu	Mircea	1550	0731114567	m	ingrijitor	80	16-MAY-08	Libraria Iulius	110	1020
405	Georgescu	Aurica	1500	0724444413	f	ingrijitor	70	10-AUG-10	Libraria Iulius	120	1010

În exemplul atașat anterior se observă că attributele *data\_infiintare* (care se referă la librărie), *denumire*, *cod\_locatie*, *cod\_oras* depind tranzitiv de cheia primară *cod\_angajat#* prin intermediul atributului *cod\_librarie*.

Dependențe funcționale:

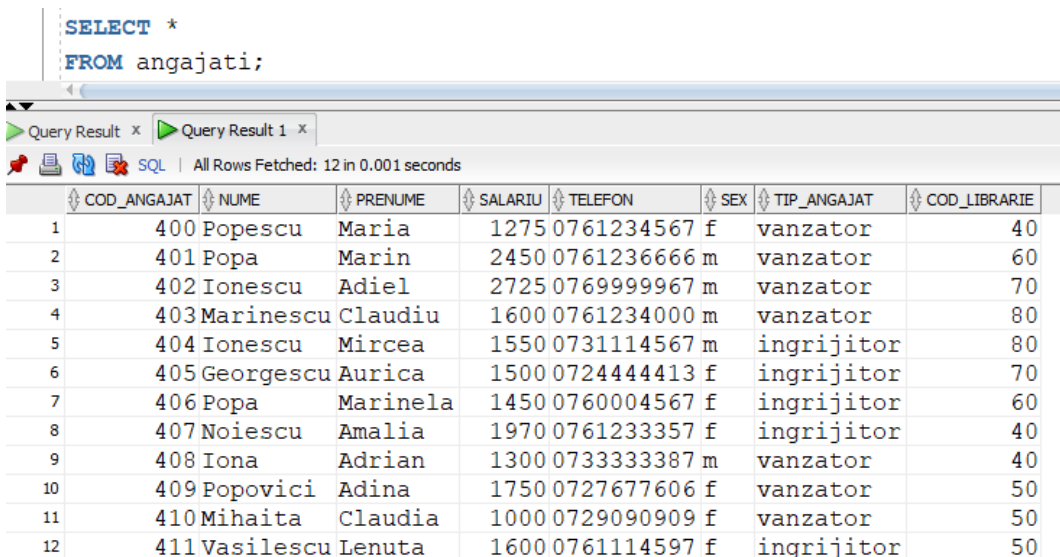
- $\{ \text{cod\_angajat\#} \} \rightarrow \{ \text{nume, prenume, salariu, telefon, sex, tip\_angajat, cod\_librărie} \}$   
(*cod\_angajat* determină funcțional attributele selectate)
- $\{ \text{cod\_angajat\#} \} \rightarrow \{ \text{cod\_librărie} \} \rightarrow \{ \text{data\_înființare, denumire, cod\_locatie, cod\_oras} \}$

Pentru a aduce relația în FN3 se aplică Casey-Delobel. Relația se descompune, prin eliminarea dependențelor funcționale tranzitive, în proiecțiile:

*ANGAJAT (cod\_angajat#, nume, prenume, salariu, telefon, sex, tip\_angajat, cod\_librărie)*

*LIBRARIE (cod\_librărie#, data\_înființare, denumire, cod\_locație, cod\_orăș)*

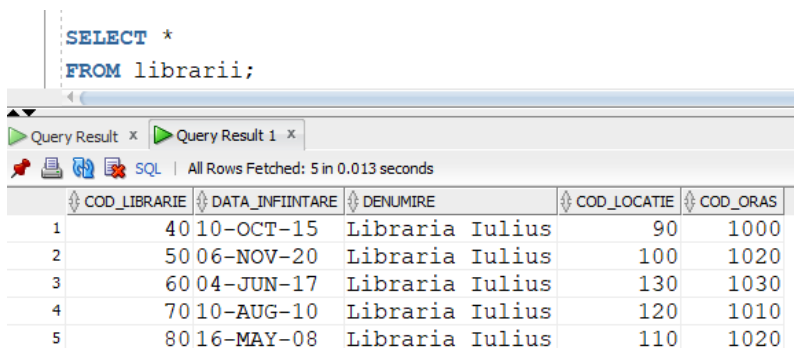
```
SELECT *
FROM angajati;
```



	COD_ANGAJAT	NUME	PRENUME	SALARIU	TELEFON	SEX	TIP_ANGAJAT	COD_LIBRARIE
1	400	Popescu	Maria	1275	0761234567	f	vanzator	40
2	401	Popa	Marin	2450	0761236666	m	vanzator	60
3	402	Ionescu	Adiel	2725	0769999967	m	vanzator	70
4	403	Marinescu	Claudiu	1600	0761234000	m	vanzator	80
5	404	Ionescu	Mircea	1550	0731114567	m	ingrijitor	80
6	405	Georgescu	Aurica	1500	0724444413	f	ingrijitor	70
7	406	Popa	Marinela	1450	0760004567	f	ingrijitor	60
8	407	Noiescu	Amalia	1970	0761233357	f	ingrijitor	40
9	408	Iona	Adrian	1300	0733333387	m	vanzator	40
10	409	Popovici	Adina	1750	0727677606	f	vanzator	50
11	410	Mihaita	Claudia	1000	0729090909	f	vanzator	50
12	411	Vasilescu	Lenuta	1600	0761114597	f	ingrijitor	50

*Tabelul ANGAJAT*

```
SELECT *
FROM librarii;
```



	COD_LIBRARIE	DATA_INFIINTARE	DENUMIRE	COD_LOCATIE	COD_ORAS
1	40	10-OCT-15	Libraria Iulius	90	1000
2	50	06-NOV-20	Libraria Iulius	100	1020
3	60	04-JUN-17	Libraria Iulius	130	1030
4	70	10-AUG-10	Libraria Iulius	120	1010
5	80	16-MAY-08	Libraria Iulius	110	1020

*Tabelul LIBRARIE*

## Implementarea tabelelor și adăugarea de informații

### Implementarea tabelelor

```
CREATE TABLE CATEGORII (  
    cod_categorie NUMBER(5) CONSTRAINT pkey_categorie PRIMARY KEY,  
    parinte_cod_categorie NUMBER(5) CONSTRAINT fkey_parinte_categorie  
REFERENCES categorii (cod_categorie),  
    gen VARCHAR2(25) CONSTRAINT gen NOT NULL  
);  
  
CREATE TABLE SERII (  
    cod_serie NUMBER(5) CONSTRAINT pkey_serie PRIMARY KEY,  
    nume VARCHAR2(25) NOT NULL  
);  
  
CREATE TABLE CARTI (  
    cod_carte NUMBER(5) CONSTRAINT pkey_carte PRIMARY KEY,  
    nume VARCHAR2(100) CONSTRAINT denumire_carte NOT NULL,  
    numar_pagini NUMBER(5),  
    pret NUMBER(5,2),  
    an_aparitie NUMBER(4),  
    numar_volum NUMBER(2) DEFAULT NULL,  
    cod_serie NUMBER(5) CONSTRAINT fkey_carte_serie REFERENCES serii  
(cod_serie)  
);  
  
CREATE TABLE LIMBI (
```



```

        cod_limba NUMBER(5) CONSTRAINT pkey_limba PRIMARY KEY,
        limba VARCHAR2(25) NOT NULL
    );

CREATE TABLE SCRIITORI(
    cod_scriitor NUMBER(5) CONSTRAINT pkey_scriitor PRIMARY KEY,
    nume VARCHAR2(25) NOT NULL,
    prenume VARCHAR2(25),
    nationalitate VARCHAR2(12),
    sex CHAR(1),
    data_nastere DATE DEFAULT NULL,
    data_deces DATE DEFAULT NULL
);

CREATE TABLE ORASE (
    cod_oras NUMBER(5) CONSTRAINT pkey_oras PRIMARY KEY,
    denumire VARCHAR2(25) NOT NULL,
    judet VARCHAR2(25),
    cod_postal CHAR(6)
);

CREATE TABLE LOCATII (
    cod_locatie NUMBER(5),
    cod_oras NUMBER(5) CONSTRAINT fk_locatie_oras REFERENCES orase
(cod_oras),
    CONSTRAINT pk_compus PRIMARY KEY(cod_locatie, cod_oras),

```

```

        strada VARCHAR2(50) NOT NULL,

        numar NUMBER(4)

    );

CREATE TABLE LIBRARII(

    cod_librarie NUMBER(5) CONSTRAINT pk_librarie PRIMARY KEY,

    data_infiintare DATE DEFAULT sysDATE,

    denumire VARCHAR2(25) NOT NULL,

    cod_locatie NUMBER(5),

    cod_oras NUMBER(5),

    CONSTRAINT fk_compus foreign KEY(cod_locatie, cod_oras) REFERENCES
    locatii (cod_locatie, cod_oras)

);

CREATE TABLE ANGAJATI(

    cod_angajat NUMBER(5) CONSTRAINT pkey_angajat PRIMARY KEY,

    nume VARCHAR2(25) NOT NULL,

    prenume VARCHAR2(25),

    salariu NUMBER(4),

    telefon CHAR(10),

    sex CHAR(1),

    tip_angajat VARCHAR2(25) NOT NULL,

    cod_librarie NUMBER(5) CONSTRAINT fkey_ang_librarie REFERENCES
    librarii (cod_librarie)

);

CREATE TABLE BONURI (

```

```

        cod_bon NUMBER(5) CONSTRAINT pk_bon PRIMARY KEY,

        pret NUMBER(5,2),

        data_bon DATE DEFAULT sysDATE,

        cod_carte      NUMBER(5)      CONSTRAINT      fk_bon_carte      REFERENCES
carti(cod_carte),

        cod_angajat  NUMBER(5)  CONSTRAINT  fk_bon_ang  REFERENCES  angajati
(cod_angajat)

);

CREATE TABLE ARE (

        cod_scriitor NUMBER(5),

        cod_carte NUMBER(5),

        cod_categorie NUMBER(5),

        CONSTRAINT fk_are_scriitor foreign KEY(cod_scriitor) REFERENCES
scriitori (cod_scriitor),

        CONSTRAINT fk_are_carte foreign KEY(cod_carte) REFERENCES  carti
(cod_carte),

        CONSTRAINT fk_are_categorie foreign KEY(cod_categorie) REFERENCES
categorii (cod_categorie),

        CONSTRAINT pk_compus_are PRIMARY KEY(cod_scriitor, cod_carte,
cod_categorie)

);

CREATE TABLE STOCURI (

        cod_carte NUMBER(5),

        cod_librarie NUMBER(5),

        nr_exemplare NUMBER(2),

        CONSTRAINT fk_stoc_carte foreign KEY(cod_carte) REFERENCES  carti
(cod_carte),

```

```

        CONSTRAINT fk_stoc_librarie foreign KEY(cod_librarie) REFERENCES
        librarii (cod_librarie),

        CONSTRAINT pk_compus_stoc PRIMARY KEY(cod_carte, cod_librarie)
);

CREATE TABLE TRADUCERI (

        cod_carte      NUMBER(5)      CONSTRAINT      fk_trad_carte      REFERENCES
        carti(cod_carte),

        cod_limba      NUMBER(5)      CONSTRAINT      fk_trad_limba      REFERENCES
        limbi(cod_limba),

        CONSTRAINT pk_compus_trad PRIMARY KEY(cod_carte, cod_limba)
);

```

### Adăugare de înregistrări în fiecare tabel creat

```

----- SCRIITORI -----

CREATE SEQUENCE SEQ_SCRIITORI

INCREMENT by 1

START WITH 100

MAXVALUE 10000

NOCYCLE;

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Oke', 'Janette', 'Canadiana', 'f',
to_date('18-02-1935','dd-mm-yyyy'), null);

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Dickens', 'Charles', 'Britanica', 'm',
to_date('07-02-1812','dd-mm-yyyy'), to_date('09-06-1870','dd-mm-
yyyy'));

```

```

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Witemeyer', 'Karen', 'Americana', 'f',
null, null);

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Drumes', 'Mihail', 'Romana', 'm',
to_date('26-11-1901','dd-mm-yyyy'), to_date('07-02-1982','dd-mm-
YYYY'));

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Bunn', 'T. Davis', 'Americana',
'm',to_date('01-01-1952','dd-mm-yyyy'), null);

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Rowling', 'J.K.', 'Britanica',
'f',to_date('31-07-1965','dd-mm-yyyy'), null);

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Dumas', 'Alexandre', 'Franceza',
'm',to_date('24-07-1802','dd-mm-yyyy'), to_date('05-12-1870','dd-mm-
YYYY'));

INSERT INTO scriitori

VALUES (SEQ_SCRIITORI.NEXTVAL, 'Tolstoi', 'Lev', 'Rusa',
'm',to_date('09-09-1828','dd-mm-yyyy'), to_date('20-11-1910','dd-mm-
YYYY'));

COMMIT;

----- SERII -----

CREATE SEQUENCE SEQ_SERII

INCREMENT by 10

START WITH 300

MAXVALUE 10000

NOCYCLE;

```

```

INSERT INTO serii
VALUES (SEQ_SERII.NEXTVAL, 'Anotimpurile inimii');

INSERT INTO serii
VALUES (SEQ_SERII.NEXTVAL, 'Cantecul Acadiei');

INSERT INTO serii
VALUES (SEQ_SERII.NEXTVAL, 'Faptele credintei');

INSERT INTO serii
VALUES (SEQ_SERII.NEXTVAL, 'Harry Potter');

INSERT INTO serii
VALUES (SEQ_SERII.NEXTVAL, 'Cei trei muschetari');

COMMIT;

```

----- CARTI -----

```

CREATE SEQUENCE SEQ_CARTI

INCREMENT by 1

START WITH 100

MAXVALUE 10000

NOCYCLE;

INSERT INTO carti
VALUES (SEQ_CARTI.NEXTVAL, 'Invitatie la vals', 336, 30.60, 1936, null,
null);

INSERT INTO carti
VALUES (SEQ_CARTI.NEXTVAL, 'Cu capul in nori', 315, 31, 2010, null,
null);

INSERT INTO carti
VALUES (SEQ_CARTI.NEXTVAL, 'Mireasa pe masura', 295, 28.50, 2010, null,

```

```

null);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Marile sperante',544, 30, 1861, null,
null);

--seria Cantecul Acadiei -oke si bunn

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Mostenirea', 294 , 27, 2001, 3, 310);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Binecuvantatul tarm',286, 28.5 , 1936, 2,
310);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Locul de intalnire',318, 27, 1999, 1, 310);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Limanul mult dorit',324, 27, 2000, 5 ,
310);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Un far calauzitor',256, 27, 2002, 4, 310);

-- serie scrisa de Oke

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'A fost odata intr-o vara',224, 21.85, 1981,
1, 300);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Promisiunea unei noi primaveri',222, 21.85,
1989, 4, 300);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Vanturi tomnatice',220, 21.85, 1987, 2,

```

```

300);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Iarna nu tine o vesnicie',216, 21.85, 1988,
3, 300);

--seria scrisa de oke si bunn

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Sotia Centurionului',321, 35, 2010, 1,
320);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Flacara ascunsa',388, 35, 2011, 2, 320);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Drumul spre Damasc',432, 35, 2011, 3, 320);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Harry Potter si camera secretelor',400,
48.48, 1998, 2, 330);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Harry Potter si pocalul de foc',728, 58,
2000, 4, 330);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Harry Potter si printul semisange', 650,
64.64, 2005, 6, 330);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Harry Potter si talismanele mortii',784,
64.64, 2007, 7, 330);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Harry Potter si ordinul phoenix',990,
64.64, 2003, 5, 330);

```



```

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Harry Potter si piatra filosofala', 532, 42,
1997, 1, 330);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Harry Potter si prizonierul din
Azkaban', 464, 48.48, 1999, 3, 330);


INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Cei trei muschetari', 224, 21, 1844, 1,
340);

INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'Dupa 20 de ani', 622, 33, 1845, 2, 340);


INSERT INTO carti

VALUES (SEQ_CARTI.NEXTVAL, 'O mie si una de nopti', 4800, 199, null,
null, null);

COMMIT;

----- CATEGORII -----

CREATE SEQUENCE SEQ_CATEG

INCREMENT by 1

START WITH 10

MAXVALUE 10000

NOCYCLE;

INSERT INTO categorii VALUES (SEQ_CATEG.NEXTVAL, 'fictiune', NULL);

```

```

INSERT INTO categorii VALUES (SEQ_CATEG.NEXTVAL, 'non-fictiune', NULL);
INSERT INTO categorii VALUES (SEQ_CATEG.NEXTVAL, 'romantic', 10);
INSERT INTO categorii VALUES (SEQ_CATEG.NEXTVAL, 'istoric', 11);
INSERT INTO categorii VALUES (SEQ_CATEG.NEXTVAL, 'crestin', 10);
INSERT INTO categorii VALUES (SEQ_CATEG.NEXTVAL, 'aventura', 10);
COMMIT;

```

```

----- ARE -----

```

```

--Janette Oke 100 si Bunn - 104 --seria cantecul Acadiei

```

```

--cod carti: 104 - 108

```

```

--gen : fictiune 10, crestin 14

```

```

INSERT INTO are VALUES (100,104,10);

```

```

INSERT INTO are VALUES (100,105,10);

```

```

INSERT INTO are VALUES (100,106,10);

```

```

INSERT INTO are VALUES (100,107,10);

```

```

INSERT INTO are VALUES (100,108,10);

```

```

INSERT INTO are VALUES (100,104,14);

```

```

INSERT INTO are VALUES (100,105,14);

```

```

INSERT INTO are VALUES (100,106,14);

```

```

INSERT INTO are VALUES (100,107,14);

```

```

INSERT INTO are VALUES (100,108,14);

```

```

INSERT INTO are VALUES (104,104,10);

```

```

INSERT INTO are VALUES (104,105,10);

```

```

INSERT INTO are VALUES (104,106,10);

```

```

INSERT INTO are VALUES(104,107,10);
INSERT INTO are VALUES(104,108,10);

INSERT INTO are VALUES(104,104,14);
INSERT INTO are VALUES(104,105,14);
INSERT INTO are VALUES(104,106,14);
INSERT INTO are VALUES(104,107,14);
INSERT INTO are VALUES(104,108,14);

--Janette Oke 100 Seria Anotimpurile inimii(cod carti)
--cod carti: 109 - 112
--gen: non-fictiune 11, romantic 12, istoric 13, crestin 14
INSERT INTO are VALUES(100,109,14);
INSERT INTO are VALUES(100,110,14);
INSERT INTO are VALUES(100,111,14);
INSERT INTO are VALUES(100,112,14);

INSERT INTO are VALUES(100,109,12);
INSERT INTO are VALUES(100,110,12);
INSERT INTO are VALUES(100,111,12);
INSERT INTO are VALUES(100,112,12);

INSERT INTO are VALUES(100,109,13);
INSERT INTO are VALUES(100,110,13);
INSERT INTO are VALUES(100,111,13);
INSERT INTO are VALUES(100,112,13);

```

```

INSERT INTO are VALUES(100,109,11);
INSERT INTO are VALUES(100,110,11);
INSERT INTO are VALUES(100,111,11);
INSERT INTO are VALUES(100,112,11);

-- Karen Witemeyer 102
--cod carti: 101, 102
--gen: fictiune 10, romantic 12, crestin 14
INSERT INTO are VALUES(102,101,12);
INSERT INTO are VALUES(102,102,12);

INSERT INTO are VALUES(102,101,10);
INSERT INTO are VALUES(102,102,10);

INSERT INTO are VALUES(102,101,14);
INSERT INTO are VALUES(102,102,14);

--Oke(100) si Bun(104) - Seria Faptele credintei(320)
--cod carti: 113, 114, 115
-- non-fictiune 11, romantic 12, istoric 13, crestin 14
-- gen: crestin 14, aventura 15
INSERT INTO are VALUES(100, 113, 14);
INSERT INTO are VALUES(100, 114, 14);
INSERT INTO are VALUES(100, 115, 14);
INSERT INTO are VALUES(100, 113, 15);

```

```

INSERT INTO are VALUES(100, 114, 15);
INSERT INTO are VALUES(100, 115, 15);

INSERT INTO are VALUES(104, 113, 14);
INSERT INTO are VALUES(104, 114, 14);
INSERT INTO are VALUES(104, 115, 14);
INSERT INTO are VALUES(104, 113, 15);
INSERT INTO are VALUES(104, 114, 15);
INSERT INTO are VALUES(104, 115, 15);
COMMIT;

--Drumes 103
--cod carti: 100
--gen: non-fictiune 11, romantic 12
INSERT INTO are VALUES(103,100,12);
INSERT INTO are VALUES(103,100,11)

--Harry potter - autoare 105
--cod carti: 116 - 122
--gen: fictiune 10
INSERT INTO are VALUES(105,116,10);
INSERT INTO are VALUES(105,117,10);
INSERT INTO are VALUES(105,118,10);
INSERT INTO are VALUES(105,119,10);
INSERT INTO are VALUES(105,120,10);
INSERT INTO are VALUES(105,121,10);

```

```

INSERT INTO are VALUES(105,122,10);

--Dumas 106
--cod carti: 123, 124
--gen: istoric 13, aventura 15
INSERT INTO are VALUES(106, 123, 13);
INSERT INTO are VALUES(106, 123, 15);
INSERT INTO are VALUES(106, 124, 13);
INSERT INTO are VALUES(106, 124, 15);

--Dickens 101
--cod carti: 103
--gen: fictiune 10, aventura 15
INSERT INTO are VALUES(101, 103, 10);
INSERT INTO are VALUES(101, 103, 15);
COMMIT;

----- ORASE -----

CREATE SEQUENCE SEQ_ORASE
INCREMENT by 10
START WITH 1000
MAXVALUE 10000
NOCYCLE;

INSERT INTO orase VALUES(SEQ_ORASE.NEXTVAL, 'Bucuresti', null,
'012581');
INSERT INTO orase VALUES(SEQ_ORASE.NEXTVAL, 'Oradea', 'Bihor',

```

```

'410001');

INSERT INTO orase VALUES (SEQ_ORASE.NEXTVAL, 'Cluj-Napoca', 'Cluj',
'010292');

INSERT INTO orase VALUES (SEQ_ORASE.NEXTVAL, 'Iasi', 'Iasi', '700028');

INSERT INTO orase VALUES (SEQ_ORASE.NEXTVAL, 'Drobeta Turnul Severin',
'Mehedinti', '220036');

INSERT INTO orase VALUES (SEQ_ORASE.NEXTVAL, 'Voluntari', 'Ilfov',
'077191');

COMMIT;

----- LOCATII -----

CREATE SEQUENCE SEQ_LOC

INCREMENT by 10

START WITH 90

MAXVALUE 1000

NOCYCLE;

INSERT INTO locatii VALUES (SEQ_LOC.NEXTVAL, 1000, 'Strada Doamnei',
20);

INSERT INTO locatii VALUES (SEQ_LOC.NEXTVAL, 1020, 'Strada Fierului',
2);

INSERT INTO locatii VALUES (SEQ_LOC.NEXTVAL, 1020, 'Strada Republicii',
35);

INSERT INTO locatii VALUES (SEQ_LOC.NEXTVAL, 1010, 'Strada Avram
Iancu', 10);

INSERT INTO locatii VALUES (SEQ_LOC.NEXTVAL, 1030, 'Strada Vasile
Lupu', 148);

INSERT INTO locatii VALUES (SEQ_LOC.NEXTVAL, 1040, 'Strada Gheorghe
Sincai', 17);

INSERT INTO locatii VALUES (SEQ_LOC.NEXTVAL, 1000, 'Strada

```

```
Baltaretului', 7);
```

```
COMMIT;
```

```
----- LIBRARII -----
```

```
CREATE SEQUENCE SEQ_LIBR
```

```
INCREMENT by 10
```

```
START WITH 40
```

```
MAXVALUE 1000
```

```
NOCYCLE;
```

```
INSERT INTO librarii VALUES (SEQ_LIBR.NEXTVAL,to_date('10-10-2015','dd-mm-yyyy'), 'Libraria Iulius',90, 1000);
```

```
INSERT INTO librarii VALUES (SEQ_LIBR.NEXTVAL,to_date('06-11-2020','dd-mm-yyyy'), 'Libraria Iulius',100, 1020);
```

```
INSERT INTO librarii VALUES (SEQ_LIBR.NEXTVAL,to_date('04-06-2017','dd-mm-yyyy'), 'Libraria Iulius',130, 1030);
```

```
INSERT INTO librarii VALUES (SEQ_LIBR.NEXTVAL,to_date('10-08-2010','dd-mm-yyyy'), 'Libraria Iulius',120, 1010);
```

```
INSERT INTO librarii VALUES (SEQ_LIBR.NEXTVAL,to_date('16-05-2008','dd-mm-yyyy'), 'Libraria Iulius',110, 1020);
```

```
COMMIT;
```

```
----- ANGAJATI -----
```

```
CREATE SEQUENCE SEQ_ANG
```

```
INCREMENT by 1
```

```
START WITH 400
```

```
MAXVALUE 10000
```

```
NOCYCLE;
```



```

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Popescu', 'Maria', 1275,
'0761234567','f','vanzator',40);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Popa', 'Marin', 2450,
'0761236666','m','vanzator',60);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Ionescu', 'Adiel', 2725,
'0769999967','m','vanzator',70);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Marinescu', 'Claudiu',
1600, '0761234000','m','vanzator',80);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Ionescu', 'Mircea',
1550, '0731114567','m','ingrijitor',80);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Georgescu', 'Aurica',
1500, '0724444413','f','ingrijitor',70);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Popa', 'Marinela', 1450,
'0760004567','f','ingrijitor',60);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Noiescu', 'Amalia',
1970, '0761233357','f','ingrijitor',40);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Iona', 'Adrian', 1300,
'0733333387','m','vanzator',40);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Popovici', 'Adina',
1750, '0727677606','f','vanzator',50);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Mihaita', 'Claudia',
1000, '0729090909','f','vanzator',50);

INSERT INTO angajati VALUES(SEQ_ANG.NEXTVAL, 'Vasilescu', 'Lenuta',
1600, '0761114597','f','ingrijitor',50);

COMMIT;

```

----- BONURI -----

```

CREATE SEQUENCE SEQ_CUMP

INCREMENT by 1

START WITH 100

```

```

MAXVALUE 1000

NOCYCLE;

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 12.2, to_date('20-04-2023
11:30', 'dd-mm-yyyy hh24:mi'),100,400);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 23.1, to_date('20-07-2022
21:17', 'dd-mm-yyyy hh24:mi'),115,408);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 32.00, to_date('20-07-
2022 21:17', 'dd-mm-yyyy hh24:mi'),114,408);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 43.99, to_date('20-07-
2022 21:17', 'dd-mm-yyyy hh24:mi'),113,408);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 22, to_date('04-10-2011
13:55', 'dd-mm-yyyy hh24:mi'),103,401);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 43, to_date('25-03-2023
11:30', 'dd-mm-yyyy hh24:mi'),104,409);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 28.5, to_date('25-03-2023
17:00', 'dd-mm-yyyy hh24:mi'),102,410);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 27, to_date('26-02-2023
10:00', 'dd-mm-yyyy hh24:mi'),104,409);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 48.48, to_date('13-10-
2022 16:44', 'dd-mm-yyyy hh24:mi'),116,410);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 21, to_date('29-08-2020
14:02', 'dd-mm-yyyy hh24:mi'),123,410);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL , 28.5, to_date('15-09-2021
15:17', 'dd-mm-yyyy hh24:mi'),105,402);

INSERT INTO bonuri VALUES(SEQ_CUMP.NEXTVAL, 35, to_date('15-08-2022
15:17', 'dd-mm-yyyy hh24:mi'),115,400);

COMMIT;

----- STOCURI -----

INSERT INTO stocuri values(100, 40, 2);

```

```
INSERT INTO stocuri values(115, 40, 3);
INSERT INTO stocuri values(114, 40, 1);
INSERT INTO stocuri values(113, 40, 4);
INSERT INTO stocuri values(103, 80, 3);
INSERT INTO stocuri values(104, 60, 5);
INSERT INTO stocuri values(102, 50, 2);
INSERT INTO stocuri values(104, 80, 6);
INSERT INTO stocuri values(116, 50, 4);
INSERT INTO stocuri values(123, 50, 4);
INSERT INTO stocuri values(105, 70, 3);
INSERT INTO stocuri values(100, 60, 2);
INSERT INTO stocuri values(101, 80, 1);
INSERT INTO stocuri values(102, 40, 2);
INSERT INTO stocuri values(103, 60, 3);
INSERT INTO stocuri values(104, 50, 6);
INSERT INTO stocuri values(105, 50, 4);
INSERT INTO stocuri values(106, 50, 4);
INSERT INTO stocuri values(107, 50, 5);
INSERT INTO stocuri values(108, 50, 4);
INSERT INTO stocuri values(109, 80, 3);
INSERT INTO stocuri values(110, 80, 2);
INSERT INTO stocuri values(111, 80, 3);
INSERT INTO stocuri values(112, 80, 5);
INSERT INTO stocuri values(103, 50, 6);
INSERT INTO stocuri values(114, 50, 7);
INSERT INTO stocuri values(115, 50, 10);
```

```

INSERT INTO stocuri values(116, 70, 12);
INSERT INTO stocuri values(117, 70, 11);
INSERT INTO stocuri values(118, 70, 2);
INSERT INTO stocuri values(119, 70, 7);
INSERT INTO stocuri values(120, 70, 8);
INSERT INTO stocuri values(121, 70, 4);
INSERT INTO stocuri values(122, 70, 6);
INSERT INTO stocuri values(123, 60, 8);
INSERT INTO stocuri values(124, 60, 9);
INSERT INTO stocuri values(109, 40, 4);
INSERT INTO stocuri values(109, 60, 5);
INSERT INTO stocuri values(123, 70, 9);
INSERT INTO stocuri values(124, 70, 6);
COMMIT;

```

----- LIMBI -----

```

CREATE SEQUENCE SEQ_LB
INCREMENT by 10
START WITH 10
MAXVALUE 10000
NOCYCLE;

```

```

insert into limbi values (SEQ_LB.NEXTVAL,'romana');
insert into limbi values (SEQ_LB.NEXTVAL,'engleza');
insert into limbi values (SEQ_LB.NEXTVAL,'franceza');
insert into limbi values (SEQ_LB.NEXTVAL,'germana');

```

```
insert into limbi values (SEQ_LB.NEXTVAL,'rusa');  
COMMIT;
```

```
----- TRADUCERI -----
```

```
insert into traduceri values(100,10);  
insert into traduceri values(101,10);  
insert into traduceri values(102,10);  
insert into traduceri values(103,10);  
insert into traduceri values(104,10);  
insert into traduceri values(105,10);  
insert into traduceri values(106,10);
```

```
insert into traduceri values(104,20);  
insert into traduceri values(105,20);  
insert into traduceri values(106,20);  
insert into traduceri values(107,20);  
insert into traduceri values(108,20);
```

```
insert into traduceri values(109,10);  
insert into traduceri values(110,10);  
insert into traduceri values(111,10);  
insert into traduceri values(112,10);
```

```
insert into traduceri values(113,10);  
insert into traduceri values(114,10);  
insert into traduceri values(115,10);
```

```
insert into traduceri values(113,20);  
insert into traduceri values(114,20);  
insert into traduceri values(115,20);
```

```
insert into traduceri values(116,10);  
insert into traduceri values(117,10);  
insert into traduceri values(118,10);  
insert into traduceri values(119,10);  
insert into traduceri values(120,10);  
insert into traduceri values(121,10);  
insert into traduceri values(122,10);
```

```
insert into traduceri values(116,20);  
insert into traduceri values(117,20);  
insert into traduceri values(118,20);  
insert into traduceri values(119,20);  
insert into traduceri values(120,20);  
insert into traduceri values(121,20);  
insert into traduceri values(122,20);
```

```
insert into traduceri values(116,30);  
insert into traduceri values(117,30);  
insert into traduceri values(118,30);  
insert into traduceri values(119,30);  
insert into traduceri values(120,30);
```

```
insert into traduceri values(121,30);  
insert into traduceri values(122,30);  
  
insert into traduceri values(123,10);  
insert into traduceri values(124,10);  
  
insert into traduceri values(123,30);  
insert into traduceri values(124,30);  
  
insert into traduceri values(103,10);  
COMMIT;
```

## Interogări

Se vor prezenta 15 interogări complexe. Pentru fiecare în parte se va preciza ce aspecte învățate au fost folosite pentru rezolvarea ei, cerința în limbaj natural, apoi rezolvarea și rezultatul rulării interogării în urma inserării înregistrărilor la punctul precedent.

**Interogare 1** (*GROUP BY, ORDER BY, LOWER, NVL, COUNT, INNER JOIN, subinterogare în clauza WHERE*)

Determinați cărțile vândute în Cluj (județ) în ordinea numărului de exemplare vândute, de la bestseller până la cea care este vândută de cele mai puține ori. Se menționează numărul de exemplare vândute și titlul acestora.

```
SELECT COUNT(cod_carte) NUMAR_CARTI_VANDUTE, nume TITLU
FROM bonuri INNER JOIN carti USING (cod_carte)
WHERE cod_angajat IN (
    SELECT cod_angajat
    FROM angajati
    WHERE LOWER(tip_angajat) = 'vanzator'
    AND cod_librarie IN (
        SELECT cod_librarie
        FROM librarii
        WHERE cod_oras IN (
            SELECT cod_oras
            FROM orase
            WHERE NVL(LOWER(judet), '-') = 'cluj'
        )
    )
)
```



```
GROUP BY cod_carte, nume
ORDER BY COUNT(cod_carte) DESC;
```

	NUMAR_CARTI_VANDUTE	TITLU
1	4	Mostenirea
2	2	Limanul mult dorit
3	1	Harry Potter si camera secretelor
4	1	Mireasa pe masura
5	1	Cei trei muschetari

## Interogare 2 (SUM, MIN, GROUP BY, HAVING, LEFT JOIN, subinterogare în clauza HAVING)

Să se determine codul librăriei cu cea mai mică sumă câștigată în urma vânzărilor.

```
WITH profit AS (
    SELECT pret, cod_librarie
    FROM bonuri LEFT JOIN angajati USING (cod_angajat)
                LEFT JOIN librarii USING (cod_librarie)
)
SELECT cod_librarie, SUM(pret) suma
FROM profit
GROUP BY cod_librarie
HAVING SUM(pret) = (
    SELECT MIN(SUM(pret)) profit_minin
    FROM profit
    GROUP BY cod_librarie
);
```

	COD_LIBRARIE	SUMA
1	60	22

### Interogare 3 (CASE, INNER JOIN, INSTR, LOWER, COUNT, subinterogare în clauza SELECT)

Unde lucrează și câte vânzări a efectuat un vânzător dat de la tastatură.

```
ACCEPT nume PROMPT 'Introduceti numele si prenumele angajatului: '

SELECT  ang.numa,
        ang.prenume,
        CASE
            WHEN o.judet IS NOT NULL THEN o.judet
            ELSE o.denumire
        END LUCREAZA_IN,
        (
            SELECT COUNT(*)
            FROM bonuri
            WHERE cod_angajat = ang.cod_angajat
        ) VANZARI
FROM angajati ang
        JOIN librarii lib ON (ang.cod_librarie = lib.cod_librarie)
        JOIN locatii loc ON (lib.cod_locatie = loc.cod_locatie)
        JOIN orase o ON (loc.cod_oras = o.cod_oras)
WHERE INSTR(LOWER('&nume'), LOWER(ang.numa)) <> 0
        AND INSTR(LOWER('&nume'), LOWER(ang.prenume)) <> 0;
```

```
ACCEPT nume PROMPT 'Introduceti numele si prenumele angajatului: '
SELECT ang.nume,
       ang.prenume,
       CASE
         WHEN o.judet IS NOT NULL THEN o.judet
         ELSE o.denumire
       END LUCREAZA_IN,
       (
         SELECT COUNT(*)
         FROM bonuri
         WHERE cod_angajat = ang.cod_angajat
       ) VANZARI
FROM angajati ang
JOIN librarii lib ON (ang.cod_librarie = lib.cod_librarie)
JOIN locatii loc ON (lib.cod_locatie = loc.cod_locatie)
JOIN orase o ON (loc.cod_oras = o.cod_oras)
WHERE INSTR(LOWER('&nume'), LOWER(ang.nume)) <> 0
      AND INSTR(LOWER('&nume'), LOWER(ang.prenume)) <> 0;
```

Enter Value

Introduceti numele si prenumele angajatului:

MARIA POPESCU

OK Cancel

```
ACCEPT nume PROMPT 'Introduceti numele si prenumele angajatului: '
SELECT ang.nume,
       ang.prenume,
       CASE
         WHEN o.judet IS NOT NULL THEN o.judet
         ELSE o.denumire
       END LUCREAZA_IN,
       (
         SELECT COUNT(*)
         FROM bonuri
         WHERE cod_angajat = ang.cod_angajat
       ) VANZARI
FROM angajati ang
JOIN librarii lib ON (ang.cod_librarie = lib.cod_librarie)
JOIN locatii loc ON (lib.cod_locatie = loc.cod_locatie)
JOIN orase o ON (loc.cod_oras = o.cod_oras)
WHERE INSTR(LOWER('&nume'), LOWER(ang.nume)) <> 0
      AND INSTR(LOWER('&nume'), LOWER(ang.prenume)) <> 0;
```

Enter Value

Introduceti numele si prenumele angajatului:

popescu maria

OK Cancel

	NUME	PRENUME	LUCREAZA_IN	VANZARI
1	Popescu Maria	Bucuresti		4

#### Interogare 4 (NVL, LEFT JOIN, COUNT, subinterogare în clauza FROM)

Câte cărți sunt în librării împărțite în funcție de scriitor.

```
SELECT nume, prenume, NVL(aparitii, 0) CARTI
```

```

FROM scriitori LEFT JOIN (
    SELECT cod_scriitor, COUNT(DISTINCT(cod_carte)) aparitii
    FROM are
    GROUP BY cod_scriitor
) USING (cod_scriitor);

```

	NUME	PRENUME	CARTI
1	Oke	Janette	12
2	Witemeyer	Karen	2
3	Dickens	Charles	1
4	Bunn	T. Davis	8
5	Rowling	J.K.	7
6	Drumes	Mihail	1
7	Dumas	Alexandre	2
8	Tolstoi	Lev	0

#### Interogare 5 (operatorul DIVISION: 2 operatori NOT EXISTS)

Să se găsească toți scriitorii (prenume + nume concatenate) care au scris cărți în toate categoriile de cărți disponibile.

```

SELECT s.prenume || ' ' || s.numa AS NUME_AUTOR
FROM scriitori s
WHERE NOT EXISTS (
    SELECT *
    FROM categorii categ
    WHERE NOT EXISTS (
        SELECT *
        FROM are ar
        WHERE ar.cod_categorie = categ.cod_categorie
        AND ar.cod_scriitor = s.cod_scriitor
    )
)

```

);

NUME_AUTOR
1 Janette Oke

**Interogare 6** (*DECODE, NVL, TO\_CHAR, INSTR, LOWER, LEFT JOIN, RIGHT JOIN, subinterogare în clauza FROM*)

Dându-se titlul unei cărți, să se determine informații despre carte (numele scriitorului și limba în care se poate găsi) și detalii despre câte exemplare se găsesc pe stoc și în ce librării. Să se trateze cazurile în care nu se cunoaște autorul, nu se găsesc informații despre limbile în care este tradusă cartea, precum și dacă se găsește sau nu pe stoc.

```
ACCEPT nume PROMPT 'Nume carte: '

SELECT c.nume AS NUME_CARTE,
       DECODE(numa_scriitor, NULL, 'Necunoscut', numa_scriitor) AS
NUMA_AUTOR,
       DECODE(l.limba, NULL, 'Necunoscuta', l.limba) AS LIMBA_TRADUSA,
       DECODE(loc.strada, NULL, 'Nu se mai gaseste pe stoc',
loc.strada || ' ' || TO_CHAR(loc.numar)) AS ADRESA,
       NVL(st.nr_exemplare, 0) EXEMPLARE
FROM (
       SELECT nume, cod_carte
       FROM carti
       WHERE INSTR(LOWER('&nume'), LOWER(numa)) <> 0
) c
LEFT JOIN (
       SELECT DISTINCT(cod_carte) cod_carte, numa numa_scriitor
       FROM scriitori RIGHT JOIN are USING (cod_scriitor)
) scr ON scr.cod_carte = c.cod_carte
```

```

LEFT JOIN traduceri t ON c.cod_carte = t.cod_carte

LEFT JOIN limbi l ON t.cod_limba = l.cod_limba

LEFT JOIN stocuri st ON c.cod_carte = st.cod_carte

LEFT JOIN librarii lib ON st.cod_librarie = lib.cod_librarie

LEFT JOIN locatii loc ON lib.cod_locatie = loc.cod_locatie AND
lib.cod_oras = loc.cod_oras;

```

```

ACCEPT nume PROMPT 'Nume carte: '
SELECT c.num AS NUME_CARTE,
       DECODE(num_scriitor, NULL, 'Necunoscut', num_scriitor) AS NUME_AUTOR,
       DECODE(l.limba, NULL, 'Necunoscuta', l.limba) AS LIMBA_TRADUSA,
       DECODE(loc.strada, NULL, 'Nu se mai gaseste pe stoc', loc.strada || ' ' || TO_CHAR(loc.numar)) AS ADRESA,
       NVL(st.nr_exemplare, 0) EXEMPLARE
FROM (
  SELECT nume, cod_carte
  FROM carti
  WHERE INSTR(LOWER('&nume'), LOWER(nume)) <> 0
) c
LEFT JOIN (
  SELECT DISTINCT(cod_carte) cod_carte, nume nume
  FROM scriitori RIGHT JOIN are USING (cod_scriitor)
) scr ON scr.cod_carte = c.cod_carte
LEFT JOIN traduceri t ON c.cod_carte = t.cod_carte
LEFT JOIN limbi l ON t.cod_limba = l.cod_limba
LEFT JOIN stocuri st ON c.cod_carte = st.cod_carte
LEFT JOIN librarii lib ON st.cod_librarie = lib.cod_librarie
LEFT JOIN locatii loc ON lib.cod_locatie = loc.cod_locatie AND lib.cod_oras = loc.cod_oras;

```

Enter Value

Nume carte:

OK Cancel

NUME_CARTE	NUME_AUTOR	LIMBA_TRADUSA	ADRESA	EXEMPLARE
1 Invitatie la vals	Drumes	romana	Strada Doamnei 20	2
2 Invitatie la vals	Drumes	romana	Strada Vasile Lupu 148	2

```
ACCEPT nume PROMPT 'Nume carte: '
SELECT c.num AS NUME_CARTE,
       DECODE(num_scriitor, NULL, 'Necunoscut', num_scriitor) AS NUME_AUTOR,
       DECODE(l.limba, NULL, 'Necunoscuta', l.limba) AS LIMBA_TRADUSA,
       DECODE(loc.strada, NULL, 'Nu se mai gaseste pe stoc', loc.strada || ' ' || TO_CHAR(loc.numar)) AS ADRESA,
       NVL(st.nr_exemplare, 0) EXEMPLARE
FROM (
  SELECT nume, cod_carte
  FROM carti
  WHERE INSTR(LOWER('&nume'), LOWER(nume)) <> 0
) c
LEFT JOIN (
  SELECT DISTINCT(cod_carte) cod_carte, nume nume_
  FROM scriitori RIGHT JOIN are USING (cod_scriitor)
) scr ON scr.cod_carte = c.cod_carte
LEFT JOIN traduceri t ON c.cod_carte = t.cod_carte
LEFT JOIN limbi l ON t.cod_limba = l.cod_limba
LEFT JOIN stocuri st ON c.cod_carte = st.cod_carte
LEFT JOIN librarii lib ON st.cod_librarie = lib.cod_librarie
LEFT JOIN locatii loc ON lib.cod_locatie = loc.cod_locatie AND lib.cod_oras = loc.cod_oras;
```

NUME_CARTE	NUME_AUTOR	LIMBA_TRADUSA	ADRESA	EXEMPLARE
1 O mie si una de nopti	Necunoscut	Necunoscuta	Nu se mai gaseste pe stoc	0

### Interogare 7 (COUNT, TO\_CHAR, TO\_DATE, ADD\_MONTH, subinterogare în clauza FROM)

Comparați câte cărți au fost vândute până în prezent față de câte cărți au fost vândute acum un an și calculați cu câte s-au vândut mai multe în prezent față de atunci.

```
SELECT PRESENT, NUMAR_IN_PREZENT, ACUM_12_LUNI, NUMAR_ACUM_12_LUNI,
       NUMAR_IN_PREZENT - NUMAR_ACUM_12_LUNI CRESTERE
FROM (
  SELECT COUNT(*) NUMAR_IN_PREZENT, SYSDATE PREZENT
  FROM bonuri b
), (
  SELECT COUNT(*) NUMAR_ACUM_12_LUNI, TO_CHAR(ADD_MONTHS(SYSDATE, -12), 'DD-MON-YYYY') ACUM_12_LUNI
  FROM bonuri
  WHERE TO_DATE(data_bon, 'DD-MON-YYYY') <=
        TO_DATE(ADD_MONTHS(SYSDATE, -12), 'DD-MON-YYYY')
);
```

	PREZENT	NUMAR_IN_PREZENT	ACUM_12_LUNI	NUMAR_ACUM_12_LUNI	CRESTERE
1	04-JAN-24		12 04-JAN-2023	8	4

### Interogare 8 (MONTHS\_BETWEEN, COUNT, LOWER, GROUP BY, subinterogare în clauza WHERE)

Afișează acum câte luni a fost vândută cartea „Moștenirea” și de câte ori a fost vândută în acele luni.

```
SELECT  ROUND(MONTHS_BETWEEN(SYSDATE, data_bon)) AS LUNI_TRECUTE,
COUNT(*) AS EXEMPLARE_VANDUTE
FROM bonuri
WHERE cod_carte = (
SELECT cod_carte
FROM carti
WHERE LOWER(ume) = 'mostenirea'
)
GROUP BY ROUND(MONTHS_BETWEEN(SYSDATE, data_bon));
```

	LUNI_TRECUTE	EXEMPLARE_VANDUTE
1	11	1
2	10	1
3	0	2

### Interogare 9 (DECODE, NVL, MAX, AVG, FULL JOIN, GROUP BY, ORDER BY, subinterogare în clauza FROM)

Să se determine numărul maxim de pagini și numărul mediu de pagini a cărților în funcție de categorii. Cărțile care nu se încadrează în nicio categorie, vor fi puse într-o categorie nouă numită „fără categorie”.

```
SELECT DECODE(c.gen, NULL, 'fara categorie', c.gen) AS CATEGORY,
NVL(MAX(ca.numar_pagini), 0) AS MAX_PAGINI,
```



```

        NVL(ROUND(AVG(ca.numar_pagini), 2), 0) AS AVG_PAGINI
FROM (
    SELECT a.cod_categorie, ca.numar_pagini
    FROM carti ca FULL JOIN are a ON ca.cod_carte = a.cod_carte
) ca FULL JOIN categorii c ON c.cod_categorie = ca.cod_categorie
GROUP BY c.gen
ORDER BY c.gen;

```

❖ CATEGORY	❖ MAX_PAGINI	❖ AVG_PAGINI
1 aventura	622	408
2 crestin	432	305.91
3 fictiune	990	432.9
4 istoric	622	288
5 non-fictiune	336	243.6
6 romantic	336	261.14
7 fara categorie	4800	4800

**Interogare 10** (CASE, SUM, GROUP BY, INNER JOIN, ADD\_MONTHS, TO\_DATE, operator pe mulțime INTERSECT, subinterogare în clauza FROM, subinterogare în clauza WHERE)

Să se determine cărțile care au fost vândute în ultimele 12 luni și care încă se mai găsesc în stocul cel puțin al unei librării, menționându-se dacă prețul vânzării a fost diferit cândva de cel curent.

```

SELECT cod_carte, nume,
    CASE
        WHEN SUM(este_diferit) = 0
        THEN 'nu'
        ELSE 'da'
    END CUMPARARE_CU_PRET_DIFERIT
FROM (

```

```

SELECT cod_carte, nume,
       CASE
         WHEN c.pret = bon.pret
         THEN 0
         ELSE 1
       END este_diferit
FROM carti c JOIN (
                SELECT cod_carte, pret
                FROM bonuri
                ) bon USING (cod_carte)
WHERE cod_carte in (
        SELECT DISTINCT(cod_carte)
        FROM bonuri
        WHERE ADD_MONTHS(TO_DATE(data_bon), 12) >= SYSDATE

        INTERSECT

        SELECT DISTINCT(cod_carte)
        FROM stocuri
        )
)
GROUP BY cod_carte, nume;

```

	⚡ COD_CARTE	⚡ NUME	⚡ CUMPARARE_CU_PRET_DIFERIT
1	100	Invitatie la vals	da
2	107	Limanul mult dorit	da
3	104	Mostenirea	da

Se afișează și datele returnate din subintegrorarea din clauza FROM pentru o mai ușoară înțelegere a întregorii:

	COD_CARTE	NUME	ESTE_DIFERIT
1	100	Invitatie la vals	1
2	104	Mostenirea	1
3	104	Mostenirea	0
4	107	Limanul mult dorit	0
5	107	Limanul mult dorit	1
6	100	Invitatie la vals	0
7	104	Mostenirea	0
8	104	Mostenirea	0

### Interogare 11 (operator pe mulțime MINUS, subinterogare în clauza WHERE)

Să se determine ce cărți care au fost publicate după anul 1990 sunt disponibile din librăria cu codul 70 și nu s-au vândut niciodată.

```

SELECT c.nume AS TITLU_CARTE
FROM carti c
WHERE c.cod_carte IN (
    SELECT s.cod_carte
    FROM stocuri s
    WHERE s.cod_librarie = 70
) AND an_aparitie > 1990

MINUS

SELECT nume AS TITLU_CARTE
FROM carti c
WHERE c.cod_carte IN (
    SELECT DISTINCT(b.cod_carte)
    FROM bonuri b

```

);

TITLU_CARTE
1 Harry Potter si ordinul phoenix
2 Harry Potter si piatra filosofala
3 Harry Potter si pocalul de foc
4 Harry Potter si printul semisange
5 Harry Potter si prizonierul din Azkaban
6 Harry Potter si talismanele mortii

**Interogare 12** (*SUBSTR, CASE, NULLIF, LEFT JOIN, SUM, ADD\_MONTHS, GROUP BY, UPPER, subinterogare în clauza FROM*)

Se dorește să se ofere o mărire pentru angajații care au facut vânzări mari în ultimul an. Vânzări mari înseamnă că suma este peste 100 lei. Se va afișa o listă cu toți vânzătorii a căror nume este codificat. Codificarea se realizează astfel: concatenarea primei litere din nume, primei litere din prenume și a codului unic. În plus se afișează pentru fiecare angajat valoarea salariului actual, valoarea vânzărilor făcute și, dacă vânzătorul merită mărirea, se afișează și valoarea nouă a salariului după mărire. Altfel se afișeaza NULL.

```
SELECT SUBSTR(nume, 1, 1) || SUBSTR(prenume, 1, 1) || cod_angajat cod,
       salariu SALARIU,
       vanzari VALOARE_VANZARI,
       CASE
           WHEN NULLIF(trunc(vanzari/100), 0) IS NOT NULL
           THEN salariu * 1.1
           ELSE NULL
       END MARIRE
FROM angajati LEFT JOIN (
    SELECT cod_angajat, SUM(pret) vanzari
    FROM bonuri
```

```

WHERE ADD_MONTHS(data_bon, 12) > SYSDATE

GROUP BY cod_angajat

) USING (cod_angajat)

WHERE UPPER(tip_angajat) = 'VANZATOR';

```

❖	COD	❖	SALARIU	❖	VALOARE_VANZARI	❖	MARIRE
1	PM400		1275		71.3		(null)
2	IA408		1300		25.65		(null)
3	PA409		1750		121.3		1925
4	MC410		1000		82.5		(null)
5	PM401		2450		(null)		(null)
6	MC403		1600		(null)		(null)
7	IA402		2725		(null)		(null)

**Interogare 13** (*DECODE, subinterogare în clauza SELECT, COUNT, START WITH, CONNECT, ORDER BY*)

Să se afișeze structura categoriilor într-un mod vizibil (categoria principală, apoi subcategoriile care sunt incluse). Împreună cu această structură, trebuie să se mai afișeze câteva detalii precum codul cărții, codul categoriei părinte (dacă are, altfel un mesaj scurt), nivelul pe care se află acea categorie și numărul de carti din baza de date pentru care s-a specificat ca este de acel gen.

```

SELECT LPAD(' ', 2 * LEVEL - 2) || gen AS NUME,
       cod_categorie,
       DECODE(parinte_cod_categorie, NULL, 'nu are',
parinte_cod_categorie) AS COD_PARINTE,
       LEVEL,
       (
         SELECT COUNT(DISTINCT(cod_carte))
         FROM are
         WHERE are.cod_categorie = categorii.cod_categorie

```

```

        ) AS NUMAR_CARTI

FROM categorii

START WITH parinte_cod_categorie IS NULL

CONNECT BY PRIOR cod_categorie = parinte_cod_categorie

ORDER SIBLINGS BY gen, cod_categorie;

```

	NUME	COD_CATEGORIE	COD_PARINTE	LEVEL	NUMAR_CARTI
1	fictiune	10	nu are	1	15
2	aventura	15	10	2	6
3	crestin	14	10	2	14
4	romantic	12	10	2	7
5	non-fictiune	11	nu are	1	5
6	istoric	13	11	2	6

**Interogare 14** (operator pe mulțime INTERSECT, operator pe mulțime MINUS, operator pe mulțime UNION, LEFT JOIN, UPPER, subinterogare în clauza FROM, ORDER BY)

Managerul vrea să promoveze citirea cărților în alte limbi și a celor scrise de autori străini. De aceea dorește să afle numele cărților accesibile (preț de cel mult 30 lei) în ordinea pretului de vanzare care sunt traduse în orice altă limba decât română și celor scrise de autori străini, specificându-se pentru fiecare din care categorie face parte sau dacă se încadrează în ambele.

```

WITH carti_limba_straina AS (
    SELECT cod_carte, nume, pret
    FROM carti c
    WHERE EXISTS (
        SELECT *
        FROM limbi lim LEFT JOIN traduceri tr USING
(cod_limba)
        WHERE UPPER(lim.limba) <> 'ROMANA'
        AND tr.cod_carte = c.cod_carte
    )
)

```

```

        )

        AND pret <= 30
    ),

    carti_autori_straini AS (

        SELECT cod_carte, nume, pret

        FROM carti c

        WHERE NOT EXISTS (

            SELECT *

            FROM scriitori LEFT JOIN are USING (cod_scriitor)

            WHERE UPPER(nationalitate) = 'ROMANA'

            AND are.cod_carte = c.cod_carte

        )

        AND pret <= 30
    ),

    carti_autor_si_limba_straina AS (

        SELECT cod_carte, nume, pret

        FROM carti_limba_straina

        INTERSECT

        SELECT cod_carte, nume, pret

        FROM carti_autori_straini

    )

SELECT cod_carte, nume, pret, categorie

FROM (

    SELECT cod_carte, nume, pret, 'limba straina' AS categorie

    FROM (

        SELECT *
    
```

```

        FROM carti_limba_straina

        MINUS

        SELECT *

        FROM carti_autor_si_limba_straina
    )

    UNION

    SELECT cod_carte, nume, pret, 'autor strain' AS categorie

    FROM (

        SELECT *

        FROM carti_autori_straini

        MINUS

        SELECT *

        FROM carti_autor_si_limba_straina

    )

    UNION

    SELECT cod_carte, nume, pret, 'limba straina + autor strain' AS
categorie

    FROM carti_autor_si_limba_straina

)

ORDER BY pret;

```



	COD_CARTE	NUME	PRET	CATEGORIE
1	123	Cei trei muschetari	21	limba straina + autor strain
2	109	A fost odata intr-o vara	21.85	autor strain
3	112	Iarna nu tine o vesnicie	21.85	autor strain
4	110	Promisiunea unei noi primaveri	21.85	autor strain
5	111	Vanturi tomnatice	21.85	autor strain
6	108	Un far calauzitor	27	limba straina + autor strain
7	106	Locul de intalnire	27	limba straina + autor strain
8	104	Mostenirea	27	limba straina + autor strain
9	107	Limanul mult dorit	27	limba straina + autor strain
10	102	Mireasa pe masura	28.5	autor strain
11	105	Binecuvantatul tarm	28.5	limba straina + autor strain
12	103	Marile sperante	30	autor strain

**Interogare 15** (*COUNT, INNER JOIN, GROUP BY, HAVING, MAX, subinterogare în clauza FROM, subinterogare în clauza HAVING, subinterogare în clauza WHERE*)

Precizați codul, numele și numărul de cărți vândute de angajații care au vândut un număr maxim de cărți, selectându-i doar pe aceia care lucrează într-o librărie care are număr maxim de angajați. Să se precizeze și unde se găsesc aceste librării (strada).

	COD_ANGAJAT	NUME_ANGAJAT	NUMAR_VANZARI	LOCATIE_LIBRARIE
1	410	Mihaita	5	Strada Fierului

```

SELECT bon.cod_angajat,
       nume AS NUME_ANGAJAT,
       COUNT(cod_carte) AS NUMAR_VANZARI,
       strada AS LOCATIE_LIBRARIE
FROM bonuri bon JOIN angajati ang ON bon.cod_angajat = ang.cod_angajat
      JOIN (
              SELECT cod_librarie, strada
              FROM locatii join librarii USING (cod_locatie)
            ) loc ON loc.cod_librarie = ang.cod_librarie
HAVING COUNT(bon.cod_angajat) = (
              SELECT MAX(COUNT(cod_angajat))

```

```

        FROM bonuri
        GROUP BY cod_angajat
    )      --numarul maxim de carti vandute de un
angajat
    AND bon.cod_angajat in (
        SELECT cod_angajat
        FROM angajati
        WHERE cod_librarie in (
            SELECT cod_librarie
            FROM angajati
            GROUP BY cod_librarie
            HAVING COUNT(cod_librarie) = (
                SELECT
MAX(COUNT(cod_librarie)) max_libr
                FROM angajati
                GROUP BY cod_librarie
            )
        )
    )
    )      --angajatii care lucreaza intr-o librerie cu numar
maxim de angajati
GROUP BY bon.cod_angajat, nume, strada;

```

## Creare tabel pentru mesaje

### Structură dorită:

Field	Data type	Comments
Message_id	NUMBER	Cheie primară
Message	VARCHAR2(255)	
Message_type	VARCHAR2(1)	Valid values: E - Error, W - Warning, I - Information
Created_by	VARCHAR2(40)	NOT NULL
Created_at	DATE	NOT NULL

```
CREATE TABLE MESAJE (  
    Message_id NUMBER CONSTRAINT PKEY_MESAJE PRIMARY KEY,  
    Message VARCHAR2(255),  
    Message_type VARCHAR2(1) CONSTRAINT TYPE_CONSTRAINT CHECK  
(Message_type IN ('E', 'W', 'I')),  
    Created_by VARCHAR2(40) NOT NULL,  
    Created_at DATE NOT NULL  
);  
  
COMMENT ON COLUMN MESAJE.Message_id IS 'Cheie primară';  
  
COMMENT ON COLUMN MESAJE.Message_type IS 'Valid values: E - Error, W -  
Warning, I - Information';  
  
COMMENT ON COLUMN MESAJE.Created_by IS 'NOT NULL';  
  
COMMENT ON COLUMN MESAJE.Created_at IS 'NOT NULL';
```

### Structura creată:

❖ COLUMN_NAME	❖ DATA_TYPE	❖ NULLABLE	DATA_DEFAULT	❖ COLUMN_ID	❖ COMMENTS
1 MESSAGE ID	NUMBER	No	(null)	1	Cheie primară
2 MESSAGE	VARCHAR2(255 BYTE)	Yes	(null)	2	(null)
3 MESSAGE TYPE	VARCHAR2(1 BYTE)	Yes	(null)	3	Valid values: E - Error, W - Warning, I - Information
4 CREATED BY	VARCHAR2(40 BYTE)	No	(null)	4	NOT NULL
5 CREATED AT	DATE	No	(null)	5	NOT NULL

## Probleme în PL/SQL

1. Creați un subprogram stocat independent (inclusiv apelare) care să utilizeze toate cele 3 tipuri de colecții învățate.

### Cerintă:

Se dorește să se țină evidența celor mai vândute categorii din librării. Se ține cont doar de cărțile vândute în anul curent, afișându-se cât de căutate sunt acele categorii și, pentru fiecare categorie, se determină lista de cărți vândute (titlul cărților) de acel gen și de câte ori au fost vândute.

### Rezultat apelare:

```
CREATE OR REPLACE PROCEDURE categoriile_cele_mai_vandute
IS
    -- tip de date record pentru a retine detalii despre o carte
    vanduta

    TYPE info_carte_record IS RECORD (
        nume CARTI.nume%TYPE,
        cod are.cod_carte%TYPE,
        copies_sold NUMBER
    );

    -- tablou imbricat pentru a retine informatii despre toate cartile
    vandute

    TYPE info_carte_tip IS TABLE OF info_carte_record;

    v_carti_vandute info_carte_tip;

    -- tablou imbricat in care se va retine numarul de vanzari a
    fiecărei categorii vandute
```

```

TYPE count_categorie_tip IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
v_count_categorii count_categorie_tip;

-- Vector in care retin coduri cartilor care se incadreaza intr-o
categories

TYPE coduri_carti_tip IS VARRAY(100) OF carti.cod_carte%TYPE;
v_coduri_carti coduri_carti_tip := coduri_carti_tip();

v_maxim NUMBER := 0;
v_categorie categorii.gen%TYPE;
BEGIN

-- Pastrarea cartilor vandute si numarului de vanzari
SELECT c.nume, b.cod_carte, COUNT(*)
BULK COLLECT INTO v_carti_vandute
FROM carti c RIGHT JOIN bonuri b ON c.cod_carte = b.cod_carte
WHERE EXTRACT(YEAR FROM data_bon) = EXTRACT(YEAR FROM SYSDATE)
GROUP BY c.nume, b.cod_carte;

-- determinarea categoriilor care au fost vandute si a numarului
de vanzari
FOR index_carte IN v_carti_vandute.FIRST .. v_carti_vandute.LAST
LOOP
    FOR categorie IN (
        SELECT DISTINCT cod_categorie cod
        FROM are
        WHERE v_carti_vandute(index_carte).cod =
are.cod_carte

```

```

        )

        LOOP

            IF v_count_categorii.EXISTS(categorie.cod)

                THEN

                    v_count_categorii(categorie.cod)                :=
v_count_categorii(categorie.cod)                                +
v_carti_vandute(index_carte).copies_sold;

                ELSE

                    v_count_categorii(categorie.cod)                :=
v_carti_vandute(index_carte).copies_sold;

                END IF;

            IF v_count_categorii(categorie.cod) > v_maxim

                THEN

                    v_maxim := v_count_categorii(categorie.cod);

                END IF;

            END LOOP;

        END LOOP;

        IF v_maxim <> 0

            THEN

                DBMS_OUTPUT.PUT_LINE('Numarul maxim de carti dintr-un anumit
tip de categorie vandute: ' || v_maxim);

                DBMS_OUTPUT.PUT_LINE('');

                -- Match-uirea categorie-carte si afisarea informatiilor
dorite

                FOR index_cod_categorie IN v_count_categorii.FIRST ..
v_count_categorii.LAST LOOP

```

```

        IF v_count_categorii.EXISTS(index_cod_categorie) THEN
            IF v_count_categorii(index_cod_categorie) = v_maxim
THEN
                SELECT gen into v_categorie
                FROM categorii
                WHERE cod_categorie = index_cod_categorie;

                DBMS_OUTPUT.PUT_LINE('>      Categorie:      ' ||
v_categorie);

                SELECT DISTINCT cod_carte
                BULK COLLECT INTO v_coduri_carti
                FROM are
                WHERE index_cod_categorie = are.cod_categorie;

                FOR index_cod IN v_coduri_carti.FIRST ..
v_coduri_carti.LAST LOOP
                    FOR index_carte IN v_carti_vandute.FIRST ..
v_carti_vandute.LAST LOOP
                        IF v_carti_vandute(index_carte).cod =
v_coduri_carti(index_cod) THEN
                            DBMS_OUTPUT.PUT_LINE(' - ' ||

v_carti_vandute(index_carte).nume ||

                                ': vanduta de ' ||

v_carti_vandute(index_carte).copies_sold || ' ori');
                        END IF;
                    END LOOP;
                END LOOP;

```

```

                                END LOOP;

                                END IF;

                                END IF;

                                END LOOP;

ELSE

                                DBMS_OUTPUT.PUT_LINE('Nu au fost cumparate carti in anul
curent.');
```

END categoriile\_cele\_mai\_vandute;

/

execute categoriile\_cele\_mai\_vandute;

### Rezultat apelare:

```

Task completed in 0.037 seconds

Numarul maxim de carti dintr-un anumit tip de categorie vandute: 5

> Categorie: fictiune
- Mireasa pe masura: vanduta de 1 ori
- Limanul mult dorit: vanduta de 2 ori
- Mostenirea: vanduta de 2 ori
> Categorie: crestin
- Mireasa pe masura: vanduta de 1 ori
- Limanul mult dorit: vanduta de 2 ori
- Mostenirea: vanduta de 2 ori

PL/SQL procedure successfully completed.
```

2. Creați un subprogram stocat independent (inclusiv apelare) care să utilizeze 2 tipuri de cursoare învățate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor.

### Cerintă:



Pentru fiecare scriitor din baza de date determinati cat costa fiecare serie pe care a scris-o in cazul in care cumparatorul ar dori sa cumpere toate volumele. Se doreste sa se afiseze pentru fiecare serie pretul acesteia, precum si un mesaj sugestiv in cazul in care scriitorul nu a scris nicio serie de carti sau nu are carti introduse.

```
CREATE OR REPLACE PROCEDURE pret_serie
IS
    CURSOR scriitori_cu_carti IS
        SELECT nume, prenume, cod_scriitor, NVL(aparitii, 0)
        FROM scriitori LEFT JOIN (
            SELECT                                cod_scriitor,
COUNT(DISTINCT(cod_carte)) aparitii
            FROM are
            GROUP BY cod_scriitor
        ) USING (cod_scriitor);

    CURSOR suma_serie (p_cod_scriitor scriitori.cod_scriitor%TYPE) IS
        SELECT sum(pret) suma, serii.nume
        FROM carti LEFT JOIN serii USING (cod_serie)
        WHERE cod_serie IS NOT NULL
            AND cod_carte IN (
                SELECT DISTINCT cod_carte
                FROM are
                WHERE cod_scriitor = p_cod_scriitor
            )
        GROUP BY serii.nume;
```

```

v_serie serii.nume%TYPE;

v_suma NUMBER;

v_cod_scriitor scriitori.cod_scriitor%TYPE;

v_nume scriitori.nume%TYPE;

v_prenume scriitori.prenume%TYPE;

v_aparitii NUMBER;

BEGIN

    OPEN scriitori_cu_carti;

    LOOP

        FETCH      scriitori_cu_carti      INTO      v_nume,      v_prenume,
v_cod_scriitor, v_aparitii;

        EXIT WHEN scriitori_cu_carti%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE (CHR(10) || 'Scriitorul      ' || v_nume || '
' || v_prenume);

        IF v_aparitii <> 0

            THEN

                OPEN suma_serie(v_cod_scriitor);

                LOOP

                    FETCH suma_serie INTO v_suma, v_serie;

                    EXIT WHEN suma_serie%NOTFOUND;

                    DBMS_OUTPUT.PUT_LINE (CHR(9)      || 'Seria      ' || v_serie || '
costa ' || v_suma || ' lei. ');

                END LOOP;

```

```

        IF suma_serie%ROWCOUNT = 0 THEN

            DBMS_OUTPUT.PUT_LINE(CHR(9) || 'Nu exista in librerie
nicio serie scrisa de acest scriitor.');
```

END IF;

```

        CLOSE suma_serie;

    ELSE

        DBMS_OUTPUT.PUT_LINE(CHR(9) || 'Nu exista in librerie
nicio carte scrisa de acest scriitor.');
```

END IF;

END LOOP;

```

    IF scriitori_cu_carti%ROWCOUNT = 0 THEN

        DBMS_OUTPUT.PUT_LINE(CHR(10) || 'Nu exista niciun scriitor in
baza de date.');
```

END IF;

```

    CLOSE scriitori_cu_carti;

    EXCEPTION

        WHEN OTHERS THEN

            INSERT INTO mesaje

                VALUES      (SEQ_ERROR.NEXTVAL,      'A aparut o eroare
necunoscuta', 'E', SYS.LOGIN_USER, SYSDATE);

            RAISE_APPLICATION_ERROR(-20004,      'A aparut o eroare
necunoscuta');
```

END pret\_serie;

/

```
BEGIN

    pret_serie;

EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('Eroarea are codul = '||SQLCODE || ' '
si mesajul = ' || SQLERRM);

END;

/
```

**Rezultat apelare:**

```

Scriitorul Oke Janette
  Seria "Anotimpurile inimii" costa 87.4 lei.
  Seria "Cantecul Acadiei" costa 136.5 lei.
  Seria "Faptele credintei" costa 105 lei.

Scriitorul Witemeyer Karen
  Nu exista in librerie nicio serie scrisa de acest scriitor.

Scriitorul Dickens Charles
  Nu exista in librerie nicio serie scrisa de acest scriitor.

Scriitorul Bunn T. Davis
  Seria "Cantecul Acadiei" costa 136.5 lei.
  Seria "Faptele credintei" costa 105 lei.

Scriitorul Rowling J.K.
  Seria "Harry Potter" costa 390.88 lei.

Scriitorul Drumes Mihail
  Nu exista in librerie nicio serie scrisa de acest scriitor.

Scriitorul Dumas Alexandre
  Seria "Cei trei muschetari" costa 54 lei.

Scriitorul Tolstoi Lev
  Nu exista in librerie nicio carte scrisa de acest scriitor.

PL/SQL procedure successfully completed.

```

3. Creați un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite; tratarea tuturor excepțiilor care pot apărea (definiți minim 2 excepții proprii); apelarea subprogramului astfel încât să fie evidențiate toate cazurile tratate.

#### Cerință:

Într-o librărie de la o locație dată prin nume vine un client care dorește să cumpere o carte anume, dată prin titlu. Să se precizeze fiecare caz posibil, adică cartea să existe în librăria respectivă, să se creeze un nou bon (nu contează ce vânzător o vinde) și să se modifice numărul de exemplare disponibile pe stoc, dar și cazul în care cartea nu exista pe stoc sau informațiile transmise nu corespund datelor din baza de date.

```
CREATE OR REPLACE FUNCTION client_cumpara_carte(p_carte_ceruta
```

```

carti.nume%TYPE DEFAULT 'no_name',
                                                                    p_librarie_solicitanta
locatii.strada%TYPE DEFAULT 'no_adress')

RETURN carti.pret%TYPE IS

v_evidenta NUMBER;
v_cod_carte stocuri.cod_carte%TYPE;
v_cod_librarie_care_vinde stocuri.cod_librarie%TYPE;
v_pret_cumparare carti.pret%TYPE;
v_bon bonuri%ROWTYPE;
NU_EXISTA_CARTE EXCEPTION;
NU_EXISTA_LIBRARIE EXCEPTION;
NU_EXISTA_CARTEA_PE_STOC EXCEPTION;
v_mesaj_eroare VARCHAR2(100);

BEGIN

SELECT COUNT(*)
INTO v_evidenta
FROM carti
WHERE UPPER(nume) = UPPER(p_carte_ceruta);

IF v_evidenta = 0 THEN
    RAISE NU_EXISTA_CARTE;
END IF;

SELECT COUNT(*)
INTO v_evidenta

```

```

FROM locatii

WHERE UPPER(strada) = UPPER(p_librarie_solicitanta);

IF v_evidenta = 0 THEN
    RAISE NU_EXISTA_LIBRARIE;
END IF;

UPDATE stocuri

SET nr_exemplare = nr_exemplare - 1

WHERE (cod_carte, cod_librarie) =

    (
        -- daca cartea se gaseste in librarie, select-ul va
        intoarce o linie

        SELECT cod_carte, st.cod_librarie

        FROM    locatii    loc    JOIN    librarii    lib    ON
        (loc.cod_locatie=lib.cod_locatie)

        JOIN    (-- selectez toate codurile
        librariile in care se gaseste cartea dorita

        SELECT cod_carte, cod_librarie

        FROM carti JOIN stocuri USING

        (cod_carte)

        WHERE    UPPER(ume)    =

        UPPER(p_carte_ceruta)

        )    st    ON(st.cod_librarie    =

        lib.cod_librarie)

        WHERE UPPER(strada) = UPPER (p_librarie_solicitanta)

    )

RETURNING    cod_carte,    cod_librarie    INTO    v_cod_carte,
v_cod_librarie_care_vinde;

```

```

-- UPDATE nu intoarce NO_DATA_FOUND

IF SQL%rowcount = 0 THEN

    RAISE NU_EXISTA_CARTEA_PE_STOC;

END IF;


-- carte este vanduta, se selecteaza pretul pentru a putea crea
bonul

SELECT pret

INTO v_pret_cumparare

FROM carti

WHERE cod_carte = v_cod_carte;


SELECT SEQ_CUMP.NEXTVAL, v_pret_cumparare,

        SYSDATE, v_cod_carte, (

                SELECT cod_angajat

                        FROM ( -- se alege in mod random un
vanzator care lucreaza la acea librerie

                                -- pentru ca fiecare sa aiba
sansa egala de a vinde

                                        SELECT cod_angajat

                                                FROM angajati

                                                        WHERE          cod_librarie          =
v_cod_librarie_care_vinde AND LOWER(tip_angajat) = 'vanzator'

                                                                ORDER BY DBMS_RANDOM.RANDOM

                                )

                        WHERE ROWNUM = 1

                )

INTO v_bon

```



```

FROM dual;

INSERT INTO bonuri

VALUES v_bon;

-- cartea poate avea reducere, deci preluam din nou bonul pentru
a-l putea transmite clientului

SELECT pret

INTO v_pret_cumparare

FROM bonuri

WHERE cod_bon = v_bon.cod_bon;

RETURN v_pret_cumparare;

EXCEPTION

    WHEN NU_EXISTA_CARTE THEN

        v_mesaj_eroare := 'Nu exista cartea ceruta ' ||
p_carte_ceruta || ' in baza noastra de date.';

        INSERT INTO mesaje

            VALUES (SEQ_ERROR.NEXTVAL, v_mesaj_eroare, 'E',
SYS.LOGIN_USER, SYSDATE);

        RAISE_APPLICATION_ERROR(-20001, v_mesaj_eroare);

    WHEN NU_EXISTA_LIBRARIE THEN

        v_mesaj_eroare := 'Nu exista nicio librerie la adresa data
- ' || p_librarie_solicitantă;

        INSERT INTO mesaje

            VALUES (SEQ_ERROR.NEXTVAL, v_mesaj_eroare, 'E',

```

```

SYS.LOGIN_USER, SYSDATE);

        RAISE_APPLICATION_ERROR(-20002, v_mesaj_eroare);

        WHEN NU_EXISTA_CARTEA_PE_STOC THEN

                v_mesaj_eroare := 'Ne pare rau, nu avem cartea ' ||
p_carte_ceruta ||

                ' pe stoc la locatia ' ||
p_librarie_solicitanta;

                INSERT INTO mesaje

                        VALUES      (SEQ_ERROR.NEXTVAL,      v_mesaj_eroare,      'E',
SYS.LOGIN_USER, SYSDATE);

                RAISE_APPLICATION_ERROR(-20003, v_mesaj_eroare);

        WHEN OTHERS THEN

                v_mesaj_eroare := SQLERRM;

                INSERT INTO mesaje

                        VALUES      (SEQ_ERROR.NEXTVAL,      v_mesaj_eroare,      'E',
SYS.LOGIN_USER, SYSDATE);

                RAISE_APPLICATION_ERROR(-20004, v_mesaj_eroare);

END client_cumpara_carte;

/

DECLARE

--      Clientul a platit 30.6 lei pe cartea Invitatie la vals.

      carte_carti.nume%TYPE := 'Invitatie la vals';

      strada_locatii.strada%TYPE := 'Strada Doamnei';

--

```

```

--      Eroarea are codul = -20003 si mesajul = ORA-20003: Ne pare rau,
nu avem cartea Invitatie la vals pe stoc la locatia Strada Fierului

--      carte carti.nume%TYPE := 'Invitatie la vals';

--      strada locatii.strada%TYPE := 'Strada Fierului';


--      Eroarea are codul = -20001 si mesajul = ORA-20001: Nu exista
cartea ceruta Carte necunoscuta in baza noastra de date.

--      carte carti.nume%TYPE := 'Carte necunoscuta';

--      strada locatii.strada%TYPE := 'Strada Doamnei';


--      Eroarea are codul = -20002 si mesajul = ORA-20002: Nu exista
nicio librerie la adresa data - Strada necunoscuta

--      carte carti.nume%TYPE := 'Invitatie la vals';

--      strada locatii.strada%TYPE := 'Strada necunoscuta';

BEGIN

      DBMS_OUTPUT.PUT_LINE('Clientul          a          platit
'||client_cumpara_carte(carte, strada)|| lei pe cartea '||carte||
|.');

      EXCEPTION

      WHEN OTHERS THEN

            DBMS_OUTPUT.PUT_LINE('Eroarea are codul = '||SQLCODE || '
si mesajul = ' || SQLERRM);

END;

/

```

### Rezultat apelare:

*Prima apelare:*

Înainte

După

SQL   All Rows Fetched: 40 in 0.001 seconds			
COD_CARTE	COD_LIBRARIE	NR_EXEMPLARE	
1	100	40	2
2	115	40	3
3	114	40	1
4	113	40	4
5	103	80	3
6	104	60	5
7	102	50	2
8	104	80	6
9	116	50	4

SQL   All Rows Fetched: 40 in 0.001 seconds			
COD_CARTE	COD_LIBRARIE	NR_EXEMPLARE	
1	100	40	1
2	115	40	3
3	114	40	1
4	113	40	4
5	103	80	3
6	104	60	5
7	102	50	2
8	104	80	6
9	116	50	4

Clientul a platit 30.6 lei pe cartea Invitatie la vals.

PL/SQL procedure successfully completed.

SQL   All Rows Fetched: 15 in 0.001 seconds					
	COD_BON	PRET	DATA_BON	COD_CARTE	COD_ANGAJAT
13	1000	12	04-JAN-24	104	409
14	1001	12	04-JAN-24	104	409
15	120	30.6	05-JAN-24	100	400

*Inserare în tabela BON*

*A doua apelare:*

Task completed in 0.061 seconds	
Eroarea are codul = -20003 si mesajul = ORA-20003: Ne pare rau, nu avem cartea Invitatie la vals pe stoc la locatia Strada Fierului	
PL/SQL procedure successfully completed.	

*A treia apelare:*

Task completed in 0.081 seconds	
Eroarea are codul = -20001 si mesajul = ORA-20001: Nu exista cartea ceruta Carte necunoscuta in baza noastra de date.	
PL/SQL procedure successfully completed.	

### *A patra apelare:*

```
Task completed in 0.076 seconds

Eroarea are codul = -20002 si mesajul = ORA-20002: Nu exista nicio librărie la adresa data - Strada necunoscuta

PL/SQL procedure successfully completed.
```

4 & 5. (Triggeri pe tabel mutating) Trigger de tip LMD la nivel de comandă și Trigger de tip LMD la nivel de linie

### Cerintă:

Se aplică reducere de 10% la cumpărarea unei cărți dacă clientul cumpără una din ultimele 4 exemplare găsite pe stoc per carte (la nivelul tuturor librăriilor). Se va specifica dacă primește reducere printr-un mesaj sugestiv și modificarea de preț care va fi și actualizată pe bon. Trebuie să se aibă grijă și de stoc, adică să se șteargă din listă cartea dacă nu mai sunt exemplare la acea librărie.

### Mod de actualizare:

- Utilitatea triggerilor la nivel de comandă și de linie: se modifica prețul cărții cumpărate de pe bon (din tabelul BON)
- Actualizarea se face o singura de fiecare dată (atunci când cineva cumpără a patra carte, apoi dacă sunt mai puțin de 4 cărți)
- Este nevoie și de un alt trigger care să șteargă cartea din stocul librăriei respectivei în cazul în care nu mai sunt exemplare

```
CREATE OR REPLACE PACKAGE pachet_trigger
IS
    TYPE evidenta IS TABLE OF stocuri.cod_carte%TYPE;
    carti_aplic_reducere evidenta;
    reducere NUMBER := 0;
```

```

        cod_bon bonuri.cod_bon%TYPE;
END;
/

CREATE OR REPLACE TRIGGER retinere_carti_reducere
AFTER INSERT ON bonuri
FOR EACH ROW
DECLARE
    index_carte NUMBER;
begin
    pachet_trigger.reducere := 0;
    pachet_trigger.cod_bon := 0;

    SELECT cod_carte
    BULK COLLECT INTO pachet_trigger.carti_aplic_reducere
    FROM stocuri
    GROUP BY cod_carte
    HAVING SUM(nr_exemplare) < 4;

    index_carte := pachet_trigger.carti_aplic_reducere.FIRST;

    WHILE index_carte <= pachet_trigger.carti_aplic_reducere.LAST AND
pachet_trigger.reducere = 0 LOOP

        IF      pachet_trigger.carti_aplic_reducere(index_carte)      =
:NEW.cod_carte THEN

            pachet_trigger.cod_bon := :NEW.cod_bon;

            pachet_trigger.reducere := 1;

        END IF;

```

```

        index_carte                                     :=
pachet_trigger.carti_aplic_reducere.NEXT(index_carte);

    END LOOP;

END;

/

CREATE OR REPLACE TRIGGER aplicare_reducere
AFTER INSERT ON bonuri
DECLARE
    v_pret bonuri.pret%TYPE;
BEGIN
    IF pachet_trigger.reducere = 1 THEN
        DBMS_OUTPUT.PUT_LINE(' Se aplica reducere de 10% deoarece
cumparati una dintre ultimele 4 exemplare din stoc');

        SELECT pret into v_pret
        FROM bonuri
        WHERE cod_bon = pachet_trigger.cod_bon;

        DBMS_OUTPUT.PUT('Din ' || v_pret || ' pretul devine ');

        UPDATE bonuri
        SET pret = pret - 0.1 * pret
        WHERE cod_bon = pachet_trigger.cod_bon
        RETURNING pret into v_pret;
    
```

```

        DBMS_OUTPUT.PUT_LINE(v_pret ||'.');

    END IF;

END;

/

--trigger la nivel de comanda pentru stergere din stoc daca nu mai
sunt exemplare

CREATE OR REPLACE TRIGGER nu_mai_exista_pe_stoc
AFTER UPDATE of nr_exemplare on stocuri
BEGIN

    DELETE FROM stocuri

    WHERE nr_exemplare = 0;

end;

/

DECLARE

-- cartea nu se gaseste la acea locatie

--     carte carti.nume%type := 'Mireasa pe masura';

--     strada locatii.strada%type := 'Strada Fierului';

-- va primi reducere

    carte carti.nume%TYPE := 'Limanul mult dorit';

    strada locatii.strada%TYPE := 'Strada Fierului';

BEGIN

    DBMS_OUTPUT.PUT_LINE('Clientul          a          platit
'||client_cumpara_carte(carte, strada)||' lei pe cartea '||carte||
'.');

```



```

EXCEPTION

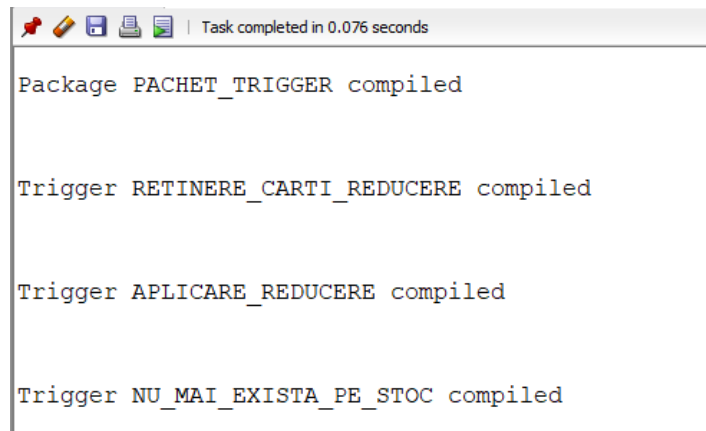
    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE (SQLERRM) ;

END;

/

```



Task completed in 0.076 seconds

```

Package PACHET_TRIGGER compiled

Trigger RETINERE_CARTI_REDCERE compiled

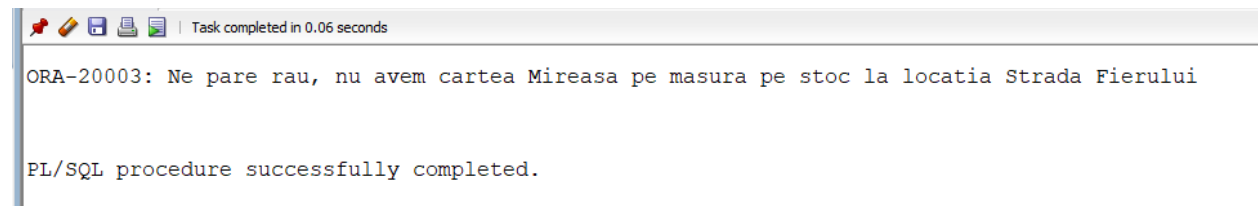
Trigger APLICARE_REDCERE compiled

Trigger NU_MAI_EXISTA_PE_STOC compiled

```

### Rezultat apelare:

*Caz în care cartea dată nu se găsește la locația oferită:*



Task completed in 0.06 seconds

```

ORA-20003: Ne pare rau, nu avem cartea Mireasa pe masura pe stoc la locatia Strada Fierului

PL/SQL procedure successfully completed.

```

*Caz în care cartea dată se găsește la locația oferită în mai mult de patru exemplare (5 exemplare), iar după 2 cumpărări se aplică reducerea:*

Înainte

	COD_CARTE	COD_LIBRARIE	NR_EXEMPLARE
13	102	40	2
14	103	60	3
15	104	50	6
16	105	50	4
17	106	50	4
18	107	50	5
19	108	50	4

	COD_BON	PRET	DATA_BON	COD_CARTE	COD_ANGAJAT
8	107	27	26-FEB-23	104	409
9	108	48.48	13-OCT-22	116	410
10	109	21	29-AUG-20	123	410
11	110	28.5	15-SEP-21	105	402
12	111	35	15-AUG-22	115	400
13	1000	12	04-JAN-24	104	409
14	1001	12	04-JAN-24	104	409
15	120	30.6	05-JAN-24	100	400

După

	COD_CARTE	COD_LIBRARIE	NR_EXEMPLARE
7	103	80	3
8	104	50	6
9	104	60	5
10	104	80	6
11	105	50	4
12	105	70	3
13	106	50	4
14	107	50	3
15	108	50	4

	COD_B...	PRET	DATA_BON	COD_CARTE	COD_ANGAJAT
6	105	43	25-MAR-23	104	409
7	106	28.5	25-MAR-23	102	410
8	107	27	26-FEB-23	104	409
9	108	48.48	13-OCT-22	116	410
10	109	21	29-AUG-20	123	410
11	110	28.5	15-SEP-21	105	402
12	111	35	15-AUG-22	115	400
13	120	30.6	05-JAN-24	100	400
14	176	27	26-JAN-24	107	409
15	177	24.3	26-JAN-24	107	409

```

Task completed in 0.05 seconds

Clientul a platit 27 lei pe cartea Limanul mult dorit.

PL/SQL procedure successfully completed.

Se aplica reducere de 10% deoarece cumparati una dintre ultimele 4 exemplare din stoc
Din 27 pretul devine 24.3.
Clientul a platit 24.3 lei pe cartea Limanul mult dorit.

PL/SQL procedure successfully completed.

```

## 6. Trigger de tip LDD

### Cerintă:

Alți clienți în afara administratorului nu au voie să modifice baza de date din spatele aplicației, iar modificările făcute trebuie să nu intervină cu programul deschis publicului. Așadar pentru schimbări referitoare la obiecte ale schemei sau ale bazei de date, administratorul trebuie să le facă în intervale orare specifice.

Se știe că în zilele lucrătoare, librăriile sunt deschise de la 8:00 la 20:00, sâmbătă de la 10:00 la 19:00, iar duminică de la 12:00 la 16:00.

Triggerul creat va limita modificări la nivel de schema și are nevoie de o procedură pentru a putea insera în tabel modificarea dorită, precum și un mesaj corespunzător.

```
CREATE SEQUENCE SEQ_INFO
INCREMENT by 1
START WITH 1
MAXVALUE 1000
NOCYCLE;

CREATE TABLE info (
    cod NUMBER CONSTRAINT PKEY_INFORMARII PRIMARY KEY,
    utilizator VARCHAR2(30) NOT NULL,
    nume_bd VARCHAR2(50),
    eveniment VARCHAR2(20),
    nume_obiect VARCHAR2(30),
    data DATE NOT NULL,
    mesaj VARCHAR2(40) NOT NULL
);

CREATE OR REPLACE PROCEDURE inserez_in_info(cod NUMBER)
IS
    mesaj VARCHAR2(40);
    PRAGMA autonomous_transaction;
BEGIN
```

```

IF cod = 0 THEN

    mesaj := 'SUCCES';

ELSIF cod = 1 THEN

    mesaj := 'EROARE: IN AFARA PROGRAMULUI';

ELSIF cod = 2 THEN

    mesaj := 'EROARE: NOT ADMIN';

ELSE

    mesaj := 'EROARE NECUNOSCUA';

END IF;


INSERT INTO info

VALUES (SEQ_INFO.NEXTVAL, SYS.LOGIN_USER, SYS.DATABASE_NAME,
SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME, SYSDATE, mesaj);

COMMIT;

END inserez_in_info;

/


CREATE OR REPLACE TRIGGER LDD
BEFORE CREATE OR DROP OR ALTER ON SCHEMA
BEGIN

    --pentru duminica

    IF (TO_CHAR(SYSDATE, 'D') = 1 AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN
12 AND 16))

    OR

    --pentru sambata

    (TO_CHAR(SYSDATE, 'D') = 7 AND (TO_CHAR(SYSDATE, 'HH24') BETWEEN 10
AND 19))

```

```

OR

--pentru zilele lucratoare

(TO_CHAR(SYSDATE, 'D') <> 1 AND to_char(SYSDATE, 'D') <> 7 AND
(TO_CHAR(SYSDATE, 'HH24') BETWEEN 8 AND 20))

THEN

    inserez_in_info(1);

    RAISE_APPLICATION_ERROR(-20000, 'Nu sunt permise modificari
ale schemei in timpul programului cu publicul');

ELSE

    --doar administratorul poate modifica

    IF SYS.login_user <> 'SBD' THEN

        inserez_in_info(2);

        RAISE_APPLICATION_ERROR(-20001, 'Doar administratorul
poate modifica baza de date');

    END IF;

END IF;

inserez_in_info(0);

END;

/

```

**Erori apărute și afișare tabel după inserări:**

Task completed in 0.095 seconds

Error starting at line : 508 in command -

```
CREATE TABLE TEST(i INT)
```

Error report -

ORA-00604: error occurred at recursive SQL level 1

ORA-20001: Doar administratorul poate modifica baza de date

ORA-06512: at line 17

00604. 00000 - "error occurred at recursive SQL level %s"

\*Cause: An error occurred while processing a recursive SQL statement  
(a statement applying to internal dictionary tables).

\*Action: If the situation described in the next error on the stack  
can be corrected, do so; otherwise contact Oracle Support.

Task completed in 0.082 seconds

Error starting at line : 508 in command -

```
CREATE TABLE TEST(i INT)
```

Error report -

ORA-00604: error occurred at recursive SQL level 1

ORA-20000: Nu sunt permise modificari ale schemei in timpul programului cu publicul

ORA-06512: at line 12

00604. 00000 - "error occurred at recursive SQL level %s"

\*Cause: An error occurred while processing a recursive SQL statement  
(a statement applying to internal dictionary tables).

\*Action: If the situation described in the next error on the stack  
can be corrected, do so; otherwise contact Oracle Support.

SQL | All Rows Fetched: 6 in 0.001 seconds

	COD	UTILIZATOR	NUME_BD	EVENTIMENT	NUME_OBIECT	DATA	MESAJ
1	1 SBD	XE	CREATE	TEST	26-JAN-24	SUCCES	
2	2 SBD	XE	DROP	TEST	26-JAN-24	SUCCES	
3	3 SBD	XE	CREATE	TEST	26-JAN-24	EROARE: NOT ADMIN	
4	4 SBD	XE	DROP	TEST	26-JAN-24	EROARE: NOT ADMIN	
5	5 SBD	XE	CREATE	TEST	26-JAN-24	EROARE: IN AFARA PROGRAMULUI	
6	6 SBD	XE	DROP	TEST	26-JAN-24	EROARE: IN AFARA PROGRAMULUI	

## Pachet

```
CREATE OR REPLACE PACKAGE pachet
```

```
IS
```

```
    PROCEDURE categoriile_cele_mai_vandute;
```

```
    PROCEDURE pret_serie;
```

```

        FUNCTION      client_cumpara_carte(p_carte_ceruta      carti.nume%TYPE
DEFAULT 'no_name',

                                p_librarie_solicitanta
locatii.strada%TYPE DEFAULT 'no_adress')

        RETURN carti.pret%TYPE;

END;

/

CREATE OR REPLACE PACKAGE BODY pachet
IS

    PROCEDURE categoriile_cele_mai_vandute
    IS

        -- tip de date record pentru a retine detalii despre o carte
        vanduta

        TYPE info_carte_record IS RECORD (

            nume CARTI.nume%TYPE,

            cod are.cod_carte%TYPE,

            copies_sold NUMBER

        );

        -- tablou imbricat pentru a retine informatii despre toate
        cartile vandute

        TYPE info_carte_tip IS TABLE OF info_carte_record;

        v_carti_vandute info_carte_tip;

        -- tablou imbricat in care se va retine numarul de vanzari a
        fiecărei categorii vandute

        TYPE count_categorie_tip IS TABLE OF NUMBER INDEX BY

```

```

PLS_INTEGER;

    v_count_categorii count_categorie_tip;

    -- Vector in care retin coduri cartilor care se incadreaza
intr-o categories

    TYPE coduri_carti_tip IS VARRAY(100) OF carti.cod_carte%TYPE;
    v_coduri_carti coduri_carti_tip := coduri_carti_tip();

    v_maxim NUMBER := 0;

    v_categorie categorii.gen%TYPE;

BEGIN

    -- Pastrarea cartilor vandute si numarului de vanzari

    SELECT c.nume, b.cod_carte, COUNT(*)
    BULK COLLECT INTO v_carti_vandute
    FROM carti c RIGHT JOIN bonuri b ON c.cod_carte = b.cod_carte
    WHERE EXTRACT(YEAR FROM data_bon) = EXTRACT(YEAR FROM SYSDATE)
    GROUP BY c.nume, b.cod_carte;

    -- determinarea categoriilor care au fost vandute si a
numarului de vanzari

    FOR      index_carte      IN      v_carti_vandute.FIRST      ..
v_carti_vandute.LAST
    LOOP

        FOR categorie IN (

            SELECT DISTINCT cod_categorie cod
            FROM are
            WHERE  v_carti_vandute(index_carte).cod
= are.cod_carte

```



```

        )

        LOOP

            IF v_count_categorii.EXISTS(categorie.cod)

                THEN

                    v_count_categorii(categorie.cod)                :=
v_count_categorii(categorie.cod)                                +
v_carti_vandute(index_carte).copies_sold;

                ELSE

                    v_count_categorii(categorie.cod)                :=
v_carti_vandute(index_carte).copies_sold;

                END IF;

            IF v_count_categorii(categorie.cod) > v_maxim

                THEN

                    v_maxim := v_count_categorii(categorie.cod);

                END IF;

            END LOOP;

        END LOOP;

        IF v_maxim <> 0

            THEN

                DBMS_OUTPUT.PUT_LINE('Numarul maxim de carti dintr-un
anumit tip de categorie vandute: ' || v_maxim);

                DBMS_OUTPUT.PUT_LINE('');

                -- Match-uirea categorie-carte si afisarea informatiilor
dorite

                FOR index_cod_categorie IN v_count_categorii.FIRST ..
v_count_categorii.LAST LOOP

```

```

        IF v_count_categorii.EXISTS(index_cod_categorie) THEN
            IF      v_count_categorii(index_cod_categorie)      =
v_maxim THEN

                SELECT gen into v_categorie
                FROM categorii
                WHERE cod_categorie = index_cod_categorie;

                DBMS_OUTPUT.PUT_LINE('>    Categorie:          ' ||
v_categorie);

                SELECT DISTINCT cod_carte
                BULK COLLECT INTO v_coduri_carti
                FROM are
                WHERE index_cod_categorie = are.cod_categorie;

                FOR    index_cod    IN    v_coduri_carti.FIRST    ..
v_coduri_carti.LAST LOOP

                    FOR    index_carte    IN    v_carti_vandute.FIRST
.. v_carti_vandute.LAST LOOP

                        IF    v_carti_vandute(index_carte).cod    =
v_coduri_carti(index_cod) THEN

                            DBMS_OUTPUT.PUT_LINE(' - ' ||

v_carti_vandute(index_carte).nume ||

                                ': vanduta de

' ||

v_carti_vandute(index_carte).copies_sold || ' ori');

                            END IF;

                        END LOOP;

```

```

                                END LOOP;

                                END IF;

                                END IF;

                                END LOOP;

ELSE
    DBMS_OUTPUT.PUT_LINE('Nu au fost cumparate carti in anul
curent.');
```

```

    END IF;

END categoriile_cele_mai_vandute;

PROCEDURE pret_serie
IS
    CURSOR scriitori_cu_carti IS
        SELECT nume, prenume, cod_scriitor, NVL(aparitii, 0)
        FROM scriitori LEFT JOIN (
            SELECT
                cod_scriitor,
COUNT(DISTINCT(cod_carte)) aparitii
            FROM are
            GROUP BY cod_scriitor
        ) USING (cod_scriitor);

    CURSOR suma_serie (p_cod_scriitor scriitori.cod_scriitor%TYPE)
IS
        SELECT sum(pret) suma, serii.nume
        FROM carti LEFT JOIN serii USING (cod_serie)

```

```

        WHERE cod_serie IS NOT NULL

        AND cod_carte IN (

                                SELECT DISTINCT cod_carte
                                FROM are
                                WHERE          cod_scriitor          =
p_cod_scriitor

                                )

        GROUP BY serii.num;

v_serie serii.num%TYPE;
v_suma NUMBER;
v_cod_scriitor scriitori.cod_scriitor%TYPE;
v_num scriitori.num%TYPE;
v_prenume scriitori.prenume%TYPE;
v_aparitii NUMBER;

BEGIN

    OPEN scriitori_cu_carti;

    LOOP

        FETCH    scriitori_cu_carti    INTO    v_num,    v_prenume,
v_cod_scriitor, v_aparitii;

        EXIT WHEN scriitori_cu_carti%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE (CHR(10)||'Scriitorul          '||v_num||'
'||v_prenume);

        IF v_aparitii <> 0

        THEN

```

```

        OPEN suma_serie(v_cod_scriitor);

        LOOP

            FETCH suma_serie INTO v_suma, v_serie;

            EXIT WHEN suma_serie%NOTFOUND;

            DBMS_OUTPUT.PUT_LINE(CHR(9) || 'Seria
'||v_serie||'" costa '||v_suma|| ' lei.');
```

END LOOP;

```

        IF suma_serie%ROWCOUNT = 0 THEN

            DBMS_OUTPUT.PUT_LINE(CHR(9) || 'Nu exista in
librarie nicio serie scrisa de acest scriitor.');
```

END IF;

```

        CLOSE suma_serie;

    ELSE

        DBMS_OUTPUT.PUT_LINE(CHR(9) || 'Nu exista in librarie
nicio carte scrisa de acest scriitor.');
```

END IF;

END LOOP;

```

    IF scriitori_cu_carti%ROWCOUNT = 0 THEN

        DBMS_OUTPUT.PUT_LINE(CHR(10) || 'Nu exista niciun scriitor
in baza de date.');
```

END IF;

CLOSE scriitori\_cu\_carti;

```

        EXCEPTION

        WHEN OTHERS THEN

            INSERT INTO mesaje

                VALUES (SEQ_ERROR.NEXTVAL, 'A aparut o eroare
necunoscuta', 'E', SYS.LOGIN_USER, SYSDATE);

            RAISE_APPLICATION_ERROR(-20004, 'A aparut o eroare
necunoscuta');

        END pret_serie;


        FUNCTION      client_cumpara_carte(p_carte_ceruta      carti.nume%TYPE
DEFAULT 'no_name',

                                            p_librarie_solicitant
locatii.strada%TYPE DEFAULT 'no_adress')

            RETURN carti.pret%TYPE IS

        v_evidenta NUMBER;

        v_cod_carte stocuri.cod_carte%TYPE;

        v_cod_librarie_care_vinde stocuri.cod_librarie%TYPE;

        v_pret_cumparare carti.pret%TYPE;

        v_bon bonuri%ROWTYPE;

        NU_EXISTA_CARTE EXCEPTION;

        NU_EXISTA_LIBRARIE EXCEPTION;

        NU_EXISTA_CARTEA_PE_STOC EXCEPTION;

        v_mesaj_eroare VARCHAR2(100);

    BEGIN

        SELECT COUNT(*)

            INTO v_evidenta

```

```

FROM carti

WHERE UPPER(ume) = UPPER(p_carte_ceruta);

IF v_evidenta = 0 THEN
    RAISE NU_EXISTA_CARTE;
END IF;

SELECT COUNT(*)
INTO v_evidenta
FROM locatii
WHERE UPPER(strada) = UPPER(p_librarie_solicitanta);

IF v_evidenta = 0 THEN
    RAISE NU_EXISTA_LIBRARIE;
END IF;

UPDATE stocuri

SET nr_exemplare = nr_exemplare - 1

WHERE (cod_carte, cod_librarie) =

    (    -- daca cartea se gaseste in librerie, select-ul
va intoarce o linie

        SELECT cod_carte, st.cod_librarie

        FROM    locatii    loc    JOIN    librarii    lib    ON
(loc.cod_locatie=lib.cod_locatie)

        JOIN (-- selectez toate codurile
librariile in care se gaseste cartea dorita

            SELECT          cod_carte,
cod_librarie

```

```

FROM carti JOIN stocuri
USING (cod_carte)

WHERE UPPER(ume) =
UPPER(p_carte_ceruta)

) st ON(st.cod_librarie =
lib.cod_librarie)

WHERE UPPER(strada) = UPPER
(p_librarie_solicitant)

)

RETURNING cod_carte, cod_librarie INTO v_cod_carte,
v_cod_librarie_care_vinde;

-- UPDATE nu intoarce NO_DATA_FOUND
IF SQL%rowcount = 0 THEN

    RAISE NU_EXISTA_CARTEA_PE_STOC;

END IF;

-- carte este vanduta, se selecteaza pretul pentru a putea
crea bonul

SELECT pret

INTO v_pret_cumparare

FROM carti

WHERE cod_carte = v_cod_carte;

SELECT SEQ_CUMP.NEXTVAL, v_pret_cumparare,

        SYSDATE, v_cod_carte, (

        SELECT cod_angajat

        FROM ( -- se alege in mod random
un vanzator care lucreaza la acea librerie

        -- pentru ca fiecare sa

```



aiba sansa egala de a vinde

```

                                SELECT cod_angajat
                                FROM angajati
                                WHERE      cod_librarie      =
v_cod_librarie_care_vinde AND LOWER(tip_angajat) = 'vanzator'
                                ORDER                                BY
DBMS_RANDOM.RANDOM
                                )
                                WHERE ROWNUM = 1
                                )

    INTO v_bon
    FROM dual;

    INSERT INTO bonuri
    VALUES v_bon;

    -- cartea poate avea reducere, deci preluam din nou bonul
    pentru a-l putea transmite clientului

    SELECT pret
    INTO v_pret_cumparare
    FROM bonuri
    WHERE cod_bon = v_bon.cod_bon;

    RETURN v_pret_cumparare;

    EXCEPTION

        WHEN NU_EXISTA_CARTE THEN

            v_mesaj_eroare := 'Nu exista cartea ceruta '||
```

```

p_carte_ceruta || ' in baza noastra de date.';

        INSERT INTO mesaje

        VALUES      (SEQ_ERROR.NEXTVAL,      v_mesaj_eroare,      'E',
SYS.LOGIN_USER, SYSDATE);

        RAISE_APPLICATION_ERROR(-20001, v_mesaj_eroare);

        WHEN NU_EXISTA_LIBRARIE THEN

                v_mesaj_eroare := 'Nu exista nicio librerie la adresa
data - ' || p_librarie_solicitanta;

        INSERT INTO mesaje

        VALUES      (SEQ_ERROR.NEXTVAL,      v_mesaj_eroare,      'E',
SYS.LOGIN_USER, SYSDATE);

        RAISE_APPLICATION_ERROR(-20002, v_mesaj_eroare);

        WHEN NU_EXISTA_CARTEA_PE_STOC THEN

                v_mesaj_eroare := 'Ne pare rau, nu avem cartea ' ||
p_carte_ceruta ||

                ' pe stoc la locatia ' ||

p_librarie_solicitanta;

        INSERT INTO mesaje

        VALUES      (SEQ_ERROR.NEXTVAL,      v_mesaj_eroare,      'E',
SYS.LOGIN_USER, SYSDATE);

        RAISE_APPLICATION_ERROR(-20003, v_mesaj_eroare);

        WHEN OTHERS THEN

                v_mesaj_eroare := SQLERRM;

        INSERT INTO mesaje

        VALUES      (SEQ_ERROR.NEXTVAL,      v_mesaj_eroare,      'E',
SYS.LOGIN_USER, SYSDATE);

```

```

        RAISE_APPLICATION_ERROR(-20004, v_mesaj_eroare);

    END client_cumpara_carte;

END;

/

DECLARE

--      SUCCES

--      carte carti.nume%TYPE := 'Mireasa pe masura';

--      strada locatii.strada%TYPE := 'Strada Doamnei';

--      Eroarea are codul = -20003 si mesajul = ORA-20003: Ne pare rau,
nu avem cartea Invitatie la vals pe stoc la locatia Strada Fierului

        carte carti.nume%TYPE := 'Invitatie la vals';

        strada locatii.strada%TYPE := 'Strada Fierului';

--      Eroarea are codul = -20001 si mesajul = ORA-20001: Nu exista
cartea ceruta Carte necunoscuta in baza noastra de date.

--      carte carti.nume%TYPE := 'Carte necunoscuta';

--      strada locatii.strada%TYPE := 'Strada Doamnei';

--      Eroarea are codul = -20002 si mesajul = ORA-20002: Nu exista
nicio librerie la adresa data - Strada necunoscuta

--      carte carti.nume%TYPE := 'Invitatie la vals';

--      strada locatii.strada%TYPE := 'Strada necunoscuta';

BEGIN

    pachet.categoriile_cele_mai_vandute;

```

```

    pachet.pret_serie;

    DBMS_OUTPUT.PUT_LINE('Clientul      a      platit      ' ||
pachet.client_cumpara_carte(carte, strada) ||

                        ' lei pe cartea ' || carte || '.');

EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('Eroarea are codul = '||SQLCODE || '
si mesajul = ' || SQLERRM);

END;

/

```

Numarul maxim de carti dintr-un anumit tip de categorie vandute: 6

> Categorie: fictiune

- Mireasa pe masura: vanduta de 2 ori
- Limanul mult dorit: vanduta de 2 ori
- Mostenirea: vanduta de 2 ori

> Categorie: crestin

- Mireasa pe masura: vanduta de 2 ori
- Limanul mult dorit: vanduta de 2 ori
- Mostenirea: vanduta de 2 ori

Scriitorul Oke Janette

- Seria "Anotimpurile inimii" costa 87.4 lei.
- Seria "Cantecul Acadiei" costa 136.5 lei.
- Seria "Faptele credintei" costa 105 lei.

Scriitorul Witemeyer Karen

Nu exista in librerie nicio serie scrisa de acest scriitor.

Scriitorul Dickens Charles

Nu exista in librerie nicio serie scrisa de acest scriitor.

Scriitorul Bunn T. Davis

- Seria "Cantecul Acadiei" costa 136.5 lei.
- Seria "Faptele credintei" costa 105 lei.

Scriitorul Rowling J.K.

- Seria "Harry Potter" costa 390.88 lei.

Scriitorul Drumes Mihail

Nu exista in librerie nicio serie scrisa de acest scriitor.

Scriitorul Dumas Alexandre

- Seria "Cei trei muschetari" costa 54 lei.

Scriitorul Tolstoi Lev

Nu exista in librerie nicio carte scrisa de acest scriitor.

Eroarea are codul = -20003 si mesajul = ORA-20003: Ne pare rau, nu avem cartea Invitatie la vals pe stoc

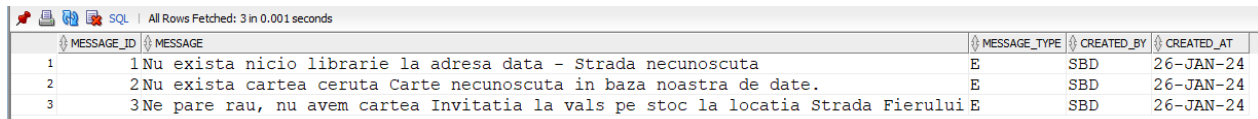
PL/SQL procedure successfully completed.

## Înregistrarea excepțiilor apărute în tabelul MESAJE

Inserarea în tabelul MESAJE se face în exercițiile anterioare.

```
CREATE SEQUENCE SEQ_ERROR  
  
INCREMENT by 1  
  
START WITH 1  
  
MAXVALUE 1000  
  
NOCYCLE;
```

### Rezultat apelare:



The screenshot shows a SQL query result with 3 rows. The status bar at the top indicates 'All Rows Fetched: 3 in 0.001 seconds'. The table has 5 columns: MESSAGE\_ID, MESSAGE, MESSAGE\_TYPE, CREATED\_BY, and CREATED\_AT.

MESSAGE_ID	MESSAGE	MESSAGE_TYPE	CREATED_BY	CREATED_AT
1	1Nu exista nicio librerie la adresa data - Strada necunoscuta	E	SBD	26-JAN-24
2	2Nu exista cartea ceruta Carte necunoscuta in baza noastra de date.	E	SBD	26-JAN-24
3	3Ne pare rau, nu avem cartea Invitatie la vals pe stoc la locatia Strada Fierului E	E	SBD	26-JAN-24