**Proiectarea Bazelor de date**

**Student:**  Pistritu Ionut-Laurentiu

**Cerinţe:**

1. Să se realizeze modelul entitate-legătură

2. Să se facă trecerea la modelul relaţional (să se detalieze operaţiile effectuate pentru fiecare entitate în parte)

3. Să se creeze o bază de date care să conţină:

a. Legături între tabele;

b. Integritatea referenţială pentru a efectua corect operaţiile de actualizare (adăugare, ştergere, modificare);

c. Chei primare şi chei externe;

d. Reguli de ştergere şi modificare în cascadă pentru entităţile cu chei externe.

4. Să se verifice calitatea proiectării bazei de date cu ajutorul formelor normale (se va normaliza baza de date până la FNBC - inclusiv).

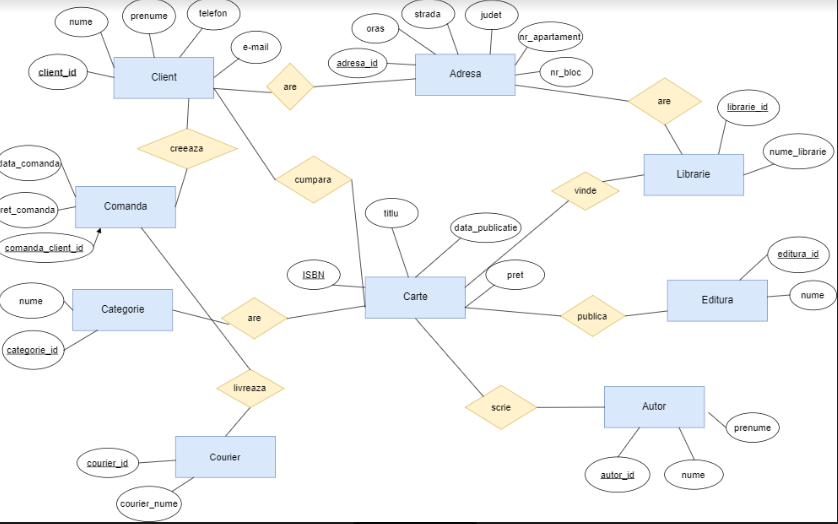
5. Să se indexeze baza de date creată (crearea minim a unui index).

6. Să se facă testarea bazei de date prin efectuarea de diferite operaţii asupra ei. Trebuie să existe cel puţin o interogare din fiecare tip (în total minim 15 interogări), 2 funcţii diferite (care să nu facă un simplu select), 2 proceduri stocate, şi 1 trigger.

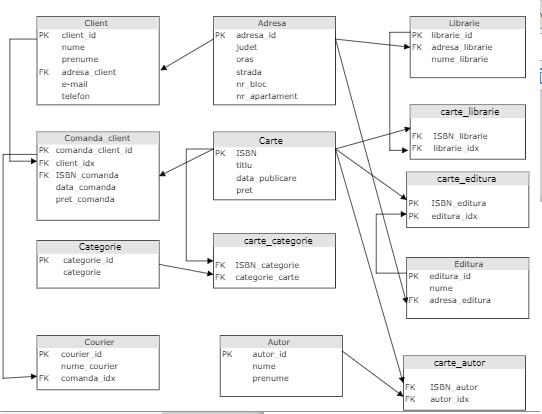
**Proiect:**

33) Să se creeze o bază de date care să ţină evidenţa unei librării. În librărie există cărţi care au data publicării, titlu, ISBN şi un preţ . Fiecare carte poate avea unul sau mai mulţi autori identificaţi prin nume şi prenume. Fiecare autor poate scrie una sau mai multe cărţi. Fiecare carte poate aparţine unei anumite categorii. Există o listă cu clienţii librăriei, care au un cod, un nume, o adresă, un telefon şi un email. Fiecare client poate face una sau mai multe comenzi ce pot conţine una sau mai multe cărţi, un număr al comenzii şi o sumă totală.

**Modelul entitate-legatura**



**Modelul relational**



**Baza de date**

CREATE TABLE adresa

(

adresa\_id NUMBER NOT NULL,

judet VARCHAR2(45),

oras VARCHAR2(45),

strada VARCHAR2(45),

nr\_bloc NUMBER,

nr\_apartament NUMBER,

PRIMARY KEY (adresa\_id)

);

CREATE TABLE carte

(

ISBN NUMBER NOT NULL,

titlu VARCHAR2(45),

data\_publicatie VARCHAR2(45),

pret NUMBER(10 ,3),

PRIMARY KEY (ISBN)

);

CREATE TABLE librarie

(

librarie\_id NUMBER NOT NULL,

nume\_librarie VARCHAR2(45),

adresa\_librarie NUMBER,

PRIMARY KEY (librarie\_id),

FOREIGN KEY (adresa\_librarie) REFERENCES adresa(adresa\_id)

);

CREATE TABLE carte\_librarie

(

ISBN\_librarie NUMBER not null,

librarie\_idx NUMBER not null,

FOREIGN KEY (ISBN\_librarie) REFERENCES carte(ISBN),

FOREIGN KEY (librarie\_idx) REFERENCES librarie(librarie\_id),

PRIMARY KEY (ISBN\_librarie, librarie\_idx)

);

CREATE TABLE client

(

client\_id NUMBER NOT NULL,

nume VARCHAR2(45),

prenume VARCHAR2(45),

adresa\_client NUMBER NOT NULL,

email VARCHAR2(45),

telefon NUMBER(10),

PRIMARY KEY (client\_id),

FOREIGN KEY (adresa\_client) REFERENCES adresa(adresa\_id)

);

CREATE TABLE comanda\_client

(

comanda\_client\_id NUMBER NOT NULL,

clinet\_idx NUMBER NOT NULL,

ISBN\_comanda NUMBER NOT NULL,

data\_comanda VARCHAR2(45),

pret\_comanda NUMBER(10 ,3),

PRIMARY KEY (comanda\_client\_id),

FOREIGN KEY (ISBN\_comanda) REFERENCES carte(ISBN)

);

CREATE TABLE categorie

(

categorie\_id NUMBER NOT NULL,

categorie VARCHAR2(45),

PRIMARY KEY (categorie\_id)

);

CREATE TABLE carte\_categorie

(

ISBN\_categorie NUMBER not null,

carte\_categorie NUMBER not null,

FOREIGN KEY (ISBN\_categorie) REFERENCES carte(ISBN),

FOREIGN KEY (carte\_categorie) REFERENCES categorie(categorie\_id),

PRIMARY KEY (ISBN\_categorie, carte\_categorie)

);

CREATE TABLE editura

(

editura\_id NUMBER NOT NULL,

nume VARCHAR2(45),

adresa\_editura NUMBER NOT NULL,

PRIMARY KEY (editura\_id),

FOREIGN KEY (adresa\_editura) REFERENCES adresa(adresa\_id)

);

CREATE TABLE carte\_editura

(

ISBN\_editura NUMBER not null,

editura\_idx NUMBER not null,

FOREIGN KEY (ISBN\_editura) REFERENCES carte(ISBN),

FOREIGN KEY (editura\_idx) REFERENCES editura(editura\_id),

PRIMARY KEY (ISBN\_editura, editura\_idx)

);

CREATE TABLE autor

(

autor\_id NUMBER NOT NULL,

nume VARCHAR2(45),

prenume VARCHAR2(45),

PRIMARY KEY (autor\_id)

);

CREATE TABLE carte\_autor

(

ISBN\_autor NUMBER not null,

autor\_idx NUMBER not null,

FOREIGN KEY (ISBN\_autor) REFERENCES carte(ISBN),

FOREIGN KEY (autor\_idx) REFERENCES autor(autor\_id),

PRIMARY KEY (ISBN\_autor, autor\_idx)

);

CREATE TABLE courier

(

courier\_id NUMBER NOT NULL,

nume\_courier VARCHAR2(45),

comanda\_idx NUMBER NOT NULL,

PRIMARY KEY (courier\_id),

FOREIGN KEY (comanda\_idx) REFERENCES comanda\_client(comanda\_client\_id)

);

**Normalizari**

FN1 – se afla toate, doarece la fiecare intersecție de rânduri și coloane din tabel există o singură valoare și niciodată o listă de valori.

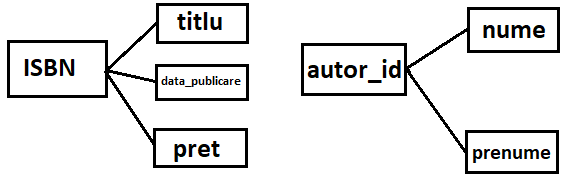
***Exemplu:***

adresa



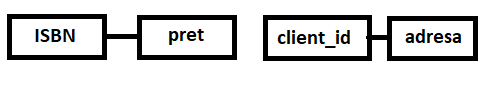
FN2 – se afla toate, deoarece fiecare coloană non-cheie trebuie să fie complet dependentă de cheia primară întreagă, nu doar de o parte din cheie.

***Exemplu:***



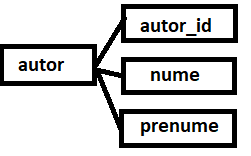
FN3 – se afla toate, deoarece nu doar fiecare coloană non-cheie trebuie să depindă de întreaga cheie primară, ci coloanele non-cheie trebuie să fie independente una de alta.

***Exemplu:***



FNBC – o relaţie este în forma normală Boyce-Codd dacă şi numai dacă orice determinant din relaţie este cheie candidat.

***Exemplu:***



**Index**

CREATE INDEX carte\_idx ON carte (ISBN);

CREATE INDEX categorie\_idx ON categorie (categorie\_id);

CREATE INDEX carte\_editura\_idx ON carte\_editura (ISBN\_editura, editura\_idx);

**Proceduri**

CREATE OR REPLACE PROCEDURE creste\_suma

(p\_id IN comanda\_client.comanda\_client\_id%Type,

p\_percent IN NUMBER)

IS

BEGIN

UPDATE comanda\_client SET pret\_comanda= pret\_comanda \*(1+p\_percent/100)

Where comanda\_client\_id = p\_id;

END creste\_suma;

**Functii:**

- CREATE OR REPLACE FUNCTION get\_suma

(p\_id number(10,3))

RETURN NUMBER IS v\_sal number;

BEGIN

SELECT pret\_comanda INTO v\_sal FROM comanda\_client WHERE canda\_client\_id = p\_id;

RETURN v\_sal;

END get\_suma;

- CREATE OR REPLACE FUNCTION update\_tip

RETURN VARCHAR2(45)

IS

v\_up VARCHAR2(45);

BEGIN

UPDATE editura

SET tip='Diverta'

WHERE editura\_id='7'

RETURN v\_up;

END update\_tip;

**Trigger**

CREATE OR REPLACE TRIGGER after\_trigger

AFTER INSERT ON carte

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Cartea a fost updatata');

END;

**Interogari:**

1. SELECT \* FROM client ORDER BY nume DESC;

2. UPDATE categorie SET categorie='razboi' WHERE categorie\_id='5';

3. DELETE oras FROM adresa WHERE adresa\_id=12;

4. SELECT MAX(pret) AS PretMaxCarte FROM carte;

5. SELECT COUNT(ISBN) FROM carte WHERE titlu='Viorica';

6. SELECT \* FROM editura WHERE nume = ' Suntem buni';

7. SELECT nume, prenume FROM client WHERE client\_id > 5;

8. SELECT \* FROM comanda\_client WHERE (pret\_comanda BETWEEN 200 AND 900) AND NOT comanda\_client\_id IN (6 ,7 ,8 ,9);

9. SELECT COUNT(adresa\_id), strada FROM adresa GROUP BY oras;

10. SELECT COUNT(client\_id), prenume FROM client GROUP BY nume

HAVING COUNT(client\_id) > 5 ORDER BY COUNT(client\_id) DESC;

11. SELECT nume\_librarie FROM librarie WHERE EXISTS (SELECT titlu FROM carte);

12. SELECT AVG(pret) FROM carte WHERE pret BETWEEN 40 AND 500;

13. SELECT carte.titlu, categorie.categorie FROM carte\_categorie INNER JOIN carte ON carte\_categorie.ISBN\_categorie = carte.ISBN INNER JOIN categorie ON carte\_categorie.categorie\_carte = categorie.categorie;

14. SELECT \* FROM adresa ORDER BY strada;

15. SELECT email FROM client WHERE client\_id IS NOT NULL;