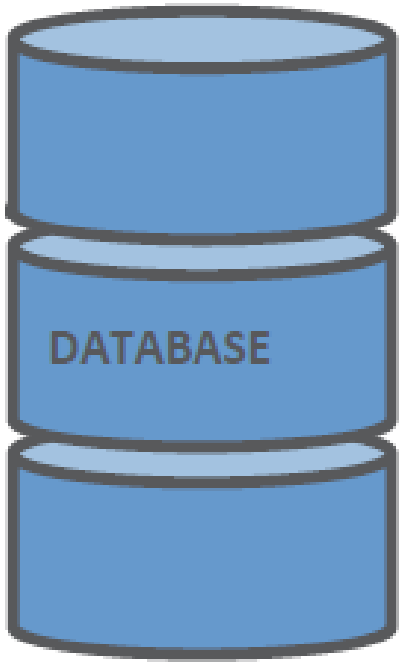


Baze de date

MODULUL 1
Proiectarea bazelor de date

Colegiul Național “Petru Rareş” Suceava
prof. Marius UDUDEC

Introducere în baze de date

- Noțiunile de dată și informație
- Conceptul de bază de date

Noțiunile de dată și informație

***Data**:*

- materie brută
- materie primară
- valoare neprelucrată
- fără un înțeles de sine stătător
- culeasă din lumea reală pe bază de observații și măsurători
- memorată

Exemple:

30 lei, 2, spectacol

***Informații**:*

- prin prelucrarea datelor și stabilirea unor relații între ele se obțin informațiile
- date prelucrate (rapoarte, statistici, diagrame, clasamente, etc.)

Exemplu:

Spectacolul durează *2* ore, iar prețul unui bilet este de *30 lei*.

Conceptul de bază de date

***Bază de date**:*

- set de date corelate și organizate în scopul prelucrării lor rapide și concomitente de către mai multe persoane

Gestionarea bazelor de date:

***SGBD - S**isteme de **G**estiune a **B**azelor de **D**ate*

Exemple de SGBD:

- Microsoft Access
- Microsoft SQL Server
- MySQL
- Visual Fox Pro
- Oracle
- IBM DB2

Modelul conceptual al unei probleme de gestiune

- Model conceptual
- Etapele procesului de dezvoltare al bazelor de date
- Entităţi şi instanţe
- Atribute
- Identificator unic
- Relaţii între entităţi (one-to-one, one-to-many, many-to-many)
- Convenții de reprezentare a relațiilor
- Relații ierarhice și relații recursive
- Relații redundante
- Relații exclusive / arce
- Relații nontransferabile
- Rezolvarea relaţiilor many-to-many
- Normalizarea datelor: prima forma normală, a doua forma normală, a treia formă normală

Model conceptual

- modelarea datelor reprezintă primul pas în procesul de dezvoltare a unei baze de date
- procesul de dezvoltare a bazelor de date are la bază modelul conceptual (modelul entități-relații)
- informațiile necesare derulării unei afaceri sunt reprezentate în diagrama entități-relații (ERD – Entity Relationships Diagram) numită și harta relațiilor
- informațiile reprezentate în ERD sunt mapate (transformate) într-o bază de date relațională utilizând tabele grafice

Etapele procesului de dezvoltare al bazelor de date

Strategie

determinarea nevoilor afacerii și colectarea datelor

Analiza

realizarea modelului conceptual

Proiectare

definirea tabelelor

Construire

realizarea bazei de date folosind un SGBD

Aplicație demonstrativă

Gestionarea informațiilor referitoare la o orchestră

- *întocmirea scenariul afacerii*
- *realizarea modelul conceptual*
- *proiectarea bazei de date*

Scenariul afacerii

*În vederea gestionării informațiilor referitoare la o orchestră se creează baza de date cu denumirea **Orchestra**.*

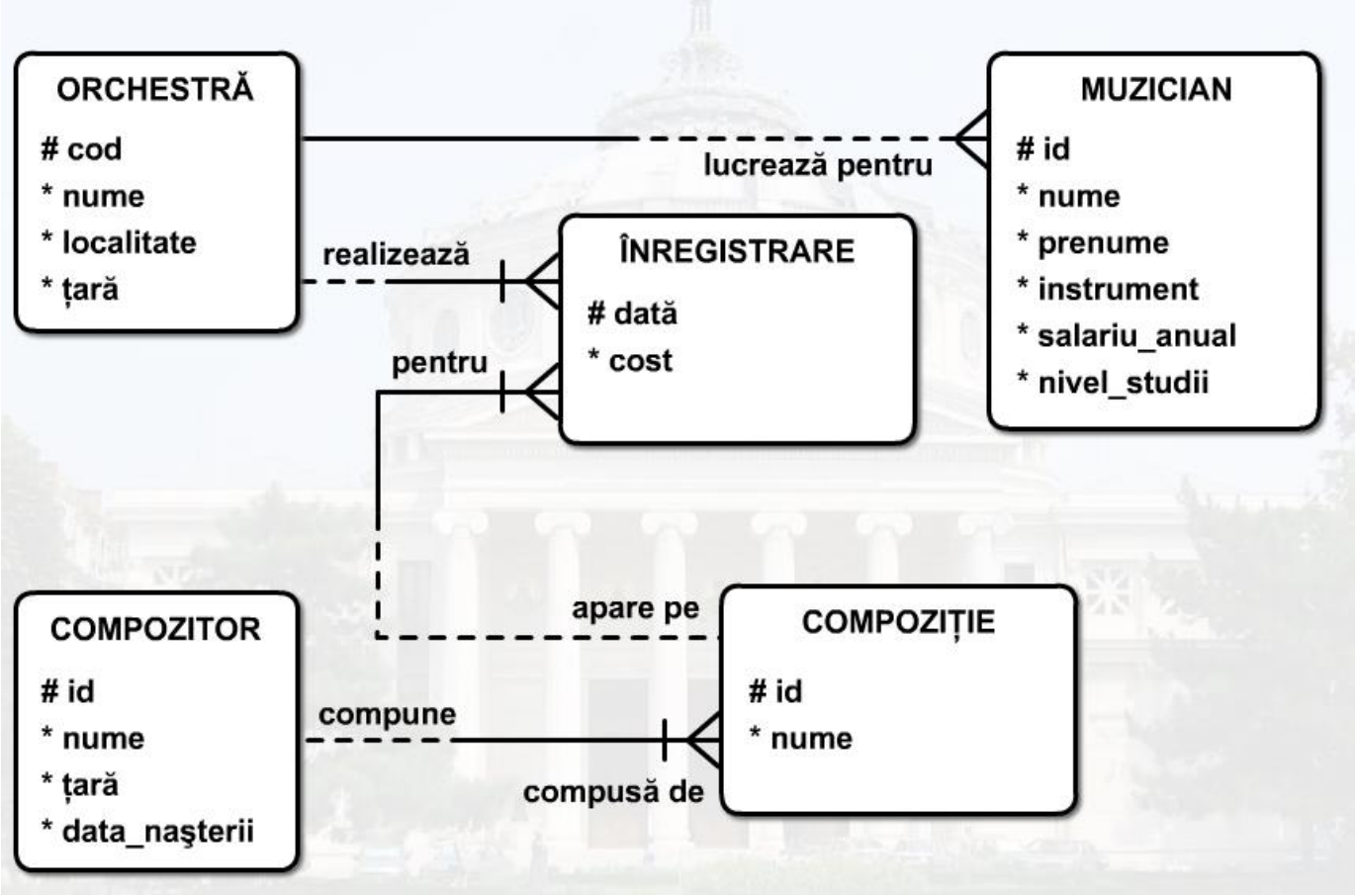
Orchestra are un cod de identificare și un nume. Pentru orchestră se mai cunosc țara și orașul în care își are sediul. Această orchestră este formată din mai mulți muzicieni, pentru fiecare muzician cunoscându-se id-ul, numele, prenumele, instrumentul la care cântă, studiile deținute precum și salariul anual obținut.

În această bază de date se înregistrează și unele informații legate de compozitorii abordați (codul, numele, data nașterii și țara de proveniență) și compozițiile acestora (identificatorul compoziției și numele compoziției).

Deoarece orchestra face și înregistrări audio, pentru fiecare astfel de înregistrare se va memora data și costul înregistrării.

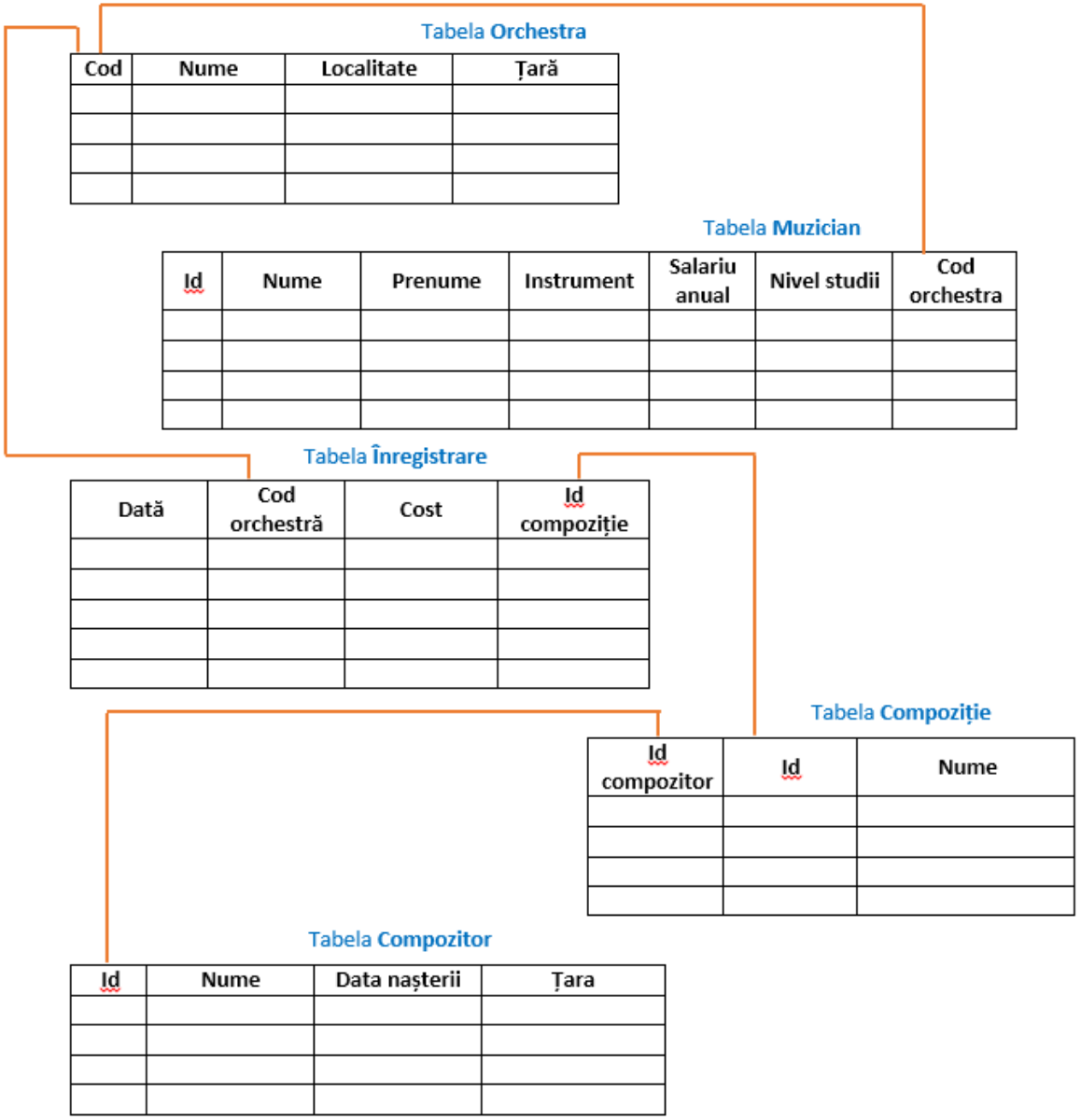
Modelul conceptual

Se întocmește diagrama entități-relații (harta relațiilor).



Baza de date

În procesul de mapare, entitățile și relațiile se transformă în obiecte ale bazei de date.



Entităţi şi instanţe

Orice bază de date foloseşte entităţi pentru a clasifica „obiectele” pe care le gestionează.

☞ ***Exemplu:***

Baza de date a școlii poate avea entitățile:

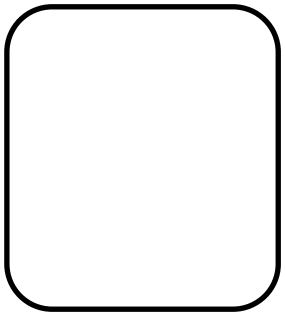
- *elev*
- *profesor*
- *disciplină*
- *clasă*

O **entitate** este un lucru, obiect, persoană, activitate, fenomen sau eveniment care are semnificaţie pentru afacerea modelată, despre care trebuie colectate şi memorate date.

☞ ***Exemple:***

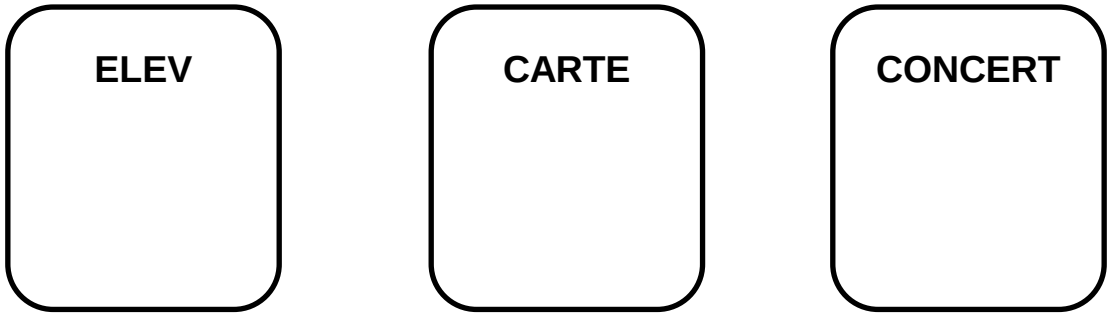
- *carte*
- *elev*
- *școală*
- *concert*
- *operație*

O entitate este reprezentată în ERD printr-un *dreptunghi cu colţurile rotunjite*, numit ***soft box***.



Numele entităţii este întotdeauna un substantiv la singular şi se scrie cu majuscule în partea de sus a dreptunghiului.

☞ ***Exemple:***



Entităţile au ***instanţe***, adică valori particulare.

O entitate reprezintă o clasă de obiecte şi pentru fiecare entitate există mai multe ***instanţe*** ale sale.

O instanţă a unei entităţi este un obiect, persoană, eveniment, individual (particular) din clasa de obiecte care formează entitatea.

☞ ***Exemple:***

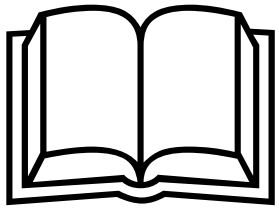
Elevul Popescu Dan este o instanță a entității ELEV.

Fiecare angajat al unei bănci este o instanță a entității ANGAJAT.

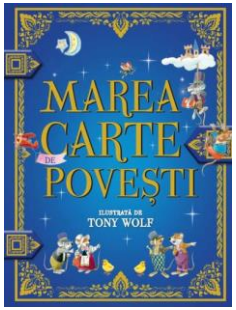
Entitate = mulțimea tuturor elementelor de un anumit tip

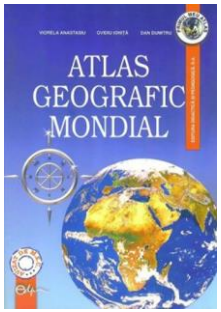
Instanță = un singur element, bine individualizat, unic, din mulțimea elementelor care formează entitatea respectivă

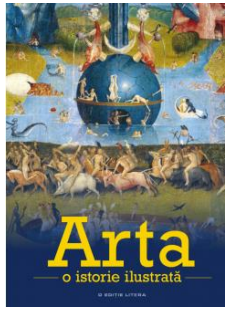
☞ ***Exemplu:***



entitatea CARTE







instanțe ale entității CARTE

Subentități:

- o entitate poate avea subentități, numite și subtipuri

- o subentitate este o clasificare a unei entități care are caracteristici și atribute comune cu entitatea generală

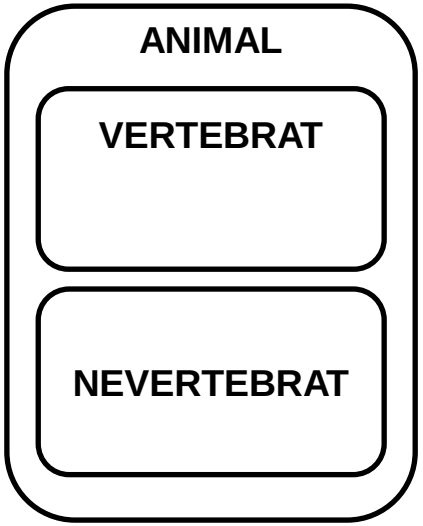
- în ERD o subentitate este o entitate inclusă în interiorul altei entități

- o entitate niciodată nu este singulară (trebuie să existe cel puțin două subentități într-o entitate)

☞ ***Exemplu:***

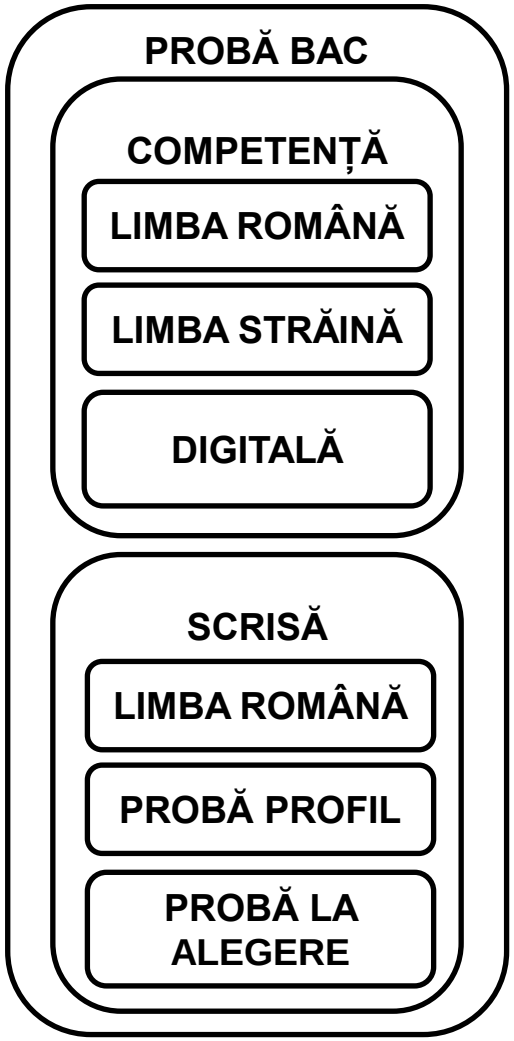
Clasificarea animalelor:

- animale vertebrate
- animale nevertebrate



O subentitate poate avea la rândul său alte subentități.

☞ ***Exemplu:***



✍ ***Exerciții:***

1. Dați exemple de entități specifice unei farmacii.
2. Formulați exemple de trei entități și de trei instanțe pentru fiecare entitate.
3. Dați un exemplu de o entitate formată din subentități.

Atribute

Un ***atribut*** reprezintă o caracteristică a unei entități.

Atributele unei entități sunt informații specifice care trebuie cunoscute și memorate.

Un atribut se identifică prin nume și pentru fiecare instanță a unei entități poate avea o valoare și numai una.

☞ ***Exemple:***

Entitatea REVISTĂ poate avea atributele:

- *cod*
- *nume*
- *număr apariții*
- *număr pagini*
- *preț*

Atributele sunt substantive la singular scrise cu litere mici în entitate, sub numele entității.

☞ ***Exemplu:***

CARTE

titlu
autor
data_nașterii
format
număr_pagini

Atributul are o valoare de un anumit tip (format), care poate fi:

- numeric
- șir de caractere
- dată calendaristică
- etc.

☞ ***Exemplu:***

Entitatea FILM

Atribut	Tip	Valoare
titlu	șir de caractere	Titanic
regizor	șir de caractere	James Cameron
data_apariției	dată calendaristică	01.11.1997
durata_minute	numeric	210

Clasificarea atributelor în funcție de valoare:

- atribute obligatorii
- atribute opționale

☞ ***Exemplu:***

MAȘINĂ

serie motor
marca
model
consum
număr locuri
culoare

Atribute obligatorii:

- au o valoare pentru fiecare instanță a entității (valoare obligatorie)
- sunt precedate în ERD de simbolul asterisc (*)
- au opțiunea NOT NULL (valoarea atributului nu poate să lipsească)

☞ ***Exemplu:***

Atribute opționale:

- valoarea atributului poate să lipsească (valoare opțională)
- sunt precedate în ERD de simbolul cerculeț (o)
- au opțiunea NULL (valoarea atributului nu este cunoscută la un moment dat sau nu este aplicabilă)

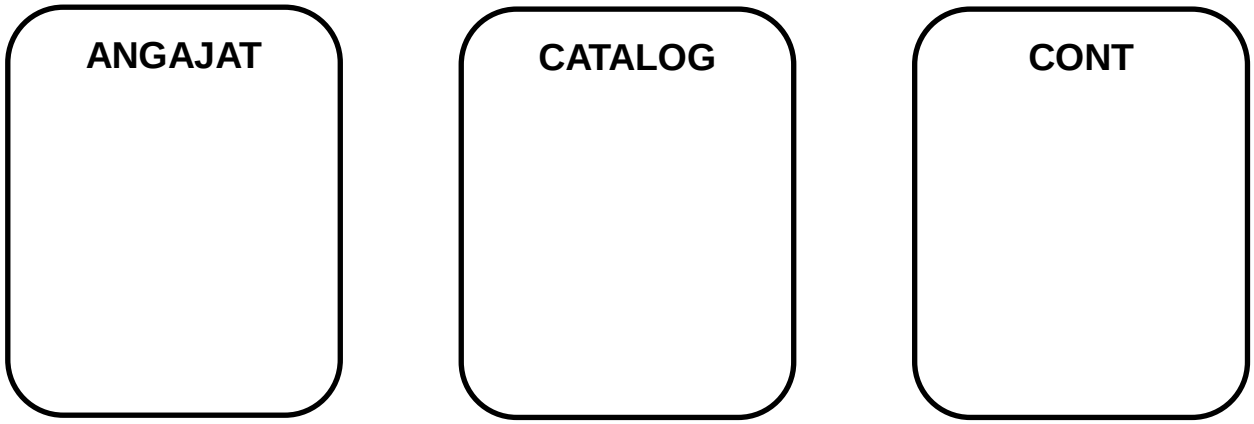
☞ ***Exemplu:***

MAȘINĂ

serie motor
marca
model
consum
o număr locuri
o culoare

✍ ***Exercițiu:***

Pentru fiecare dintre următoarele entități completați lista atributelor. Specificaţi pentru fiecare atribut caracterul pe care îl are: obligatoriu/opţional, volatil/nevolatil



Clasificarea atributelor în funcție de variația valorilor:

- atribute volatile
- atribute nevolatile

☞ ***Exemplu:***

MAȘINĂ

serie motor
marca
model
consum
număr locuri
culoare

Atribute volatile:

- valorile se schimbă frecvent și automat

☞ ***Exemple:***

- vârsta unei persoane
- stocul medicamentelor dintr-o farmacie
- cursul valutar

Atribute nevolatile:

- valorile nu se schimbă (valori fixe)

☞ ***Exemple:***

- data nașterii unei persoane
- locația unei păduri
- serie motor

Identificator unic

Identificatori unici:

- UID – Unique IDentifier
- definesc în mod unic instanțele unei entități
- sunt întotdeauna obligatorii
- în diagrama entități-relații sunt precedate de simbolul diez (#)

☞ ***Exemplu:***

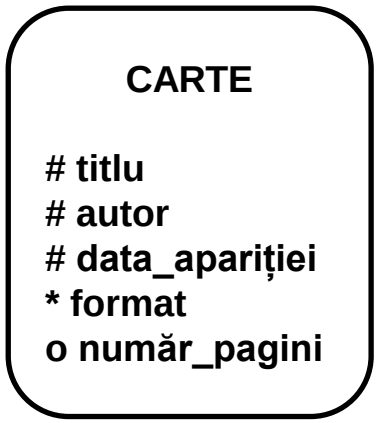
Un identificator unic poate fi:

- simplu (format dintr-un singur atribut)
- compus (format dintr-o combinație de două sau mai multe atribute)

☞ ***Exemple:***

UID simplu

UID compus



✍ ***Exercițiu:***

Se dau entitățile ECHIPĂ și JUCĂTOR. Rezolvați următoarele sarcini:

- reprezentați grafic entitățile date;
- enumerați pentru fiecare entitate câteva atributele specifice;
- indicați simbolul care trebuie să preceadă fiecare atribut;
- scrieți tipul valorii pentru fiecare atribut.

Relaţii între entităţi

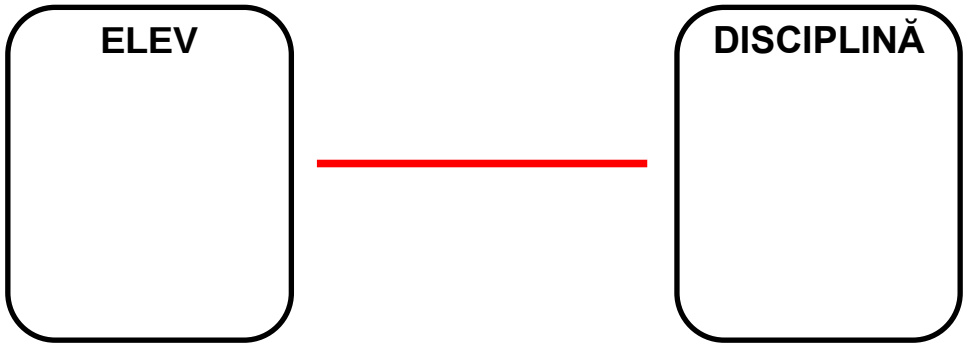
Atunci când un atribut al unei entități face referire la o altă entitate din baza de date se stabilește o *relație* între cele două entități

O ***relație*** este o asociere, o legătură sau conexiune existentă între entități și care are semnificație pentru afacerea modelată.

Relațiile sunt întotdeauna bidirecționale (se citesc de la stânga la dreapta și de la dreapta la stânga) și pot fi:

- binare (stabilite între două entități);
- recursive (de la o entitate la ea însăși).

☞ ***Exemple:***

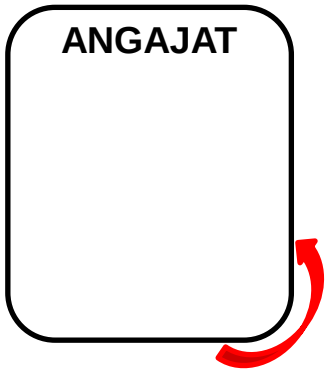


Relații binare:

Fiecare ELEV studiază mai multe DISCIPLINE.

Fiecare DISCIPLINĂ este studiată de mai mulți ELEVI.

☞ ***Exemplu:***

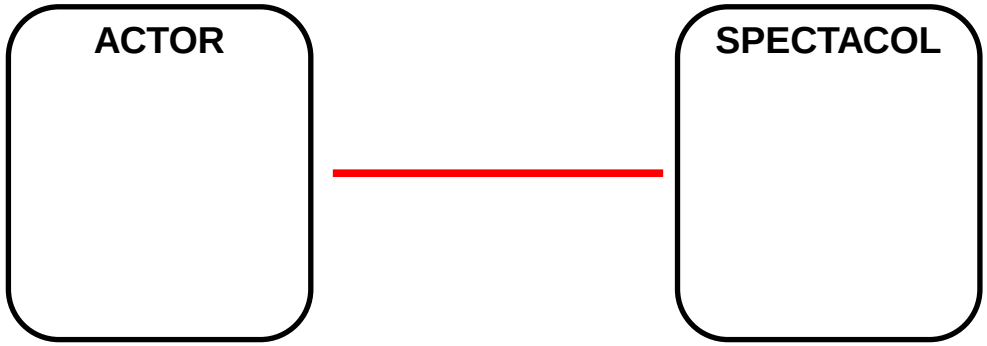


Relație recursivă:

Fiecare ANGAJAT conduce mai mulți ANGAJAȚI.

O relație se identifică prin numele relației care este un *verb* sau o *expresie verbală*.

☞ ***Exemplu:***



Relații:

Fiecare ACTOR joacă în SPECTACOL.

Fiecare SPECTACOL poate avea mai mulți ACTORI.

Relațiile se caracterizează prin:

- cardinalitate (grad);
- opționalitate.

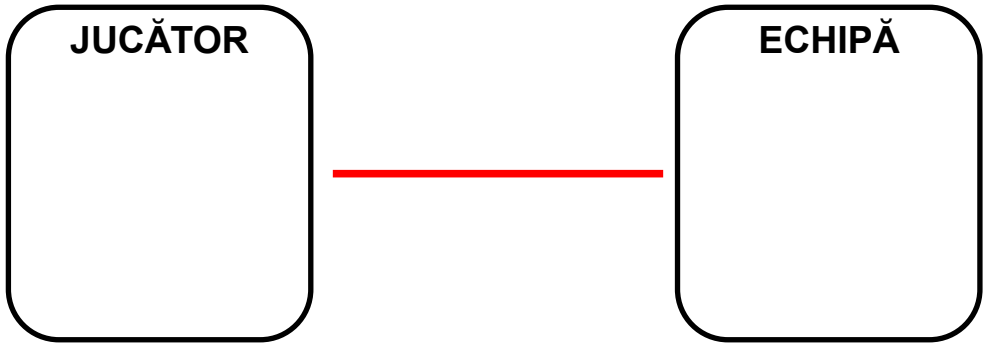
Cardinalitatea (gradul) relațiilor

Cardinalitatea unei relații este dată de numărul de instanțe ale unei entități care pot intra în relație cu o instanță a altei entități.

Cardinalitatea poate fi exprimată în două moduri:

- ”*unul și numai unul*” (unei instanțe a unei entități îi corespunde o instanță a altei entități);
- ”*unul sau mai mulți*” (unei instanțe a unei entități îi pot corespunde una sau mai multe instanțe ale altei entități).

☞ ***Exemplu:***



Relații:

Fiecare JUCĂTOR joacă la o ECHIPĂ și numai una.

Fiecare ECHIPĂ poate avea unul sau mai mai mulți JUCĂTORI.

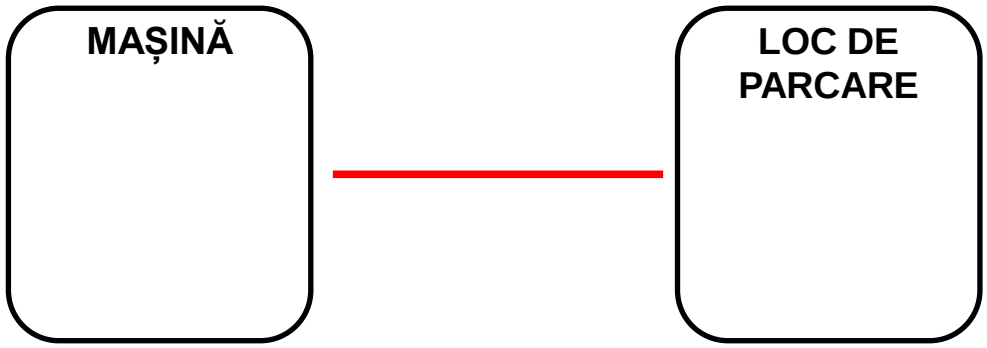
Opționalitatea relațiilor

Opționalitatea unei relații se referă la caracterul unei relații de a fi:

- relație obligatorie (relație sigură - orice instanță a entității trebuie să fie legată de una sau mai multe instanțe ale celeilalte entități);
- relație opțională (relație posibilă - orice instanță a entității ar putea să fie legată de una sau mai multe instanțe ale celeilalte entități).

O relație opțională se exprimă prin cuvântul ***poate***, iar o relație obligatorie se exprimă prin cuvântul ***trebuie***.

☞ ***Exemplu:***



Relații:

Fiecare MAȘINĂ poate avea un LOC DE PARCARE.

Fiecare MAȘINĂ trebuie să aibă un LOC DE PARCARE.

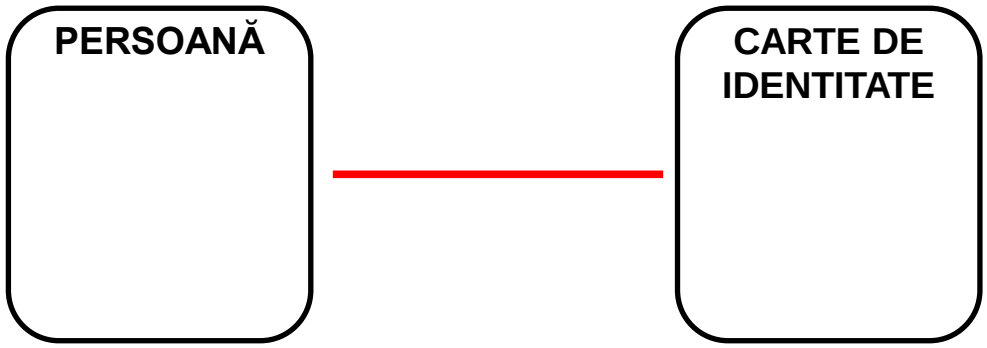
Tipuri de relații

- relații unu-la-unu (one-to-one);
- relații unu-la-mai-mulți (one-to-many);
- relații (mai-mulți-la-mai-mulți) many-to-many.

Relații unu-la-unu (one-to-one)

- fiecărei instanțe a unei entități îi corespunde cel mult o instanță a altei entități
- se notează 1:1
- o relație 1:1 este cel mai puțin tip de relație întâlnit
- uneori o relație 1:1 poate fi modelată prin transformarea unei entități în atribut al altei entități

☞ ***Exemplu:***



Relații:

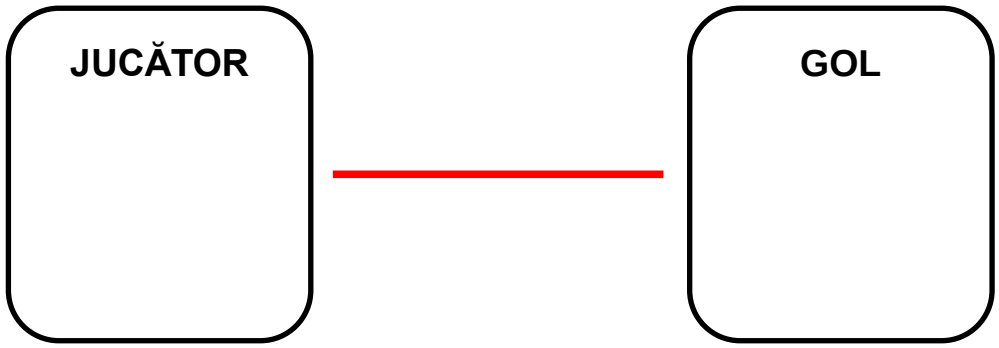
Fiecare PERSOANĂ are o CARTE DE IDENTITATE și numai una.

Fiecare CARTE DE IDENTITATE aparține unei PERSOANE și numai uneia.

Relații unu-la-mai-mulți (one-to-many)

- fiecărei instanțe a unei entități îi corespunde una sau mai multe instanțe ale altei entități
- se notează 1:M sau M:1
- o relație 1:M este cel mai întâlnit tip de relație

☞ ***Exemplu:***



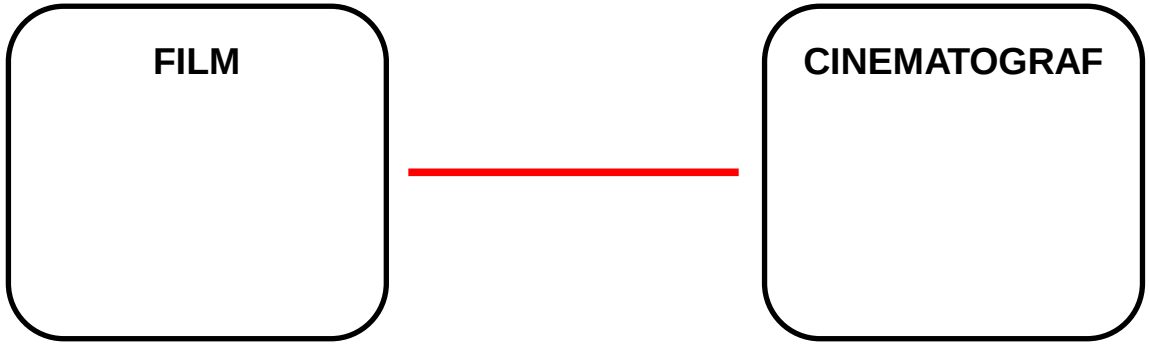
Relație:

Fiecare JUCĂTOR poate înscrie unul sau mai multe GOLURI.

Relații mail-mulți-la-mai-mulți (many-to-many)

- mai multor instanțe ale unei entități le corespund una sau mai multe instanțe ale altei entități
- se notează M:M
- o relație M:M poate apărea în prima etapă a proiectării unei baze da date, însă trebuie ulterior eliminată

☞ ***Exemplu:***



Relații:

Fiecare FILM rulează într-unul sau mai multe CINEMATOGRAFE.

Fiecare CINEMATOGRAF rulează unul sau mai multe FILME.

Convenții de reprezentare a relațiilor

În diagrama entități-relații, o relație între două entități este reprezentată printr-un *segment* (linie) care le unește.

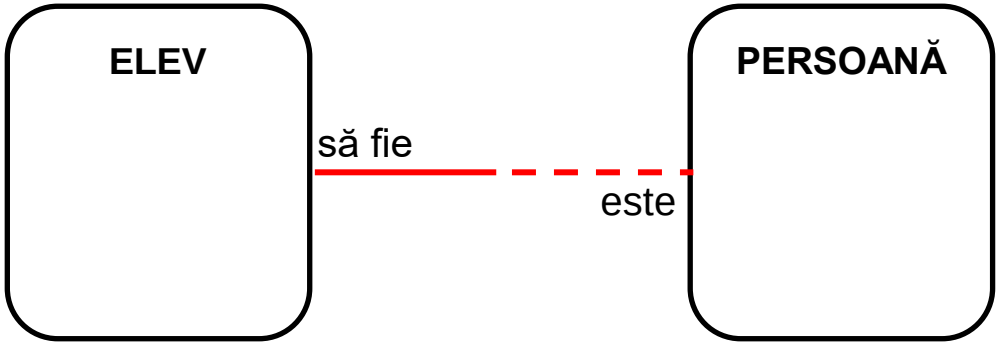
Numele relației se scrie în două moduri:

- deasupra liniei care desemnează relația pentru entitatea din stânga;
- sub linie pentru entitatea din dreapta.

Pentru a exprima opționalitatea unei relații, segmentul poate fi desenat:

- cu linie punctată (dacă relația este opțională);
- cu linie continuă (dacă relația este obligatorie).

☞ ***Exemplu:***



Relații:

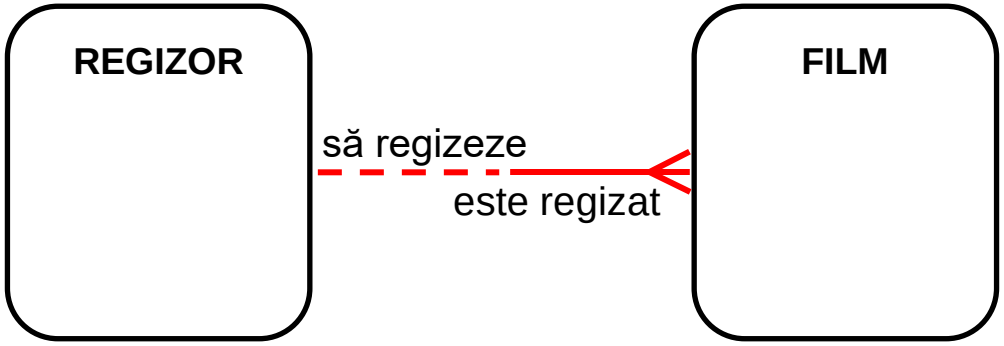
Fiecare ELEV trebuie să fie o PERSOANĂ și numai una.

Fiecare PERSOANĂ poate fi un ELEV și numai unul.

Pentru a exprima cardinalitatea unei relații, capătul segmentului poate fi desenat:

- cu linie simplă (o instanță a unei entități este conectată cu o instanță a altei entități);
- cu linie cu bifurcații (mai multe instanțe ale entității respective sunt în relație cu cealaltă entitate).

☞ ***Exemplu:***



Relații:

Fiecare REGIZOR poate să regizeze unul sau mai multe FILME.

Fiecare FILM trebuie să fie regizat de un REGIZOR și numai unul.

Citirea relațiilor

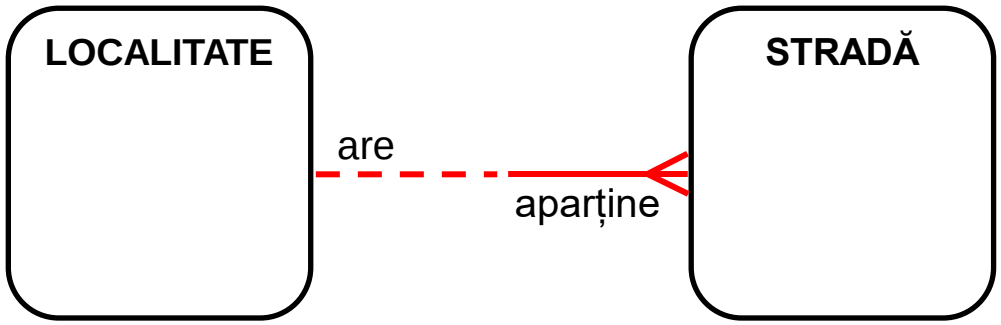
Citirea unei relații între două entități A și B se face astfel:

Fiecare [entitate A] [opționalitate] [nume relație][cardinalitate][entitate B].
Fiecare [entitate B] [opționalitate] [nume relație][cardinalitate][entitate A].

unde:

entitate A – numele primei entități (din stânga relației)
entitate B – numele celei de a doua entități (din dreapta relației)
opționalitate – cuvântul *poate* sau cuvântul *trebuie*
nume relație – verbul sau expresia verbală care identifică relația
cardinalitate – *unul și numai unul* sau *unul sau mai mulți*

☞ ***Exemplu:***



Fiecare [entitate A] [opționalitate] [nume relație][cardinalitate][entitate B].

Relația:

Fiecare LOCALITATE poate avea una sau mai multe STRĂZI.

Fiecare [entitate B] [opționalitate] [nume relație][cardinalitate][entitate A].

Relația:

Fiecare STRADĂ trebuie să aparțină unei LOCALITĂȚI și numai uneia.

✍ ***Exercițiu:***

Se dau entitățile ANGAJAT și DEPARTAMENT. Rezolvați următoarele sarcini:

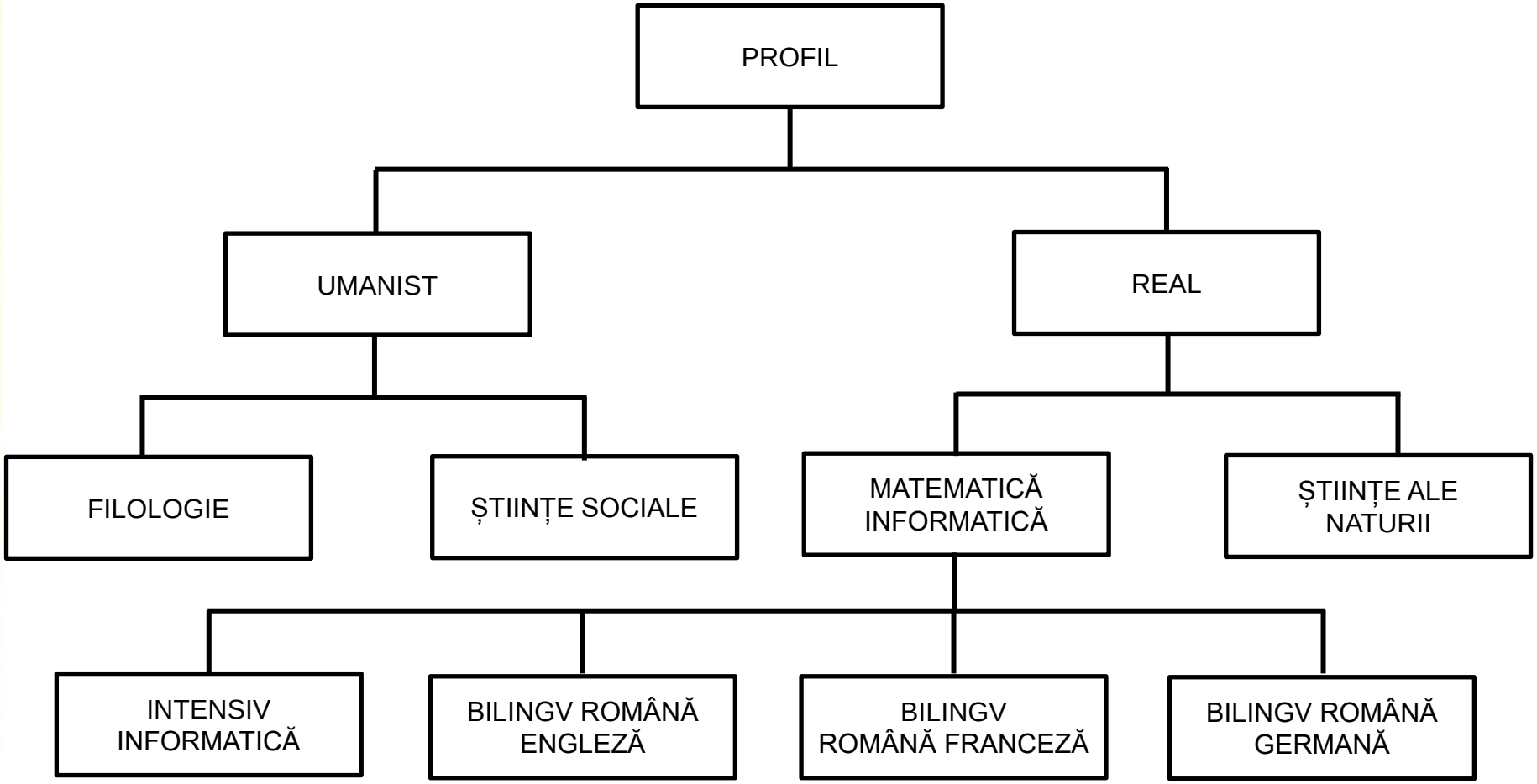
- desenați diagrama entități relații, reprezentând grafic entitățile date și relațiile dintre cele două entități;
- enumerați pentru fiecare entitate câteva atributele specifice;
- indicați simbolul care trebuie să preceadă fiecare atribut;
- scrieți relațiile dintre cele două entități și precizați tipul relațiilor.

Relații ierarhice și relații recursive

- în funcție de modul de subordonare, persoanele, instituțiile sau alte structuri pot fi ierarhizate
- folosirea relaţiilor ierarhice este recomandată în cazul în care se modelează anumite ierarhii ale instanţelor unei entităţi

☞ ***Exemplu:***

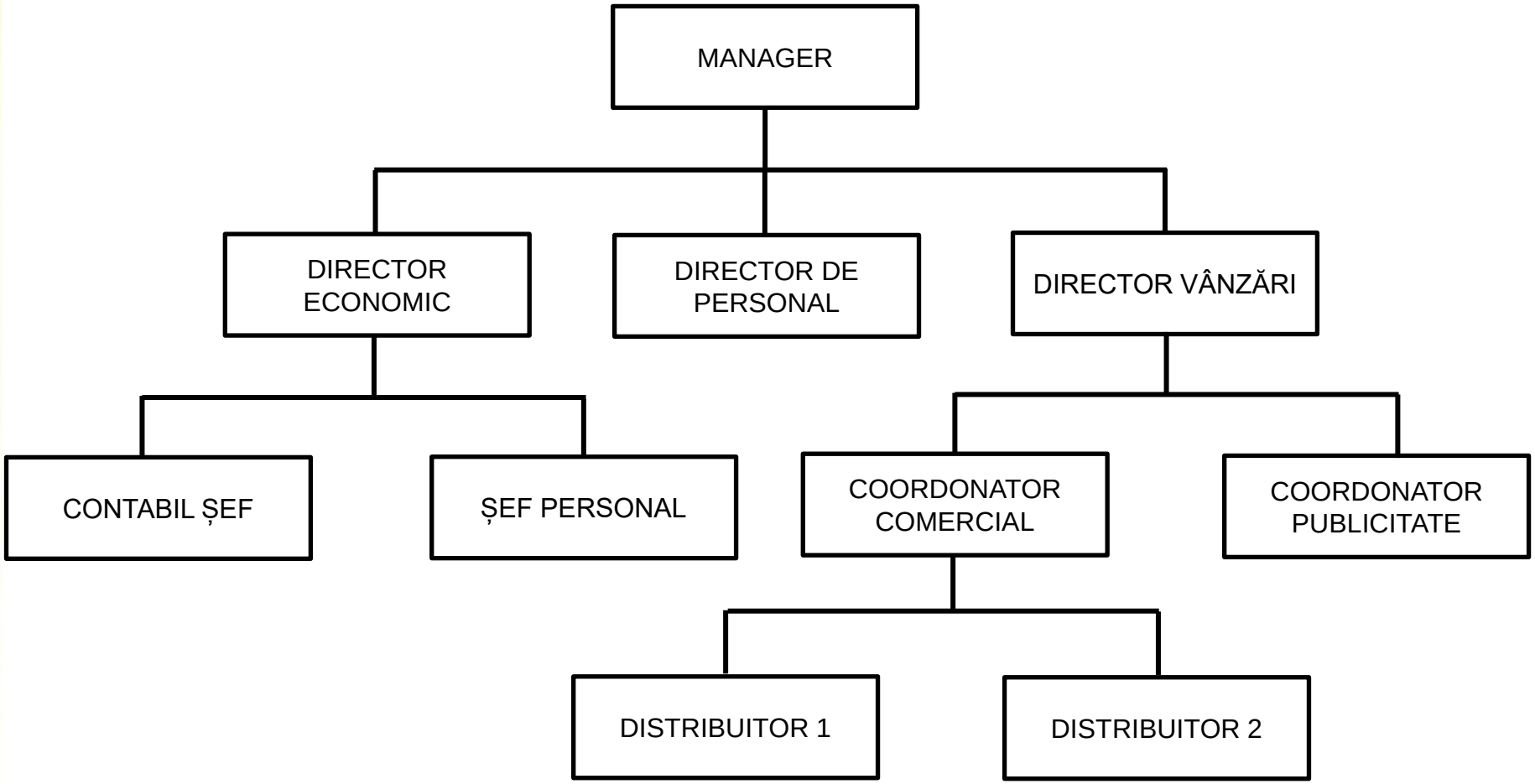
Structura ierarhică a claselor unui colegiu



- fiecare nivel din cadrul unei structuri ierarhice poate fi reprezentat în diagrama entități-relații cu ajutorului unei entități
- entitățile care formează o ierarhie și care au atribute comune pot fi modelate cu ajutorul unei singure entități
- se poate forma astfel o relație de la acea entitate la entitatea însăși
- o relație recursivă (relație în buclă) este o relație de la o entitate la ea însăși
- relația recursivă se stabilește pe o entitate care poate avea mai multe roluri

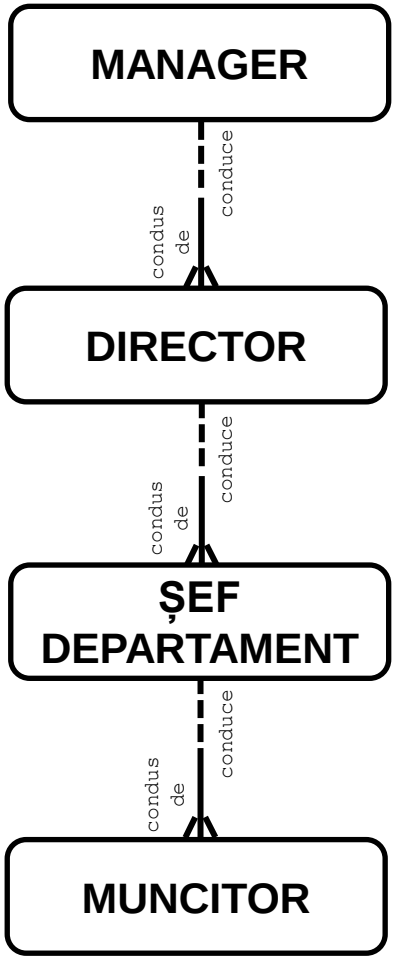
☞ ***Exemplu:***

Structura ierarhică a personalului dintr-o instituție publică



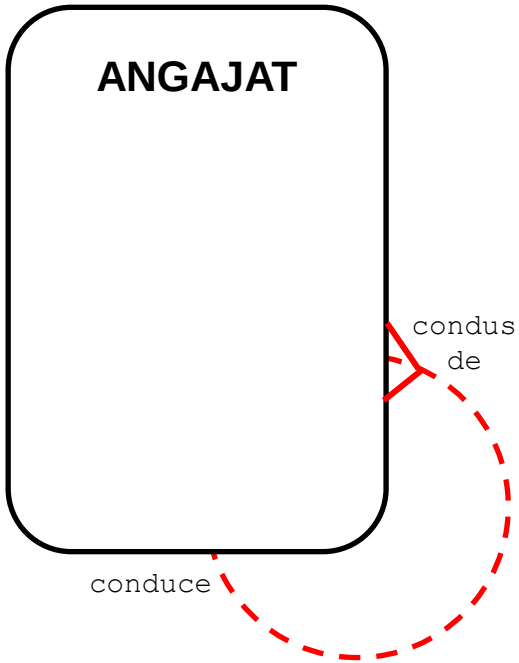
☞ ***Exemplu:***

Reprezentarea în ERD a structurii ierarhice a personalului



☞ ***Exemplu:***

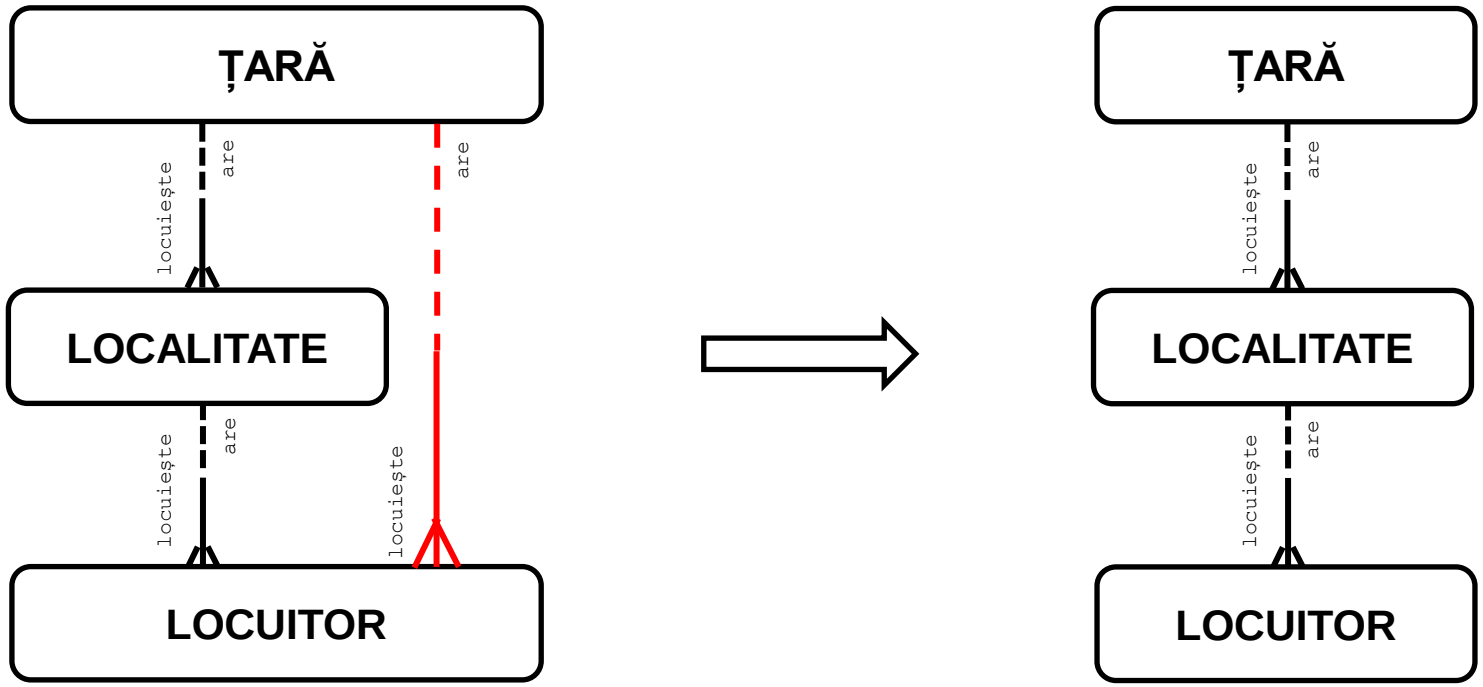
Modelarea ERD-ului folosind o singură entitate



Relații redundante

- o relație redundantă este o relație care poate fi dedusă din alte relații deja existente în baza de date
- la realizarea diagramei entități-relații trebuie evitate cazurile în care apar relații redundante

☞ ***Exemplu:***


ȚARĂ
locuiește
are
are
LOCALITATE
locuiește
are
locuiește
LOCUITOR
ȚARĂ
locuiește
are
LOCALITATE
locuiește
are
LOCUITOR

Relații exclusive / arce

- relațiile exclusive sunt relațiile care se pot exclude reciproc (dintr-un grup de relații existente între entități, doar una singură poate avea loc la un moment dat)
- un grup de relații exclusive este reprezentat în diagrama entități-relații printr-un arc peste relațiile care fac parte din grupul respectiv
- toate relațiile care fac parte din grupul de relații exclusive trebuie să aibă aceeași opționalitate
- un arc aparține unei singure entități (include doar relații care pleacă de la o aceeași entitate)
- o entitate poate avea mai multe arce, dar o relație nu poate face parte decât dintr-un singur arc

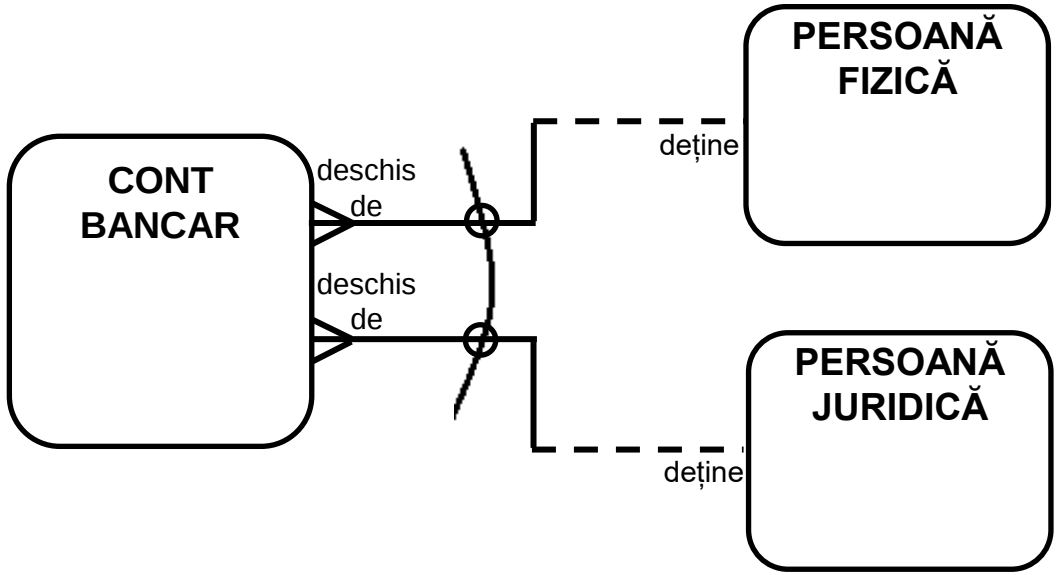
Tipuri de relații exclusive:

- relații exclusive obligatorii
- relații exclusive opționale

Relații exclusive obligatorii

- toate relațiile care fac parte din arcul respectiv sunt obligatorii
- de fiecare dată, una dintre relații are obligatoriu loc

☞ ***Exemplu:***



Relații:

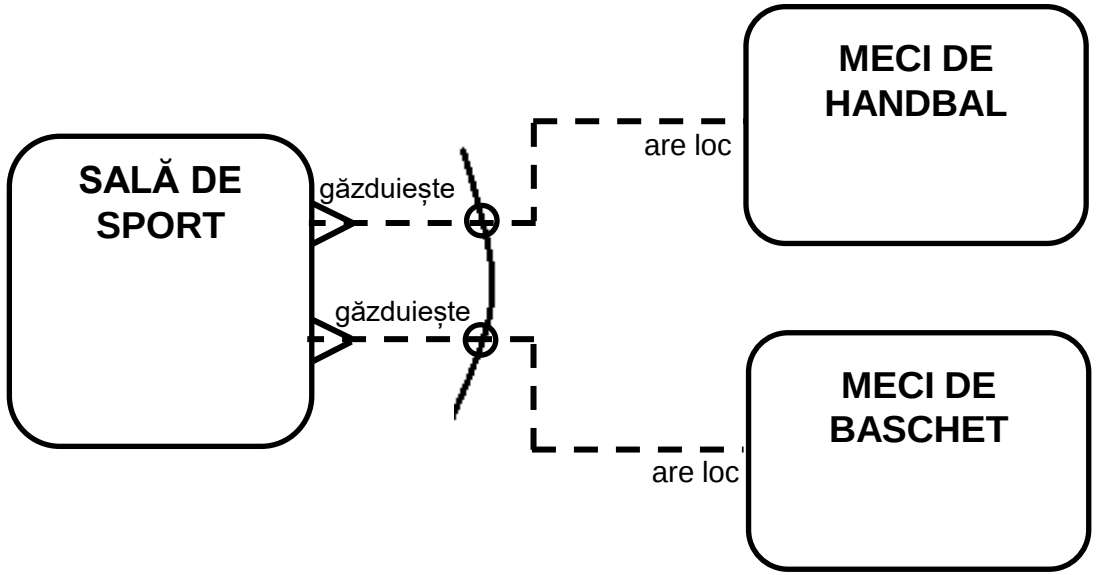
Fiecare CONT BANCAR trebuie să fie deschis de o PERSOANĂ FIZICĂ și numai una.

Fiecare CONT BANCAR trebuie să fie deschis de o PERSOANĂ JURIDICĂ și numai una.

Relații exclusive opționale

- toate relațiile care fac parte din arc sunt opționale
- de fiecare dată, cel mult una dintre relații are loc

☞ ***Exemplu:***



Relații:

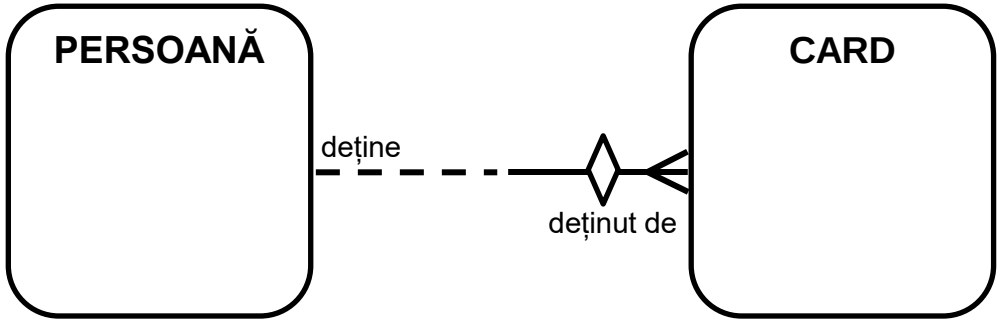
Fiecare SALĂ DE SPORT poate să găzduiască un MECI DE HANDBAL și numai unul.

Fiecare SALĂ DE SPORT poate să găzduiască un MECI DE BASCHET și numai unul.

Relații nontransferabile

- o relație este nontransferabilă dacă o asociere între două instanțe ale celor două entități, odată stabilită, nu mai poate fi modificată
- stabilirea relațiilor nontransferabile se face în funcție de regulile speciale ale afacerii modelate
- condiția de nontransferabilitate a unei relații se asigură în etapa de programare, restricția urmând să apară în documentație
- în diagrama entități-relații, o relație nontransferabilă se notează cu un romb pe linia corespunzătoare relației obigatorii

☞ ***Exemplu:***



Relații:

Fiecare PERSOANĂ poate deține unul sau mai multe CARDURI.

Fiecare CARD trebuie să fie deținut de o PERSOANĂ și numai una.

✍ ***Exercițiu:***

Dați câte un exemplu de ERD și scrieți relațiile obținute, pentru fiecare dintre următoarele tipuri de relații:

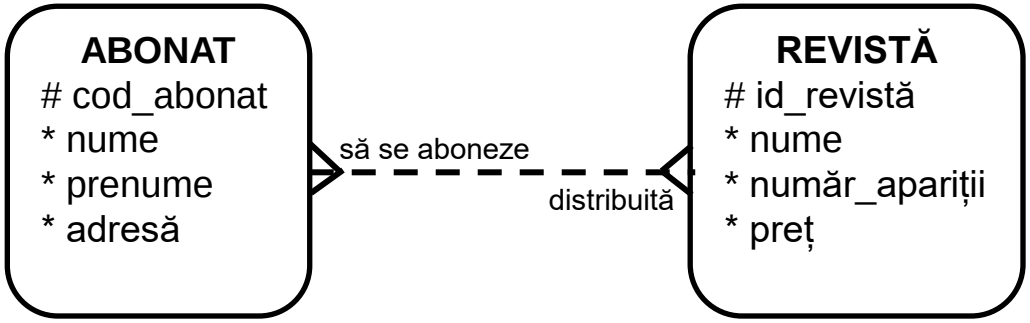
- relații recursive;
- relații redundante;
- relații nontransferabile.

Rezolvarea relaţiilor many-to-many

Relațiile de tipul *mai-mulți-la-mai-mulți* (M:M) pot să apară în diagrama entități-relații în prima etapă a proiectării bazei de date, însă aceste relații trebuie eliminate din ERD-ul final.

O relație *mai-mulți-la-mai-mulți* trebuie înlocuită cu două relații de tipul *unu-la-mai-mulți*.

☞ ***Exemplu:***



Relații:

Fiecare ABONAT poate să se aboneze la una sau mai multe REVISTE.

Fiecare REVISTĂ poate fi distribuită unuia sau mai multor ABONAȚI.

Observație:

Relația M:M nu specifică exact care instanță din prima entitate este în legătură cu care instanță din a doua entitate.

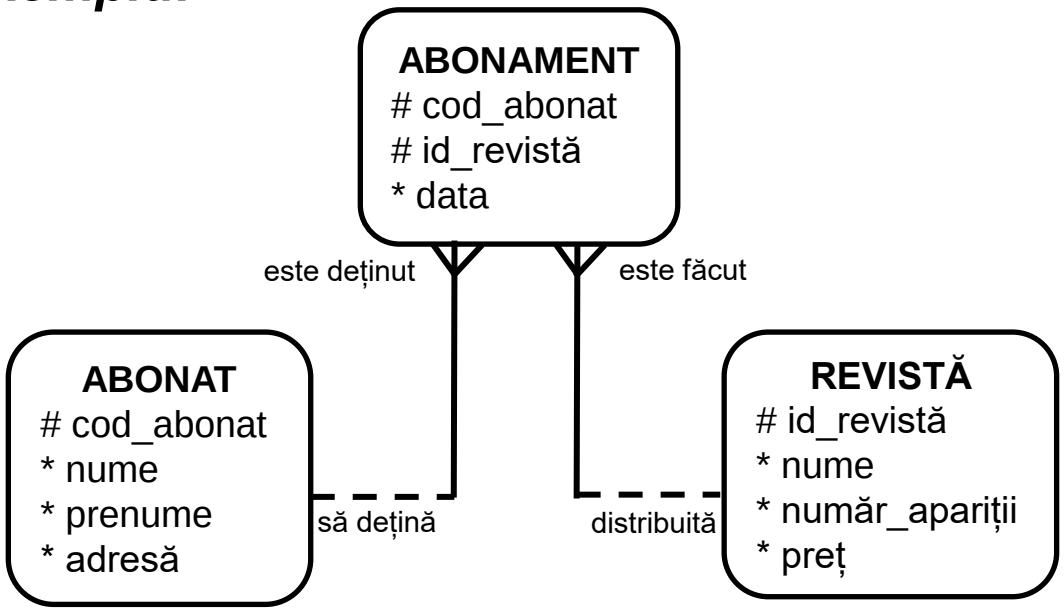
Rezolvarea unei relații *mai-mulți-la-mai-mulți* presupune introducerea între cele două entități, unite prin acest tip de relație, a unei noi entități numită *entitate de intersecție.*

Entitatea de intersecție va fi legată de cele două entități originale prin câte o relație de tipul *unu-la-mai-mulți.*

Metoda de rezolvare a unei relații *mai-mulți-la-mai-mulți*:

1. se identifică identificatorul unic pentru fiecare din cele două entități originale
2. se definește o nouă entitate (entitatea de intersecție), descrisă prin atributele UID din entitățile originale și eventual alte atribute proprii
3. se stabilesc două relații de tip *unu-la-mai-mulți*, între entitățile originale și entitatea de intersecție

☞ ***Exemplu:***

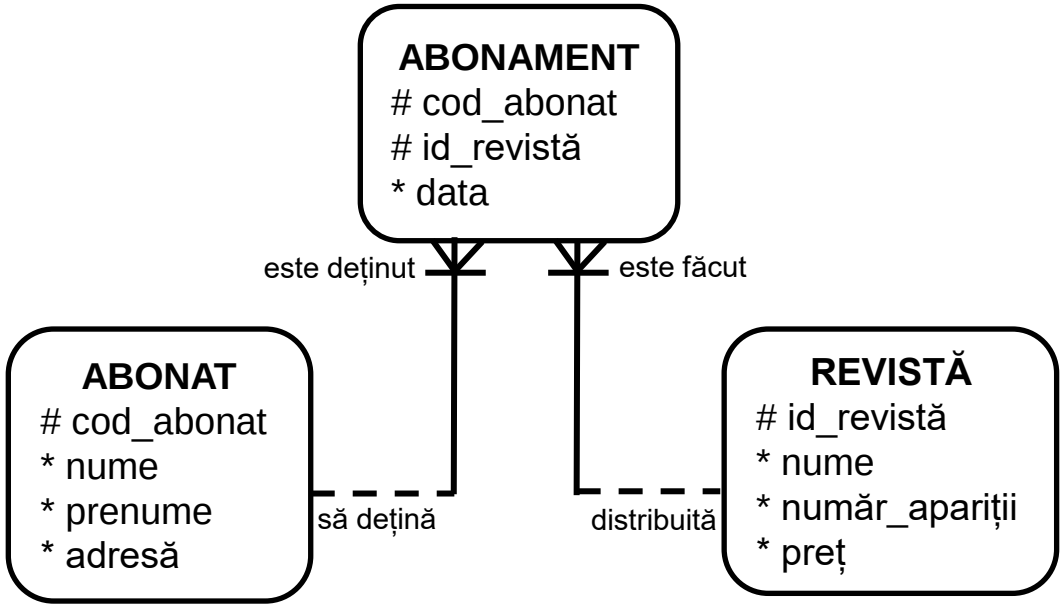


Observații:

- relațiile introduse sunt de tip 1:M
- relațiile care pleacă din entitatea de intersecție vor fi întotdeauna obligatorii în această parte
- partea cu bifurcație a relațiilor introduse va fi întotdeauna înspre entitatea de intersecție

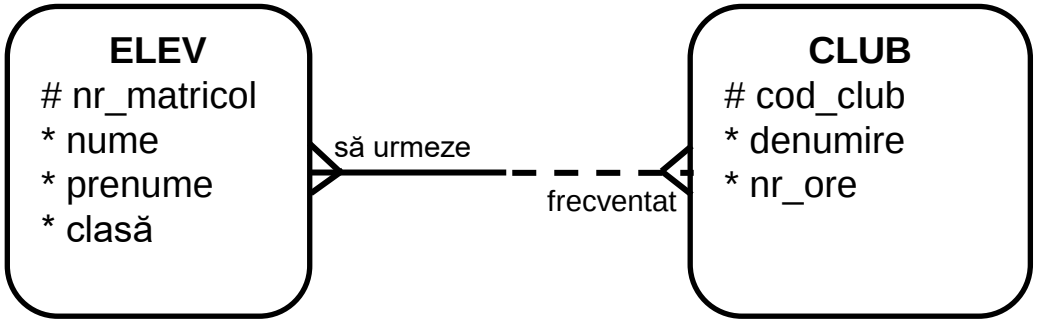
Pentru a indica faptul că a fost introdusă o entitate de intersecție și că identificatorul unic al acesteia preia identificatorul unic din altă entitate cu care este legată, relația este barată înspre entitatea de intersecție.

☞ ***Exemplu:***



✍ ***Exercițiu:***

Modificați ERD-ul de mai jos, eliminând relația M:M dintre entități.



Normalizarea datelor

Modelul conceptual al unei baze de date trebuie optimizat pentru a elimina anumite aspecte nedorite, astfel încât să nu mai apară erori în procesul de utilizare a bazei de date (selecţie date, actualizare datelor, etc.).

Aspectele nedorite țin de anomalii și inconsistențe ale datelor:

- aceleași informații să se regăsească în locuri diferite ale bazei de date (redundanța datelor);
- memorarea în baza de date a unor informații care pot fi deduse din alte informații deja existente în baza de date.

Procesul de optimizare a modelului conceptual al unei baze de date se numește ***normalizare***.

Normalizarea:

- este o tehnică de proiectare a bazelor de date prin care se elimină anumite anomalii şi inconsistenţe a datelor
- implică descompunerea unei entități în două sau mai multe entități, prin compunerea cărora se obțin aceleași informații ca și în entitatea inițială
- este ultima etapă în procesul de proiectare a unei baze de date și precede etapa de implementare a acesteia.

Conceptul de normalizare a unei baze de date a fost inițiat de Edgar Frank Codd.

Procesul de normalizare se realizează în mai multe etape, numite ***forme de normalizare (forme normale)*.**

O ***formă normală*** presupune anumite condiții sau reguli de corectitudine pe care trebuie să le îndeplinească valorile.

Dacă o bază de date nu este normalizată, ea nu poate fi utilizată cu maximă eficiență.

E.F. Codd a definit primele trei forme normale 1NF, 2NF şi 3NF. Ulterior s-au mai definit formele normale 4NF, 5NF, 6NF care însă sunt rar folosite în proiectarea bazelor de date. Raymond Boyce a introdus, împreună cu E.F. Codd forma normală Boyce-Codd (BCNF), ca o definiție mai completă a celei de a treia forme normale.

Este foarte importantă cunoașterea și utilizarea primei forme normale, celelalte de multe ori fiind opționale. Se recomandă însă normalizarea până la forma a treia formă normală.

Prima formă normală (1NF - First Normal Form)

O entitate se află în prima formă normală dacă:

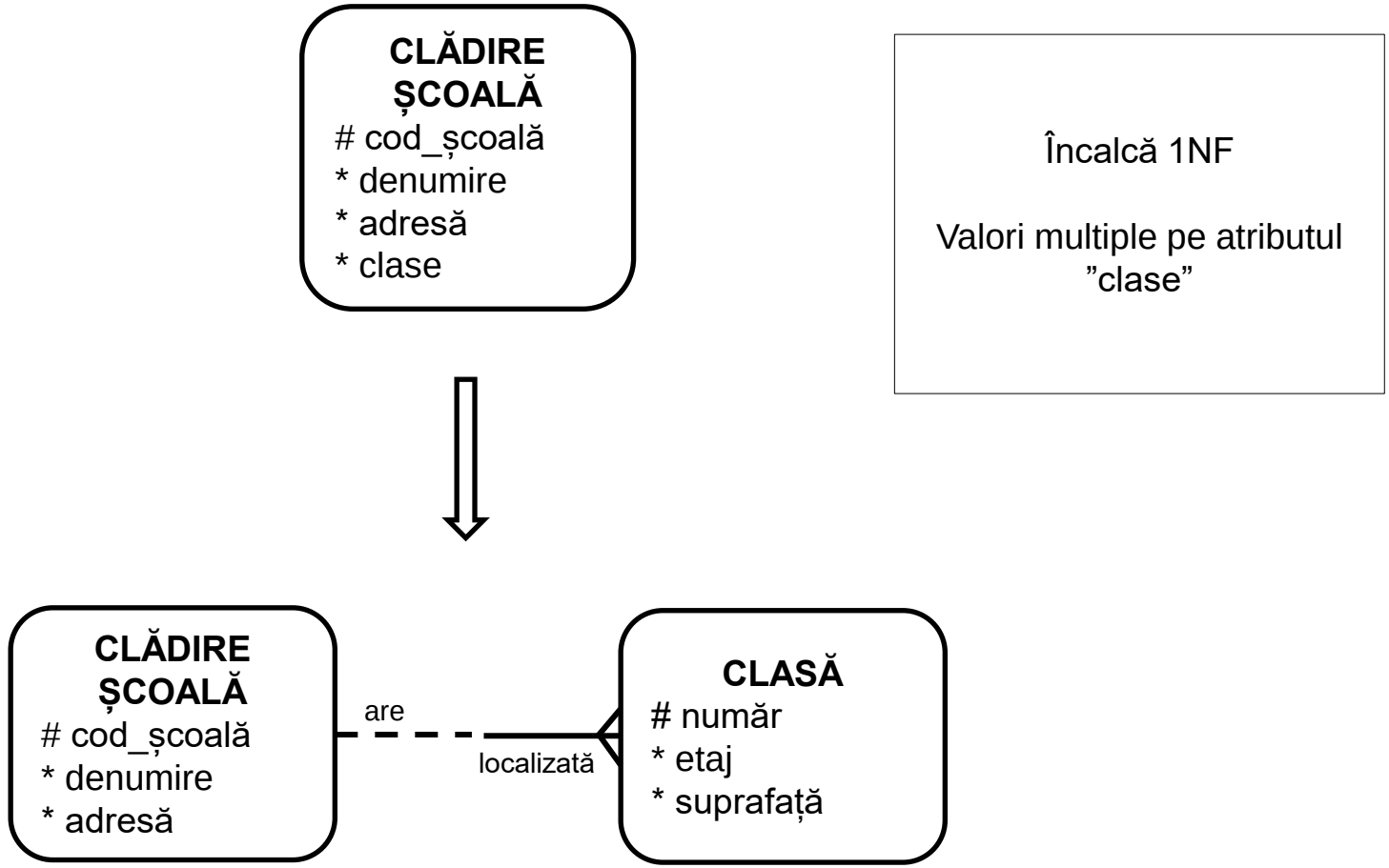
- nu există atribute cu valori multiple;
- nu există atribute sau grupuri de atribute care se repetă.

Prima formă normală prevede ca fiecare instanță să dețină o valoare atomică pentru fiecare atribut al entității și să nu existe grupe de date repetitive.

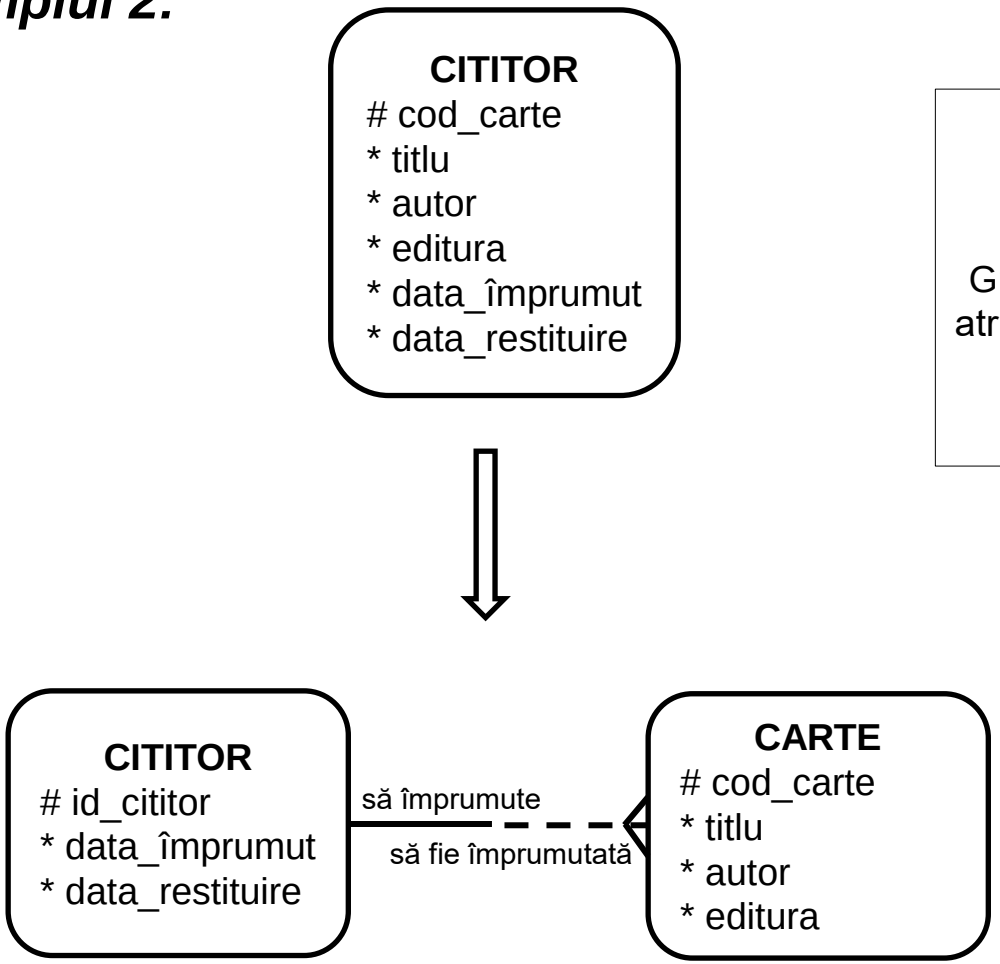
O valoare este atomică dacă este elementară, indivizibilă (nu poate fi descompusă) și dacă este folosită ca un întreg, nu porțiuni ale sale.

Eliminarea valorilor multiple sau a grupurilor repetitive se face prin adăugarea unei noi entități care preia unele dintre atributele entității inițiale. Această nouă entitate va fi relaționată de entitatea originală printr-o relație 1:M.

☞ ***Exemplul 1:***


CLĂDIRE ȘCOALĂ
cod_școală
* denumire
* adresă
* clase
Încalcă 1NF
Valori multiple pe atributul ”clase”
CLĂDIRE ȘCOALĂ
cod_școală
* denumire
* adresă
are
localizată
CLASĂ
număr
* etaj
* suprafață

☞ ***Exemplul 2:***


CITITOR
cod_carte
* titlu
* autor
* editura
* data_împrumut
* data_restituire
Încalcă 1NF
Grup de valori repetitive pe atributele ”data_împrumut” și ”data_restituire”.
CITITOR
id_cititor
* data_împrumut
* data_restituire
să împrumute
să fie împrumutată
CARTE
cod_carte
* titlu
* autor
* editura

A doua formă normală (2NF – Second Normal Form)

O entitate se află în a doua formă normală dacă:

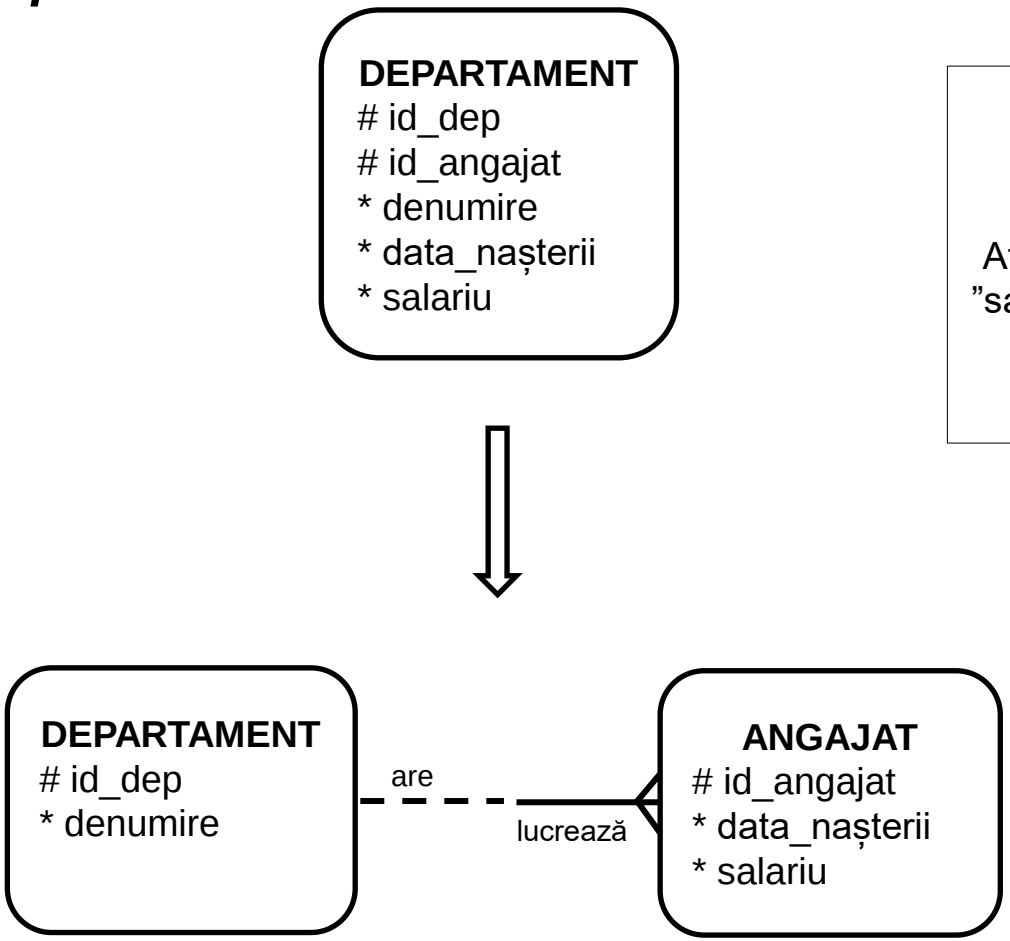
- se află în 1NF;
- orice atribut care nu este identificator unic (UID) depinde de întreg UID-ul nu numai de o parte a acestuia.

A doua formă normală prevede ca orice atribut care nu este identificator unic al entității să fie dependent de întregul identificator unic.

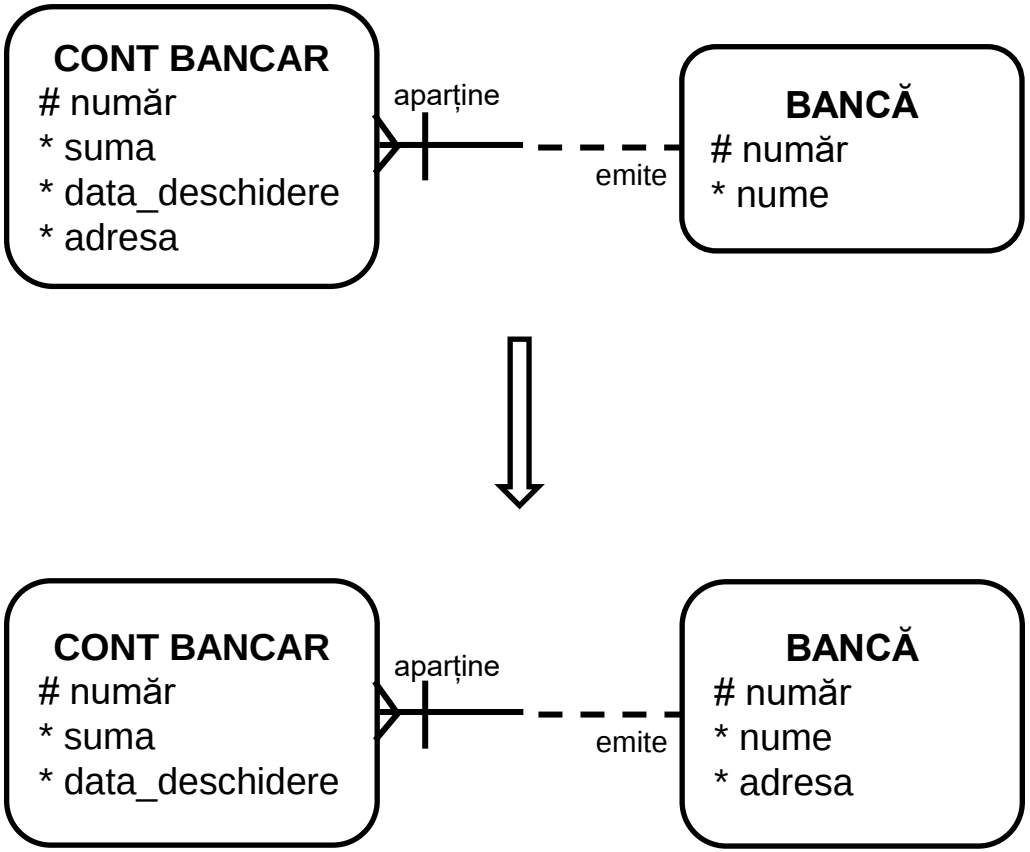
Această situație se pune doar în cazul în care identificatorul unic este compus (este format din mai multe atribute) sau este o combinație de atribut și o relație (relație barată).

Toate atributele unei entități sunt dependente funcţional de UID, cu excepţia atributelor care o formează. Normalizarea în a doua formă se face prin identificarea şi eliminarea tuturor dependenţelor funcţionale parțiale.

☞ ***Exemplul 1:***


DEPARTAMENT
id_dep
id_angajat
* denumire
* data_nașterii
* salariu
Încalcă 2NF
Atributele ”data_nașterii” și ”salariu” nu depind de întreg UID-ul.
DEPARTAMENT
id_dep
* denumire
are
lucrează
ANGAJAT
id_angajat
* data_nașterii
* salariu

☞ *Exemplul 2:*



Încalcă 2NF

UID-ul entității CONT BANCAR este compus, iar atributul ”adresa” depinde parțial funcțional de acesta.

A treia formă normală (3NF – Third Normal Form)

O entitate se află în a treia formă normală dacă:

- se află în 2NF;
- niciun atribut care nu este parte a identificatorului unic (UID) nu depinde de un alt atribut care nu este UID.

A doua formă normală prevede ca orice atribut care nu este identificator unic al entității să nu fie dependent de un alt atribut care nu face parte din identificator unic.

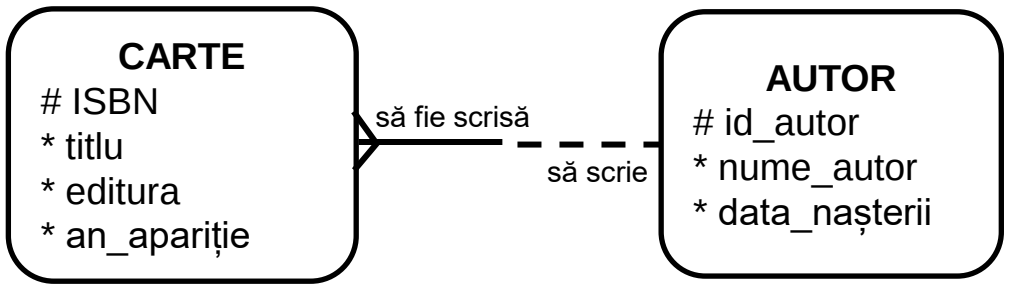
Toate atributele care formează entitatea trebuie să depindă direct de identificatorul unic al acesteia.

Pentru ca o entitate să fie în 3NF trebuie făcută o împărțire a entității în două entități, fiecare preluând atributele specifice astfel încât toate atributele să depindă numai de identificatorul unic.

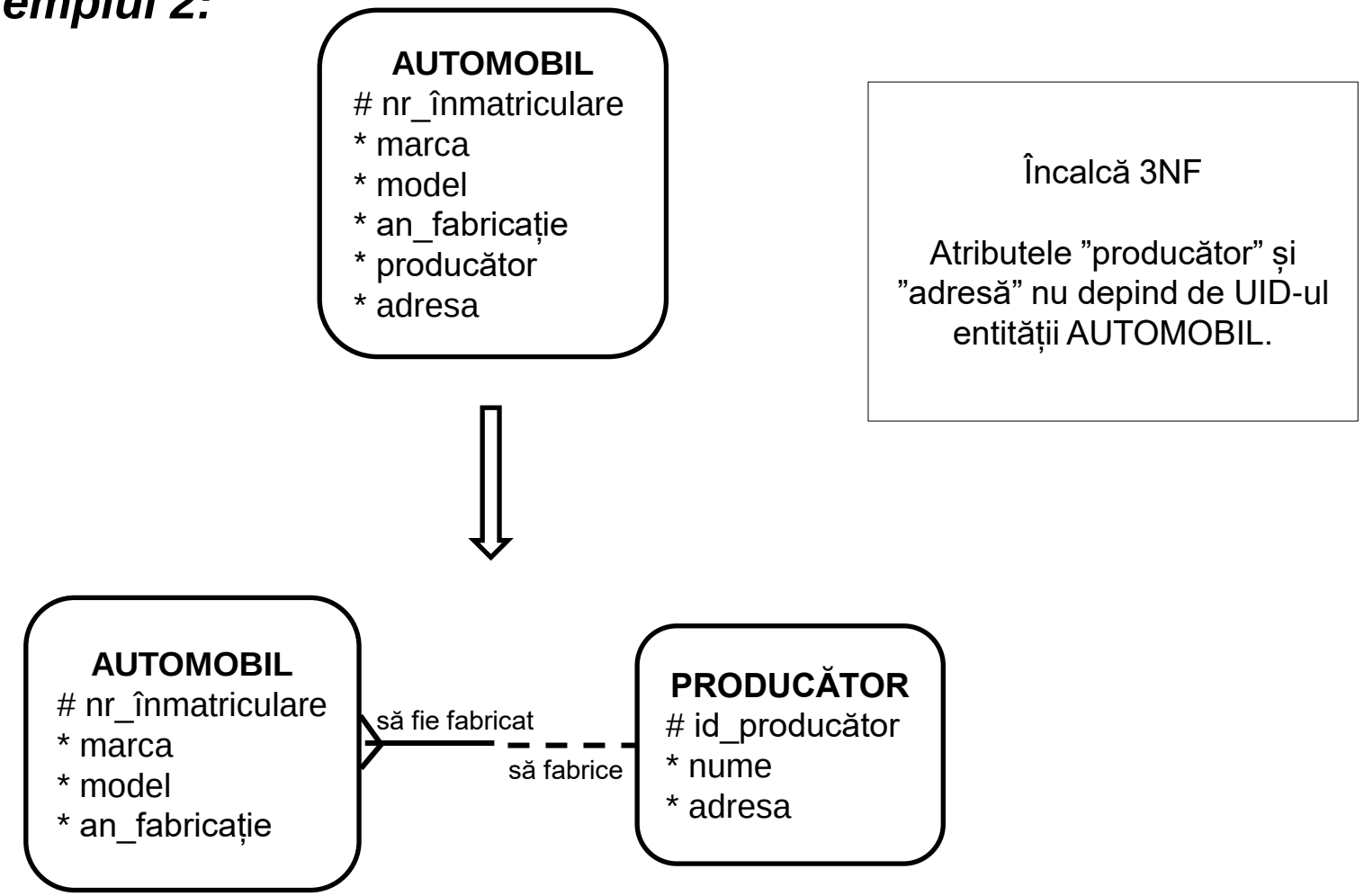
☞ ***Exemplul 1:***

Încalcă 3NF

Atributele ”nume_autor” și ”data_nașterii” nu depind de UID-ul ISBN.



☞ ***Exemplul 2:***


AUTOMOBIL
nr_înmatriculare
* marca
* model
* an_fabricație
* producător
* adresa
Încalcă 3NF
Atributele ”producător” și ”adresă” nu depind de UID-ul entității AUTOMOBIL.
AUTOMOBIL
nr_înmatriculare
* marca
* model
* an_fabricație
să fie fabricat
să fabrice
PRODUCĂTOR
id_producător
* nume
* adresa

Tabele

- Crearea structurii tabelelor
- Conținutul unui tabel
- Operații specifice prelucrării datelor

Crearea structurii tabelelor

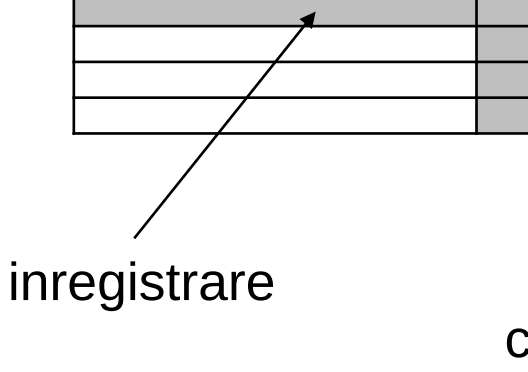
Crearea unei tabele constă în stabilirea:

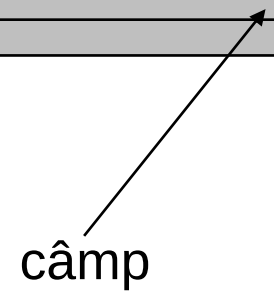
- numelui tabelei;
- coloanelor care compun tabela;
- tipurilor de date pe care le au coloanele tabelei;
- restricțiilor (constrângerilor) care asigură integritatea și coerența informațiilor din baza de date.

Tabelă = o structură utilizată pentru stocarea și organizarea datelor

- tabelele sunt formate din linii (rânduri) și coloane
- liniile unei tabele se numesc înregistrări
- coloanele unei tabele se numesc câmpuri

Nume_coloană 1	Nume_coloană 2	Nume_coloană 3





Coloanele unei tabele reţin date de un anumit tip. Fiecare coloană a unei tabele va memora date de acelaşi tip.

Tip de dată = o mulţime de valori predefinită în sistem sau definită de utilizator

În funcţie de aplicaţia utilizată pentru implementarea bazelor de date, pot fi acceptate diferite tipuri de date.

Tipuri de date Oracle:

- number
- varchar2
- char
- date
- long
- row
- row long
- bfile

1. Tipul NUMBER

- utilizat pentru memorarea numerelor în virgulă fixă și virgulă mobilă

Sintaxa:

nume_câmp NUMBER (precizie, scala)

unde: ”precizie” reprezintă numărul total de cifre ale numărului
”scala” reprezintă numărul de zecimale

☞ ***Exemple:***

preț NUMBER
preț NUMBER (3)
preț NUMBER (3,2)

2. Tipul VARCHAR2

- utilizat pentru definirea datelor șir de caractere de lungime variabilă

Sintaxa:

nume_câmp VARCHAR2(număr_octeți)

- lungimea șirului de caractere este cuprinsă în intervalul 1 – 4000 octeți
- pentru un șir mai scurt decât numărul de octeți setați, șirul nu se completează cu spații

☞ ***Exemplu:***

autor VARCHAR2(30)

3. Tipul CHAR

- utilizat pentru definirea datelor șir de caractere de lungime fixă

Sintaxa:

nume_câmp CHAR(număr octeți)

- pentru un șir mai scurt decât numărul de octeți setați, se adaugă spații la sfârșitul șirului până la atingerea lungimii specificate la definire

☞ ***Exemplu:***

titlu CHAR(50)

4. Tipul DATE

- utilizat pentru definirea datelor de tip dată calendaristică

Sintaxa:

nume_câmp DATE

- formatul intern:
  - pentru dată: DD-Mon-RR
    (exp. 15-Sep-10)
  - pentru oră: HH:MM:SS
    (exp. 11:05:40)

☞ ***Exemplu:***

data_apariției DATE

Alte tipuri de date:

- ***tipul LONG*** (utilizat pentru definirea datelor șir de caractere cu lungime variabilă de cel mult 2GB)
- ***tipul ROW*** (utilizat pentru stocarea datelor binare sau șiruri de octeți, similar cu tipul VARCHAR2)
- ***tipul BFILE*** (utilizat pentru stocarea obiectelor binare cu dimensiune mare de până la 4GB, în afara bazei de date, în fișierele sistemului de operare)

Conținutul unei tabele

O tabelă poate fi văzută ca un tablou bidimensional dispus pe linii și coloane.

☞ ***Exemplu:***

Tabela ELEVI

Număr matricol	Nume	Prenume	Data nașterii	Media	Telefon

Tabela este implementarea practică a unei entități din modelul conceptual care are rolul de a memora informații omogene relative la o entitate.

Atributele unei entități reprezintă coloane într-o tabelă, iar instanțele entității reprezintă liniile tabelei.

entitate	⟶	***tabelă***
atribut	⟶	***coloană/câmp***
instanță	⟶	***înregistrare***

☞ ***Exemplu:***

Entitatea CARTE

Tabela CĂRȚI

ISBN	Titlu	Editura	An apariție	Nume autor	Data nașterii

Câmpurile se caracterizează prin:

- nume (identificatorul coloanei)
- domeniul de valori (mulțimea valorilor acceptate)

Domeniul de valori este determinat de tipul atributului și de eventualele restricții stabilite în funcție de caracteristicile modelului conceptual modelat.

Dacă o instanță a tabelei nu are valori pentru toate câmpurile, acele câmpuri vor memora valoarea specială NULL.

NULL este o metavaloare care indică faptul că valoarea este necunoscută, neatribuită sau lipsește.

Această metavaloare este diferită de 0 (zero), spațiu sau șirul vid.

☞ ***Exemplu:***

Entitatea ANGAJAT

CARTE
id_angajat
* nume
* prenume
* data_angajării
* salariul
○ telefon

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Data angajării	Salariul	Telefon
106	Pop	Ana	10 Feb 02	2800	014 228
289	Avram	Ion	01 Apr 10	2500	
149	David	Mara	15 Ian 90	3000	

NULL

Operații specifice prelucrării tabelelor

După ce a fost definită structura fiecărei tabele ce formează baza de date, se trece la etapa de introducere propriu-zisă a datelor în tabele, încheind astfel operația de creare a bazei de date.

Crearea unei tabele se realizează în două etape:

- stabilirea structurii tabelei;
- introducerea datelor în tabelă.

Operații specifice prelucrării tabelelor:

- actualizarea datelor din tabele
- validarea datelor
- vizualizarea conținutului tabelei bazei de date
- sortarea tabelelor de date

Actualizarea datelor din tabele

- adăugarea/introducerea de noi înregistrări
- ștergerea unei înregistrări
- modificarea valorii câmpurilor dintr-o înregistrare

Validarea datelor

Orice bază de date trebuie să respecte regulile de integritate care să garanteze că datele introduse în baza da date sunt corecte și valide.

Regulile unei afaceri sunt de două tipuri:

- reguli structurale;
- reguli procedurale.

Regulile structurale indică ce tipuri de informații sunt memorate într-o tabelă și cum sunt relaționate elementele informaționale. Aceste reguli pot fi reprezentate întotdeauna în ERD.

Regulile procedurale descriu procesul afacerii. Aceste reguli nu pot fi reprezentate în ERD și trebuie să fie cuprinse în documentație pentru a fi implementate cu ajutorul operațiilor de validare.

Vizualizarea conținutului tabelei bazei de date

- integral, pentru toate înregistrările
- selectiv sau prin interogare, pentru înregistrările care respectă anumite proprietăți

Sortarea tabelelor de date

- afișarea informațiilor din tabele după un criteriu de ordine crescător sau descrescător

Baze de date

- Modele de baze de date
- Relaţionare, cheie primară, chei externe
- Reguli de integritate
- Programe de validare, de acţiune
- Operaţii specifice prelucrării bazelor de date (interogări, rapoarte)

Modele de baze de date

Bazele de date permit stocarea volumelor mari de date între care pot fi stabilite anumite relații.

Descrierea datelor dintr-o bază de date și a relațiilor dintre acestea se face cu ajutorul unei scheme a bazei de date (a unui model de bază de date).

Un ***model*** reprezintă o structură (abstractizare) ce simbolizează toate entitățile specifice unui scenariu de afacere.

Un ***model de dată*** reprezintă o colecție de concepte necesare descrierii datelor, a relațiilor dintre ele, precum și o serie de constrângeri aplicate datelor.

Un model de date descrie:

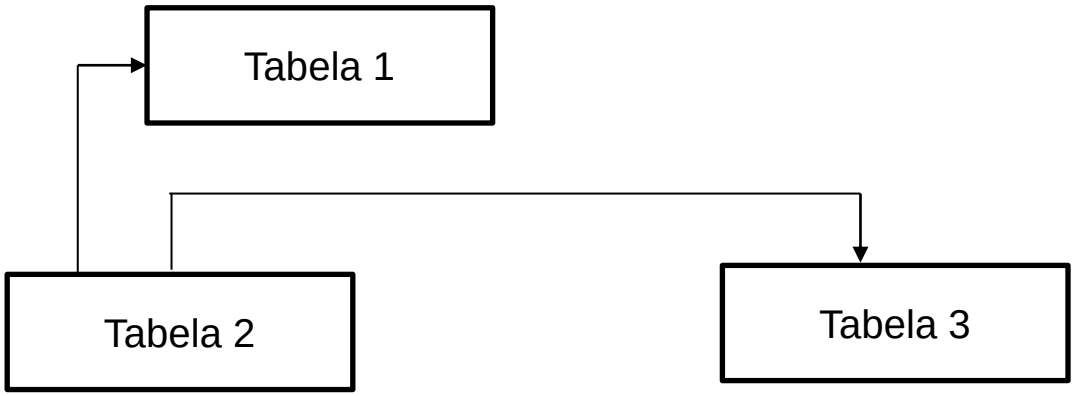
- schema bazei de date;
- structura datelor;
- legăturile dintre date;
- constrângerile impuse.

Modelele de date utilizate pentru gestionarea bazelor de date sunt:

- modelul relațional
- modelul ierarhic
- model rețea

1. Modelul relațional

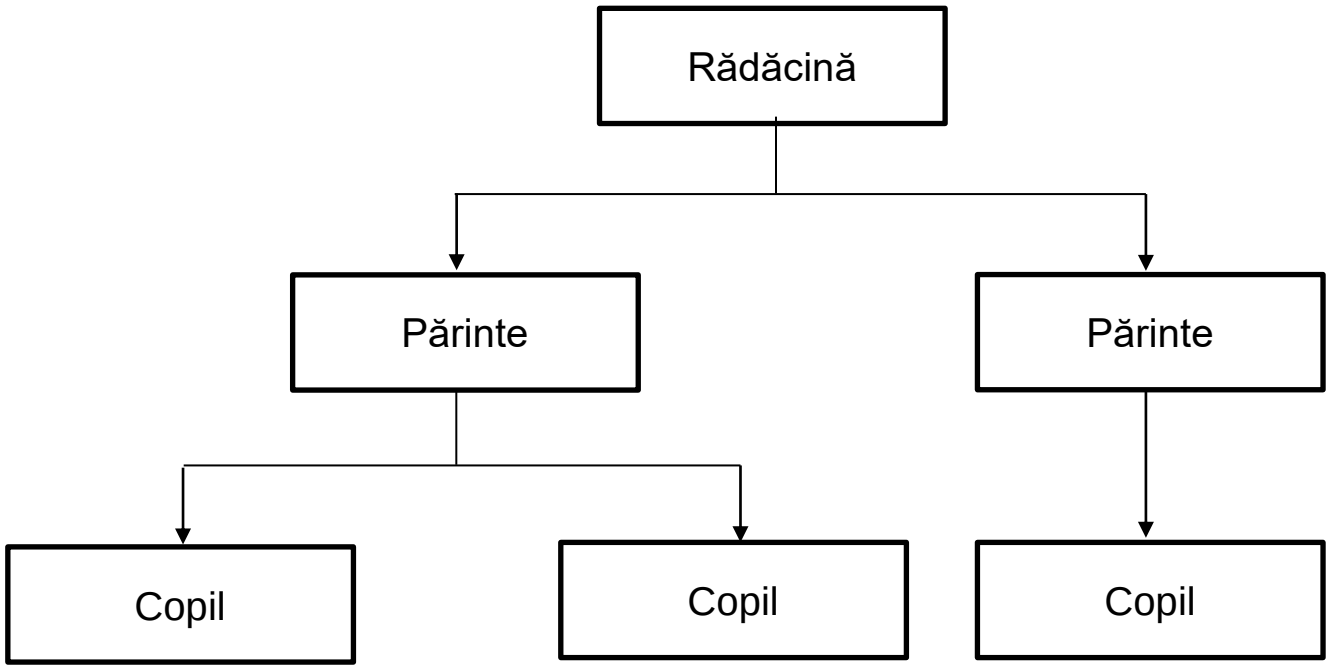
- este principalul model de date, fiind propus de E.F. Codd în anul 1970
- are la bază conceptul matematic de ***relație***, reprezentată fizic sub forma unei ***tabele***
- baza de date este reprezentată de un grup de tabele corelate
- datele sunt structurate logic sub formă de tabele formate din linii și coloane
- liniile reprezintă înregistrări individuale (tuple)
- coloanele reprezintă atribute (câmpuri)



Modelul relațional al bazei de date

2. Modelul ierarhic

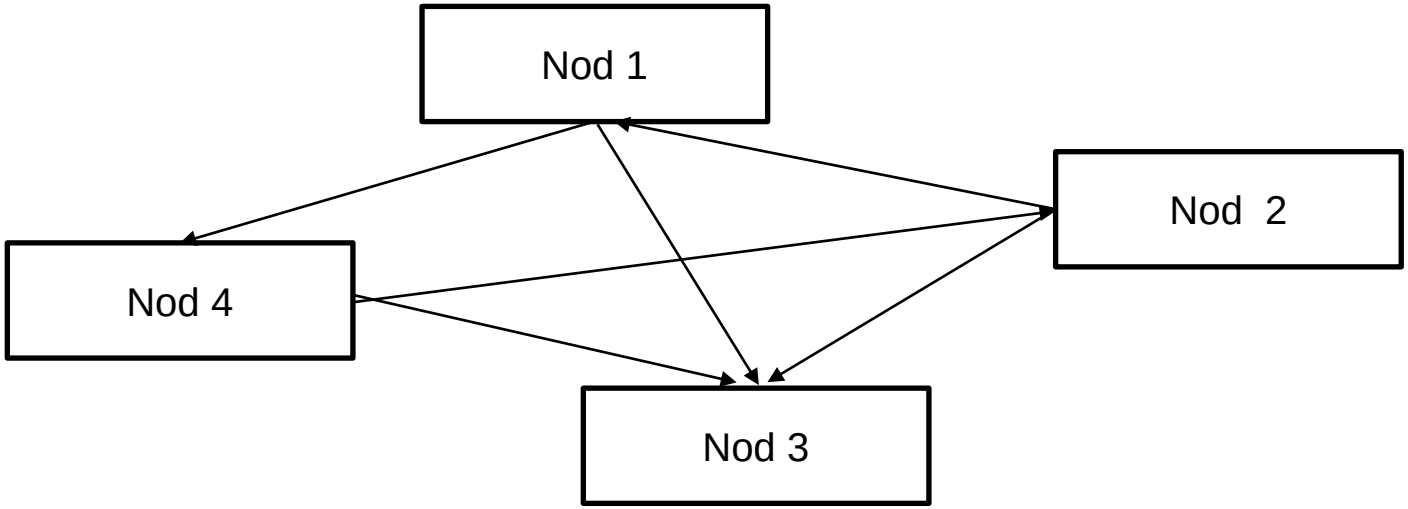
- este primul model de date, fiind dezvoltat de firma IBM
- datele sunt organizate sub forma unor structuri arborescente formate din noduri și arce
- nodurile structurii arborescente reprezintă clase de obiecte
- arcele structurii reprezintă legăturile dintre clasele de obiecte
- există o rădăcină cu mai mulți descendenți, care poate avea, la rândul lor, alți descendenți



Modelul ierarhic al bazei de date

3. Modelul rețea

- este creat ca o încercare de a rezolva unele dintre problemele modelului ierarhic
- este format dintr-o colecție de noduri și seturi (legături)
- un nod reprezintă o colecție de înregistrări
- legăturile reprezintă relațiile din cadrul bazei de date
- o bază de date de tip rețea poate fi reprezentată grafic sub forma unui graf orientat
- vârfurile grafului reprezintă entitățile (nodurile)
- relațiile suntr reprezentate prin săgețile dintre vârfuri



Modelul rețea al bazei de date

Alte modelele de date utilizate pentru gestionarea bazelor de date:

- modelul tabelar

  (toate datele suntr reprezentate sub forma unui singur tabel)
- modelul obiectual

  (suportă modele de obiecte comlexe)
- model hibrid

  (reprezintă o îmbinare de alte modele)

Relaţionare, cheie primară, chei externe

Modelul relațional se bazează pe conceptul matematic de ***relație***, prezentat fizic sub formă de ***tabelă***.

Crearea tabelelor bazei de date reprezintă prima etapă a procesului de transformare din model conceptual în model fizic.

Procesul de transformare a modelului conceptual (diagrama entități-relații) în model fizic (model relațional) se numește ***mapare***.

O ***bază de date relaţională*** reprezintă un ansamblu de relaţii (tabele), prin care se reprezintă datele şi legăturile dintre ele.

Conceptele specifice modelului relațional sunt:

- relație (tabelă care conține date)
- atribut (coloană a tabelei)
- domeniu (mulțimea de valori permise pentru un atribut)
- tuplu (rând în tabelă)
- schema relației (denumirea relației, set de perechi de atribute și denumiri de domenii)

Mapare

Model conceptual	Model fizic
entitate	tabelă
atribut	câmp
instanță	înregistrare
identificator unic	cheie primară
relație	cheie străină
regulile afacerii	constrângeri

Cheie primară

- conform teoriei relaţionale, o tabelă nu poate să conţină două sau mai multe rânduri (tupluri) identice
- fiecare rând al unei tabele trebuie să poată fi identificat într-o manieră clară, prin intermediul unui singur atribut sau a unui grup de atribute ce aparţin tabelei
- o ***cheie primară*** este un atribut sau set de atribute pentru identificarea unică a înregistrărilor într-o tabelă a bazei de date
- o cheie primară formată din mai multe atribute se numește ***cheie compusă***
- cheia primară trebuie să conțină valori unice și nu admite valori nule, iar în cazul cheilor compuse, nici un atribut parte din cheie nu poate avea valori nule

☞ ***Exemplu:***

Entitatea CARTE

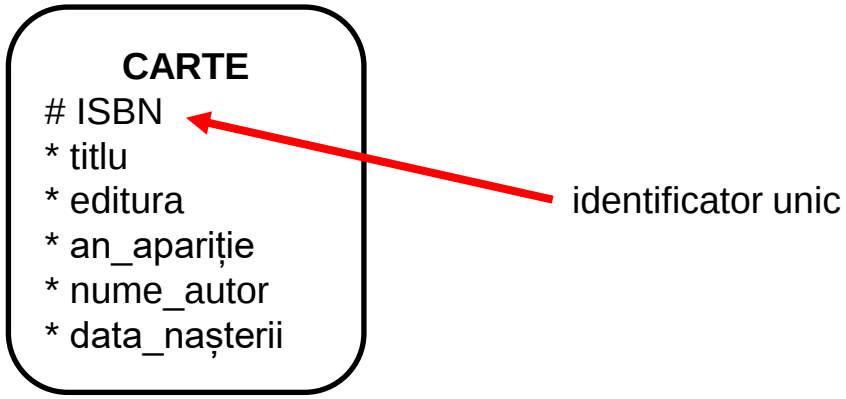


Tabela CĂRȚI

ISBN	Titlu	Editura	An apariție	Nume autor	Data nașterii

cheie primară

☞ *Exemplu:*

Entitatea CARTE

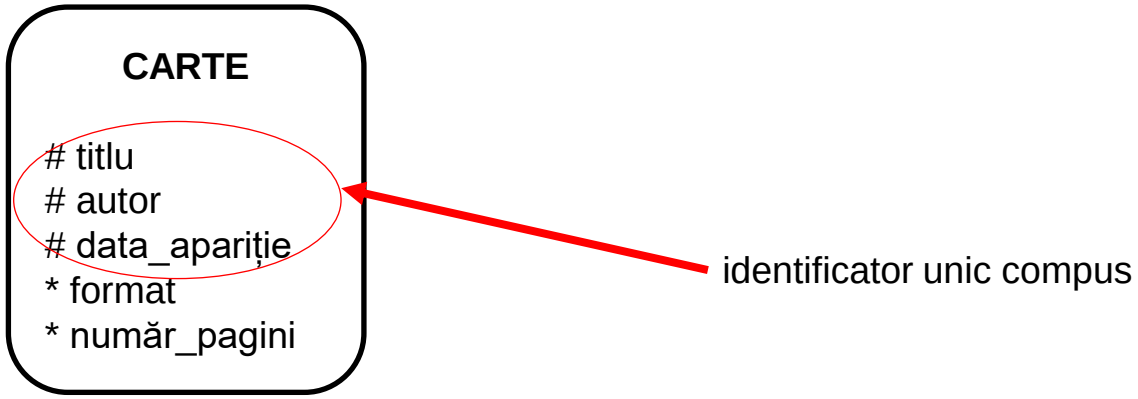


Tabela CĂRȚI

Titlu	Autor	Data apariției	Format	Număr pagini

cheie compusă

Cheie străină

- un atribut care este cheie într-un tabel dar apare şi în alt tabel, pentru a face legătura dintre cele două tabele, se numeşte ***cheie externă (străină)***
- o cheie externă este un câmp al tabelei care face referire la o cheie primară a altei tabele
- în procesul de mapare relația existentă între două entități din ERD se mapează într-un câmp de cheie străină în tabela corespunzătoare entității cu realația bifurcată

☞ *Exemplu:*

Diagrama ER

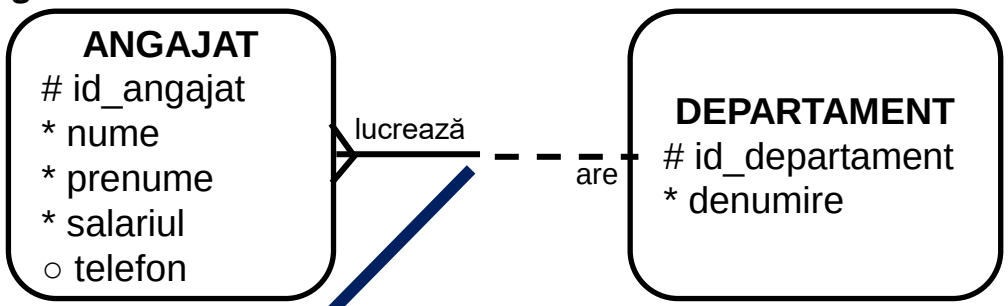


Tabela ANGAJAȚI

Id angajat	Id departament	Nume	Prenume	Salariul	Telefon
106	D1	Pop	Ana	2800	014 228
289	D2	Avram	Ion	2500	
149	D1	David	Mara	3000	

cheie primară

cheie externă

Tabela DEPARTAMENTE

Id departament	Denumire
D1	Comercial
D2	Economic

cheie primară

Reguli de integritate

O bază de date este realizabilă și poate fi în realitate utilă dacă datele pe care le conține sunt corecte.

Regulile de integritate garantează că datele introduse în baza de date sunt corecte şi valide.

În Oracle, regulile de integritate se definesc la crearea tabelelor folosind ***constrângerile***. O constrângere este o regulă aplicată bazei de date.

Tipuri de reguli de integritate:

- integritatea entităților
- integritatea de domeniu
- integritatea referențială

Integritatea entităților

- nicio coloană ce face parte din cheia primară nu poate avea valoare NULL (lipsă valoare)
- pentru fiecare înregistrare din tabelă, cheia primară trebuie să fie unică

☞ *Exemplu:*

Entitatea ANGAJAT

ANGAJAT
id_angajat
* nume
* prenume
* salariul
○ telefon

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656
126	Zota	Ema	4500	
405	Micu	Dinu	3200	014 352
331	Niță	Leo	4100	016 551

Integritatea de domeniu

- valorile introduse într-un câmp al tabelei trebuie să aparțină unui domeniu stabilit

☞ *Exemplu:*

Entitatea ELEV

ELEV
nr_matricol
* nume
* prenume
* clasa
* media

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Profil	Media
11	Popescu	Vasile	real	9.50
42	Adam	Angela	umanist	10
91	Gheorghe	Dana	real	8.50
45	Enache	Mircea	real	8.50
77	David	Adrian	real	9.00
62	Coca	Emil	umanist	10
21	Radu	Sandu	umanist	9.50

Integritatea referențială

- fiecare valoare a cheii străine trebuie să corespundă unei valori a cheii primare din tabela referită

☞ *Exemplu:*

Diagrama ER

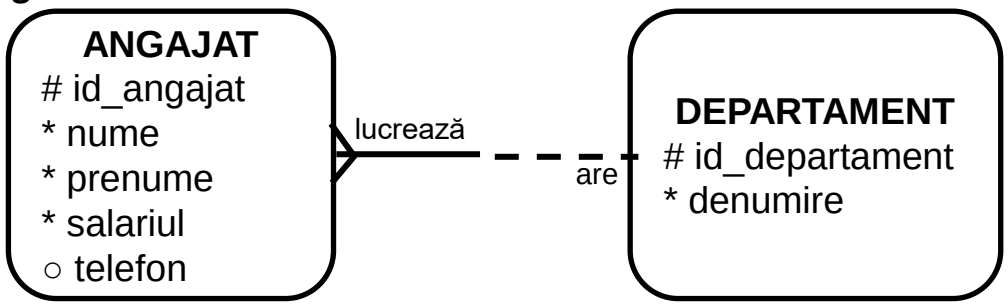


Tabela ANGAJAȚI

Id angajat	Id departament	Nume	Prenume	Salariul	Telefon
106	D1	Pop	Ana	2800	014 228
289	D2	Avram	Ion	2500	
149	D2	David	Mara	3000	

Tabela DEPARTAMENTE

Id departament	Denumire
D1	Comercial
D2	Economic

Programe de validare și de acțiune

Întocmirea diagramei entități-relații se face ținând cont de unele aspecte precum:

- modul de funcționare a afacerii modelate;
- natura datelor care trebuie memorate în baza de date;
- modul în care datele sunt relaționate;
- etc.

Aceste aspecte pot fi implementate prin regulile afacerii modelate.

Regulile simple ale afacerii se pot implementa prin intermediul relațiilor dintre entități și se numesc ***reguli structurale***.

Regulile care se referă la procesul afacerii se pot implementa prin anumite funcții sau proceduri și poartă denumirea de ***reguli procedurale***.

Într-o bază de date se pot include funcții sau programe care se rulează automat în cazul efecturăii unor operații (adăugare, modificare, ștergere, etc.) asupra datelor unei tabele.

Aceste funcții sau programe stocate în baza de date au rolul de a păstra integritatea referențială a bazei de date și se numesc ***programe de validare*** sau ***programe de acțiune***.

În Oracle aceste programe de validare se numesc ***declanșatoare*** (***triggere***) și se scri folosind limbajul PL/SQL.

Operații specifice prelucrării bazelor de date

Proiectarea unei baze de date presupune:

- întocmirea scenariului afacerii modelate;
- analiza cerințelor informaționale și definirea datelor;
- definirea tabelor, a structurii acestora și a relațiilor dintre tabele.

După etapa de proiectare a bazei de date urmează prelucrarea bazei de date. Prelucrarea unie baze de date se face prin două operații specifice:

- interogarea bazei de date;
- crearea rapoartelor.

Interogarea bazelor de date

- interogarea bazei de date reprezintă unul din scopurile principale ale organizării datelor în tabele;
- interogarea (cererea) presupune extragerea dalelor din tabelele bazei de date în funcție de anumite criterii, fără ca datele să fie șterse sau eliminate din baza de date;

Crearea rapoartelor

- rapoartele permit extragerea datelor din baza de date și prezentarea lor într-un anumit format;
- un raport aranjează datele extrase din baza de date într-un format prestabilit și poate fi vizualizat sau listat la imprimantă;

Introducere în SQL
Structura comenzilor SQL

- Crearea și modificarea structurii tabelelor
- Inserarea, modificarea, ștergerea datelor în tabele
- Selecție și proiecție
- Interogări simple

Introducere în SQL

Pentru interacţiunile utilizatorilor cu baza de date este necesar un limbaj prietenos, cu o sintaxă simplă, numit limbaj de interogare (QL - Query Language).

Un limbaj de interogare a bazelor de date include facilităţi pentru:

- crearea și definirea structurii bazei de date;
- inserarea, ştergerea şi modificarea datelor din baza de date.

Printre aceste limbaje se numără limbajul standard de interogare SQL.

Limbajul SQL

Limbajul SQL (Structured Query Language - limbaj de interogare structurat) este un limbaj de programare utilizat în prelucrarea bazelor de date structurate conform modelului relaţional.

Unul din motivele popularităţii de care se bucură acest limbaj este faptul că el a fost standardizat de American National Standards Institute (ANSI).

SQL a fost inițial dezvoltat și comercializat de compania IBM în anul 1974. Compania Relational Software Inc. (în prezent Oracle Corporation) a introdus în anul 1979 prima implementare comercială disponibilă de SQL, Oracle.

Crearea și modificarea structurii tabelelor

Crearea tabelelor

Crearea unei tabele a bazei de date presupune:

- stabilirea numelui tabelei;
- sabilirea coloanelor tabelei;
- stabilirea tipului de dată pe care îl au coloanele tabelei;
- declararea restricțiilor (constrângerilor) care asigură integritatea datelor.

Crearea unei tabele a bazei de date se face folosind comanda CREATE TABLE.

Comanda CREATE TABLE

Sintaxa:

```
CREATE TABLE nume_tabelă
(     coloana1 tip1 [tip_constrângere],
      coloana2 tip2 [tip_constrângere],
      …
      coloanan tipn [tip_constrângere]
);
```

unde:

- **nume_tabelă** este identificatorul tabelei
- **coloana$_1$**, **coloana$_2$**, …, **coloana$_n$** reprezintă numele coloanelor
- **tip$_1$**, **tip$_2$**, …, **tip$_n$** reprezintă tipul datelor memorate în coloane
- **tip_constrângere** reprezintă regulile de integritate care asigură că datele introduse sunt corecte și valide

Regulile de integritate se definesc la crearea tablei folosind ***constrângerile***.

Constrângeri:

- NOT NULL – valoarea din coloană trebuie obligatoriu completată
- PRIMARY KEY – coloana/coloanele identifică în mod unic înregistrările din tabelă
- UNIQUE – valoarile coloanei trebuie să fie unice pentru toate liniile tabelei
- FOREIGN KEY – stabilește o relație de cheie străina între coloana tabelei care se creează și coloana tabelei de referință
- CHECK – stabilește o condiție ce trebuie să fie îndeplinită la introducerea datelor în coloana respectivă

☞ *Exemplu:*

Diagrama ER

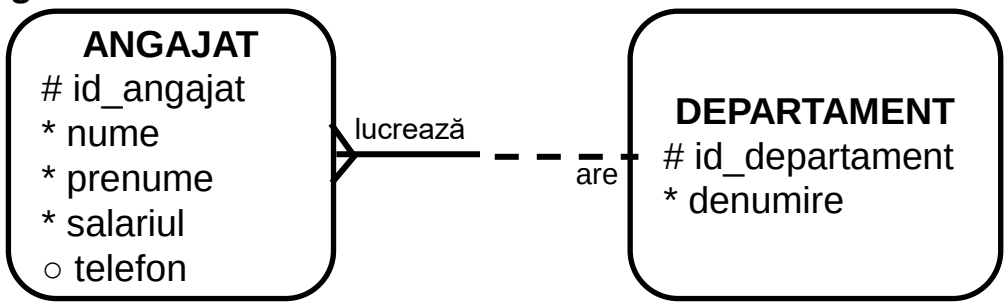


Tabela ANGAJAȚI

Id angajat	Id departament	Nume	Prenume	Salariul	Telefon
106	D1	Pop	Ana	2800	014 228
289	D2	Avram	Ion	2500	
149	D1	David	Mara	3000	

Tabela DEPARTAMENTE

Id departament	Denumire
D1	Comercial
D2	Economic

Id angajat	Id departament	Nume	Prenume	Salariul	Telefon
106	D1	Pop	Ana	2800	014 228
289	D2	Avram	Ion	2500	
149	D1	David	Mara	3000	

Id departament	Denumire
D1	Comercial
D2	Economic

Crearea tabelei DEPARTAMENTE

```
CREATE TABLE departamente
(  id_departament VARCHAR2(3) PRIMARY KEY,
   denumire VARCHAR2(25) NOT NULL
);
```

Id angajat	Id departament	Nume	Prenume	Salariul	Telefon
106	D1	Pop	Ana	2800	014 228
289	D2	Avram	Ion	2500	
149	D1	David	Mara	3000	

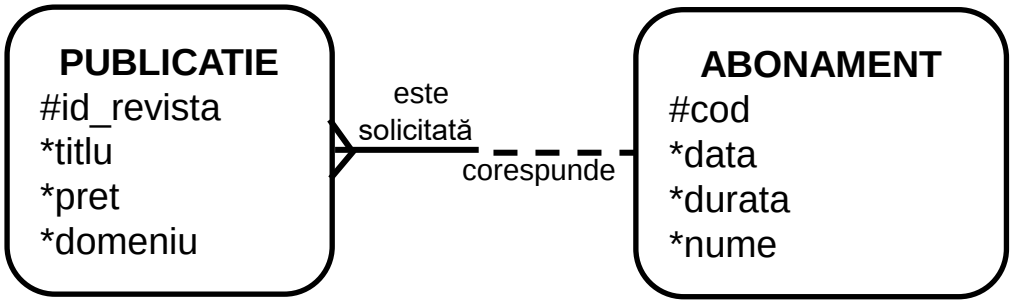
Id departament	Denumire
D1	Comercial
D2	Economic

Crearea tabelei ANGAJAȚI

```
CREATE TABLE angajați
(  id_angajat NUMBER(5) PRIMARY KEY,
   id_departament VARCHAR2(3) REFERENCES
                departamente(id_departament),
    nume VARCHAR2(20) NOT NULL,
    prenume VARCHAR2(20) NOT NULL,
    salariul NUMBER(6) NOT NULL,
    telefon VARCHAR2(10) );
```

✍ ***Exercițiu:***

Scrieți declarațiile SQL pentru crearea tabelelor din diagrama entități-relații de mai jos.



Modificarea structurii tabelelor

Modificarea structurii unei tabele presupune:

- adăugarea de coloane;
- ștergerea de coloane;
- modificarea definiției unei coloane;
- crerarea unei noi constrângeri;
- ștergerea unor constrângeri existente.

Modificarea structurii unei tabele a bazei de date se face folosind comanda ALTER TABLE.

Adăugarea unei noi coloane

Se utilizează clauza ADD a comenzii ALTER TABLE.

Sintaxa:

```
ALTER TABLE nume_tabelă
ADD coloană tip_data tip_constrângere
```

unde:

- **`nume_tabelă`** este identificatorul tabelei
- **`coloana`** reprezintă numele coloanelor
- **`tip_data`** reprezintă tipul datelor memorate în coloane
- **`tip_constrângere`** reprezintă regulile de integritate care asigură că datele introduse sunt corecte și valide

☞ *Exemplu:*

Entitatea ANGAJAT

ANGAJAT
id_angajat
* nume
* prenume
* salariul
○ telefon

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon

Adăugarea câmpului obigatoriu ***data_angajării*** de tip dată calendaristică.

```
ALTER TABLE angajați
ADD data_angajării DATE NOT NULL
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon	Data angajării

Ștergerea unei coloane

Se utilizează clauza DROP COLUMN a comenzii ALTER TABLE.

Sintaxa:

```
ALTER TABLE nume_tabelă
DROP COLUMN coloană
```

unde:

- **`nume_tabelă`** este identificatorul tabelei
- **`coloana`** reprezintă numele coloanelor

☞ *Exemplu:*

Entitatea ANGAJAT

ANGAJAT
id_angajat
* nume
* prenume
* salariul
○ telefon
* data_angajării

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon	Data angajării

Ștergerea câmpului obigatoriu ***data_angajării*** de tip dată calendaristică.

```
ALTER TABLE angajați
DROP COLUMN data_angajării
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon

Modificarea unei coloane

Se utilizează clauza MODIFY a comenzii ALTER TABLE.

Sintaxa:

```
ALTER TABLE nume_tabelă
MODIFY coloană tip_dată
```

unde:

- **nume_tabelă** este identificatorul tabelei
- **coloana** reprezintă numele coloanelor
- **tip_data** reprezintă tipul datelor memorate în coloane

☞ *Exemplu:*

Entitatea ANGAJAT

ANGAJAT
id_angajat
* nume
* prenume
* salariul
○ telefon
* data_angajării

Modificarea câmpului opțional ***telefon*** din tipul VARCHAR2 în tipul NUMBER.

```
ALTER TABLE angajați
MODIFY telefon NUMBER(9)
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon

Adăugarea unei constrângeri

Se utilizează clauza ADD CONSTRAINT a comenzii ALTER TABLE.

Sintaxa:

```
ALTER TABLE nume_tabelă
ADD CONSTRAINT nume_constrângere
                tip_constrângere
```

unde:

- **nume_tabelă** este identificatorul tabelei
- **nume_constrângere** reprezintă identificatorul constrângerii
- **tip_constrângere** reprezintă regula de integritate care asigură că datele introduse sunt corecte și valide

☞ *Exemplu:*

Entitatea ANGAJAT

ANGAJAT
id_angajat
* nume
* prenume
* salariul
○ telefon
* data_angajării

Adăugarea constrângerii PRIMARY KEY câmpului ***id_angajat***, modificându-l în câmp de cheie primară.

```
ALTER TABLE angajați
ADD CONSTRAINT cheie_primară
                    PRIMARY KEY (id_angajat)
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon

Ștergerea unei constrângeri

Se utilizează clauza DROP CONSTRAINT a comenzii ALTER TABLE.

Sintaxa:

```
ALTER TABLE nume_tabelă
DROP CONSTRAINT nume_constrângere
```

unde:

- **nume_tabelă** este identificatorul tabelei
- **nume_constrângere** reprezintă identificatorul constrângerii

☞ *Exemplu:*

Entitatea ANGAJAT

ANGAJAT
id_angajat
* nume
* prenume
* salariul
○ telefon
* data_angajării

Eliminarea constrângerii PRIMARY KEY a câmpului ***id_angajat***.

```
ALTER TABLE angajați
DROP CONSTRAINT cheie_primară
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon

Inserarea, ștergerea și modificarea datelor

Inserarea datelor în tabele

Inserarea sau adăugarea datelor în tabelele bazei de date presupune popularea tabelelor cu date.

Inserarea datelor în tabelele bazei de date se face folosind comanda INSERT.

Comanda INSERT

Sintaxa:

```
INSERT INTO (lista_coloane)
VALUES (lista_valori)
```

unde:

- **`lista_coloane`** precizează coloanele care vor primi date
- **`lista_valori`** specifică valorile pe care le vor lua, pe rând, coloanele din lista de coloane

☞ ***Exemplu:***

```
INSERT INTO angajați (id_angajat, nume, prenume,
salariu, telefon, data_nașterii)
VALUES (140,'Pop','Ana',1900,'0415222','11.03.2000')
```

sau

```
INSERT INTO angajați
VALUES (141,'Alexa','Dan',1800,null,'15.05.2005')
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon	Data angajării
140	Pop	Ana	1900	041 52 22	11.03.2000
141	Alexa	Dan	1800		15.05.2005

Ștergerea datelor din tabele

Ștergera datelor din tabelele bazei de date se face folosind comanda DELETE.

Comanda DELETE

Sintaxa:

```
DELETE FROM nume_tabelă
WHERE condiție
```

unde:

- **nume_tabelă** este identificatorul tabelei
- **condiție** precizează criteriul de selecție a liniilor din tabelă ce vor fi șterse

Observație:

Dacă lipsește clauza WHERE vor fi șterse toate liniile tabelei.

```
DELETE FROM nume_tabelă
```

☞ ***Exemplu:***

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon	Data angajării
140	Pop	Ana	1900	041 52 22	11.03.2000
141	Alexa	Dan	1800		15.05.2005

```
DELETE FROM angajați
WHERE id_angajat = 141
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon	Data angajării
140	Pop	Ana	1900	041 52 22	11.03.2000

Modificarea datelor dintr-o tabelă

Modificarea înregistrărilor (liniilor) din tabelele bazei de date se face folosind comanda UPDATE.

Comanda UPDATE

Sintaxa:

```
UPDATE nume_tabelă
SET coloană1 = valoare1, coloană2 = valoare2, ...
WHERE condiție
```

unde:

- **nume_tabelă** este identificatorul tabelei
- **coloană$_1$, coloană$_2$, ...** reprezintă identificatorul coloanei
- **valoare$_1$, valoare$_2$, ...** reprezintă noua valoare a coloanei
- **condiție** precizează criteriul de selecție a liniilor din tabelă ce vor fi actualizate

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon	Data angajării
140	Pop	Ana	1900	041 52 22	11.03.2000
141	Alexa	Dan	1800		15.05.2005

```
UPDATE angajați
SET prenume = 'Marian'
WHERE id_angajat = 141
```

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon	Data angajării
140	Pop	Ana	1900	041 52 22	11.03.2000
141	Alexa	Marian	1800		15.05.2005

Selecție și proiecție

Interogarea bazelor de date

Interogarea (query), este operaţia prin care se obţin datele dorite dintr-o bază de date, selectate conform unui anumit criteriu (condiţie).

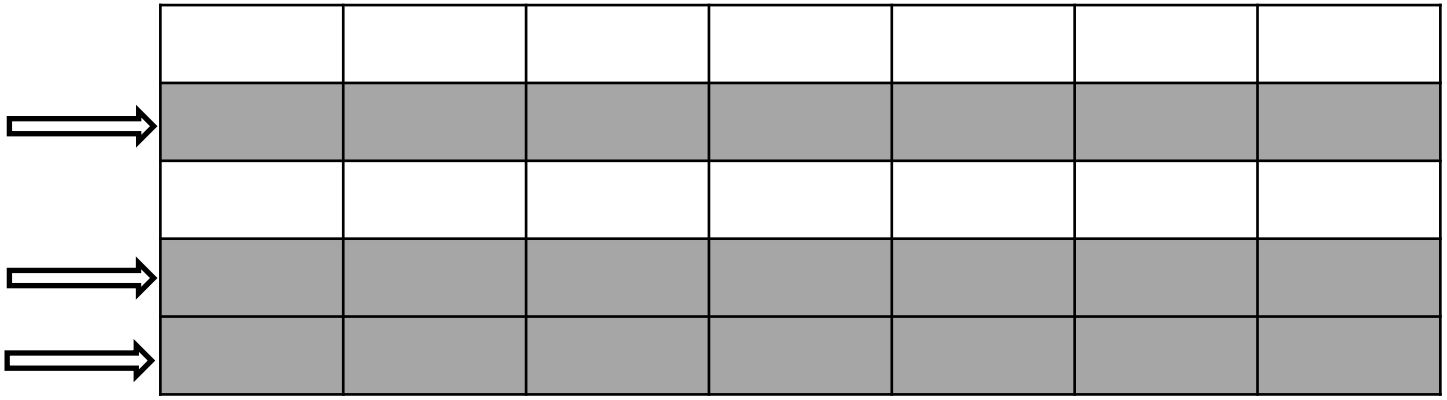
Pentru interogarea bazelor de date se utilizează comanda SELECT.

Comanda SELECT realizează trei tipuri de operații:

- selecție (selection);
- proiecție (projection);
- joncțiune (join).

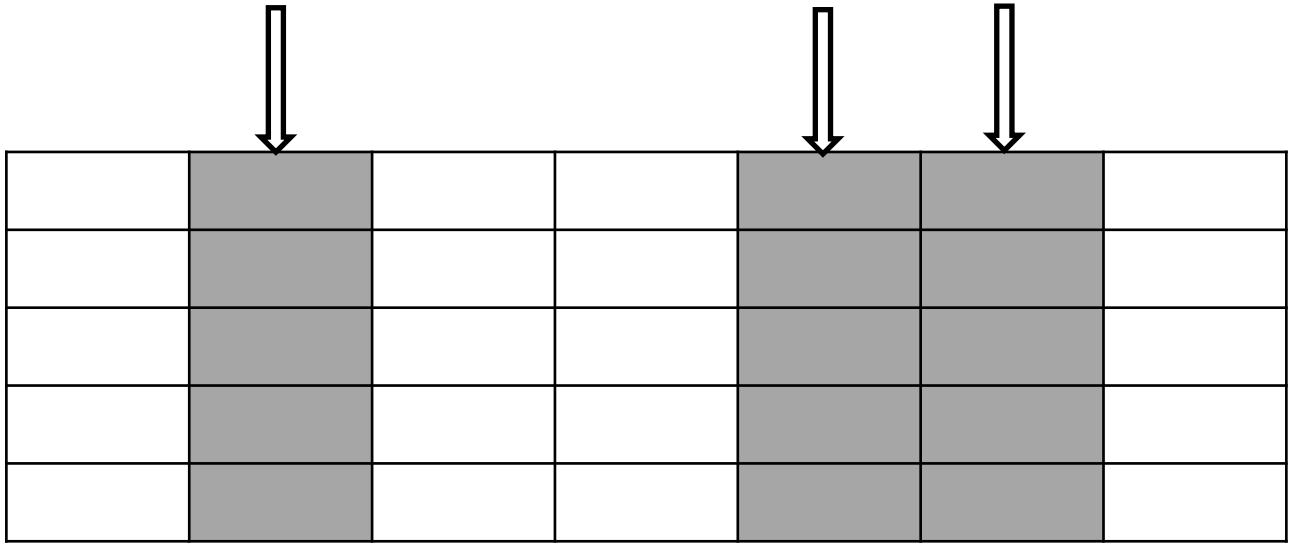
1. Selecția

- filtrează liniile care vor fi afișate
- utilizează clauză WHERE pentru a stabili criteriul de filtrare



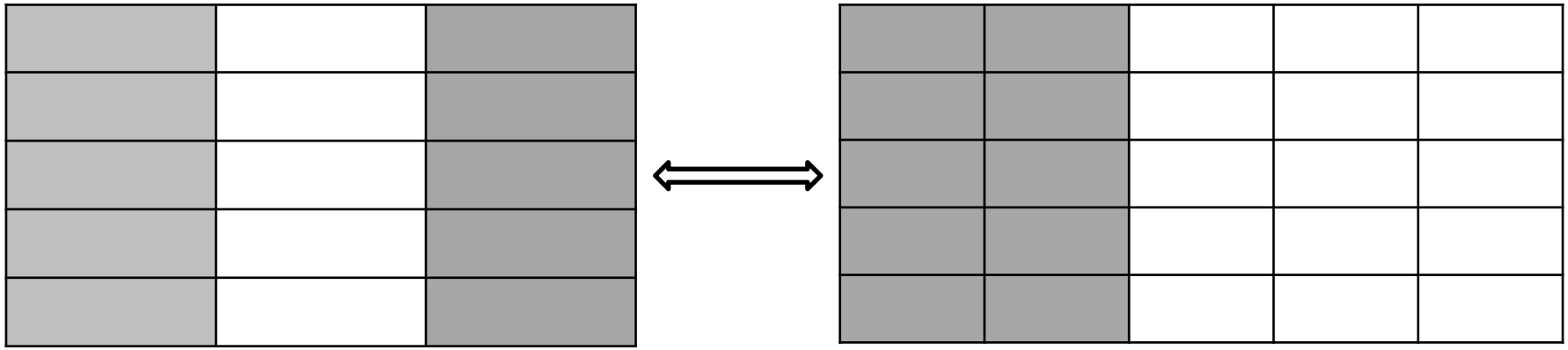
2. Proiecția

- filtrează coloanelor care vor fi afișate
- se realizează în clauza SELECT



3. Join

- îmbină date din două sau mai multe tabele, care vor fi afișate într-un singur raport
- utilizează operațiile join



Interogări simple

Operația de interogare are ca scop obținerea de informații din două sau mai multe tabele.

Interogările sunt de două tipuri:

- interogări simple;
- interogări multiple.

O ***interogare simplă*** (selecție și sau proiecție) filtrează și afișează date dintr-o singură tabelă a bazei de date.

O ***interogare multiplă*** (join) combină date din două sau mai multe tabele și le afișează într-un singur raport.

Interogarea bazelor de date se face utilizând comanda SELECT.

Comanda SELECT

Sintaxa:

```
SELECT  coloana1, coloana2, ...
FROM nume_tabelă;
```

unde:

- **coloana$_1$, coloana$_2$, ...** precizează coloanele care vor fi afișate
- **SELECT** este clauza care precizează lista coloanelor sau expresiilor ce se vor afișa
- **FROM** este clauza care precizează tabela din care se vor extrage coloanele ce vor fi afișate

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  id_angajat, nume, salariu
FROM angajați;
```

Proiecția tabelei ANGAJAȚI

Id angajat	Nume	Salariul
106	Pop	2800
289	Avram	2500
149	David	3000
394	Adam	2200

Filtrarea liniilor

O tabelă a unei baze de date poate avea foarte multe înregistrări (linii). Pentru a selecta anumite linii ale unei tabele se adaugă clauza WHERE la comanda SELECT.

Clauza WHERE realizează operația de selecție.

Clauza WHERE

Sintaxa:

```
SELECT  coloana1, coloana2, ...
FROM nume_tabelă
WHERE condiție;
```

unde:

- **coloana$_1$, coloana$_2$, ...** precizează coloanele care vor fi afișate
- **nume_tabelă** reprezintă tabela din care vor fi preluate coloanele
- **SELECT** este clauza care precizează lista coloanelor sau expresiilor ce se vor afișa
- **FROM** este clauza care precizează tabela din care se vor extrage coloanele ce vor fi afișate
- **WHERE** este clauza care specifică condiția/condițiile (**condiție**) care trebuie îndeplinite de întregistrările tabelei pentru a fi afișate

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  *
FROM angajați
WHERE salariul > 2600;
```

Selecția tabelei ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
149	David	Mara	3000	017 548

Aliasul unei coloane

Când se afișează rezultatul unei interogări, SQL folosește în mod normal numele coloanei selectate ca header de coloană.

Pentru a schimba numele coloanei în raportul afișat se utilizează un alias pentru coloana respectivă.

Aliasul se adaugă în clauza SELECT în lista de coloene, după numele coloanei, folosind cuvântul cheie AS. Dacă alias-ul conține spații sau caractere speciale (cum ar fi # sau $), trebuie scris între semne de citare (" ").

Sintaxa:

```
SELECT  coloana1 AS ”nume nou”,
        coloana2 AS ”nume nou”,
        ...
```

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  nume, salariul,
        salariul*10 AS ”Salariul mărit”
FROM angajați;
```

Selecția tabelei ANGAJAȚI

Nume	Salariul	Salariul mărit
Pop	2800	28000
David	3000	30000

Operatorul de concatenare

Coloanele unei tabele pot fi legata cu alte coloane, expresii aritmetice, sau valori constante pentu a crea o expresie de caractere, folosind operatorul de concatenare (||).

Coloanele din stânga sau dreapta operatorului sunt combinate pentru a forma o singură coloană de ieșire.

Sintaxa:

```
SELECT  coloana1 || coloana2 || ...
```

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  nume ||' '|| prenume ||
        ' are salariul egal cu '||
         salariul AS "Informații salariale"
FROM angajați WHERE salariul > 2600;
```

Informații salariale
Pop Ana are salariul egal cu 2800
David Mara are salariul egal cu 3000

Eliminarea liniilor duplicate

Dacă nu este indicat altfel, SQL afișează rezultatele unei cereri fără a elimina rândurile duplicat.

Pentru a elimina rândurile duplicat, trebuie inclus cuvântul cheie DISTINCT în clauza SELECT imediat după cuvantul cheie SELECT.

Sintaxa:

```
SELECT  DISTINCT coloana
```

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2800	012 656

```
SELECT  DISTINCT salariul
FROM angajați;
```

Salariul
2800
2500
3000

Programare SQL

- Funcții
- Gruparea datelor
- Sortarea datelor
- Subinterogări
- Interogări multiple

Funcții

Funcțiile de grup sunt funcții care returnează o singură valoare pentru un grup sau set de linii dintr-o tabelă a bazei da date.

Principalele funcții de grup sunt:

- COUNT
- MIN
- MAX
- SUM
- AVG

1. *Funcția COUNT*

- returnează numărul de linii
- dacă argumentul este * se numără şi valorile NULL, altfel se numără doar valorile nenule
- argumentul poate lua tipul CHAR, VARCHAR2, NUMBER, DATE.

Sintaxa:

COUNT (expresie)

☞ ***Exemplu:***

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  COUNT (id_angajat)
FROM angajați;
```

2. *Funcția MIN*

- returnează valoare minimă
- argumentul poate lua tipul CHAR, VARCHAR2, NUMBER, DATE

Sintaxa:

MIN (nume_coloană)

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  MIN (salariul)
FROM angajați;
```

3. *Funcția MAX*

- returnează valoare maximă
- argumentul poate lua tipul CHAR, VARCHAR2, NUMBER, DATE

Sintaxa:

MAX (nume_coloană)

☞ ***Exemplu:***

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  MAX (salariul)
FROM angajați;
```

4. *Funcția SUM*

- returnează suma valorilor
- argumentul trebuie să fie numeric.

Sintaxa:

SUM (nume_coloană)

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  SUM (salariul)
FROM angajați;
```

5. *Funcția AVG*

- returnează media valorilor
- argumentul trebuie să fie numeric.

Sintaxa:

AVG (nume_coloană)

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  AVG (salariul)
FROM angajați;
```

Gruparea datelor

Gruparea liniilor unei tabele permite obținerea de informații despre grupurile respective. Gruparea datelor se poate face folosind clauza GROUP BY.

Clauza GROUP BY este utilizată pentru a diviza liniile unui tabel în grupuri. Pentru a returna informaţia corespunzătoare fiecărui astfel de grup, pot fi utilizate funcţiile de grup.

Clauza GROUP BY poate fi utilizată și fără funcţii de grup, caz în care liniile tabelei vor fi afișate după un anumit criteriu.

☞ *Exemplul 1:*

Tabela ANGAJAȚI

Id angajat	Id departament	Nume	Prenume	Salariul	Telefon
106	D1	Pop	Ana	2800	014 228
289	D2	Avram	Ion	2500	
149	D1	David	Mara	3000	

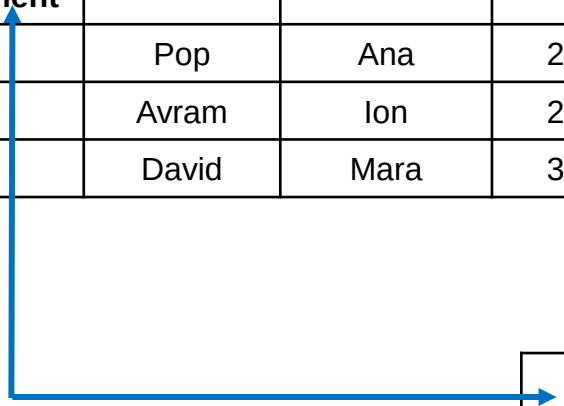


Tabela DEPARTAMENTE

Id departament	Denumire
D1	Comercial
D2	Economic

```
SELECT  AVG (salariul)
FROM angajați
GROUP BY id_departament;
```

☞ *Exemplul 2:*

Tabela ANGAJAȚI

Id angajat	Id departament	Nume	Prenume	Salariul	Telefon
106	D1	Pop	Ana	2800	014 228
289	D2	Avram	Ion	2500	
149	D1	David	Mara	3000	

Tabela DEPARTAMENTE

Id departament	Denumire
D1	Comercial
D2	Economic

```
SELECT id_nagajat, nume, prenume, salariul
FROM angajați
GROUP BY id_departament;
```

Sortarea datelor

Sortarea liniilor unei tabele permite afișarea (la execuția unei comenzi SELECT) datelor unei tebele în ordine crescătoare sau descrescătoare. Sortarea datelor se poater face folosind clauza ORDER BY.

Clauza ORDER BY trebuie să fie ultima clauză într-o comană SELECT.

Pentru sortarea crescătoare a datelor se poate utiliza opțiunea ASC, însă aceasta este opțională deoarece implicit datele sunt sortate crescător. Pentru sortarea descrescătoare a datelor se utilizază opțiunea DESC.

☞ *Exemplu:*

Tabela ANGAJAȚI

Id angajat	Nume	Prenume	Salariul	Telefon
106	Pop	Ana	2800	014 228
289	Avram	Ion	2500	
149	David	Mara	3000	017 548
394	Adam	Dana	2200	012 656

```
SELECT  nume, prenume, salariul
FROM angajați
ORDER BY nume, prenume;
```

```
SELECT  nume, prenume, salariul
FROM angajați
ORDER BY salariul DESC;
```