Partea a II-a. Proiectarea bazelor de date

Capitolul 6. Tehnici de proiectare şi modele

În capitolele precedente s-au analizat modele de baze de date şi limbaje, presupunând în cele mai multe cazuri că există o bază de date cu care utilizatorii pot interacţiona.

În acest capitol se va pune problema proiectării unei baze de date conform cerinţelor utilizatorilor.

Proiectarea unei baze de date presupune definirea

- structurii
- caracteristicilor
- conţinutului

Utilizarea unor tehnici corespunzătoare este esenţială pentru crearea unui produs de calitate bună.

6.1. Procesul de proiectare a bazei de date

6.1.1. Ciclul de viaţă al informaţiilor din sistem

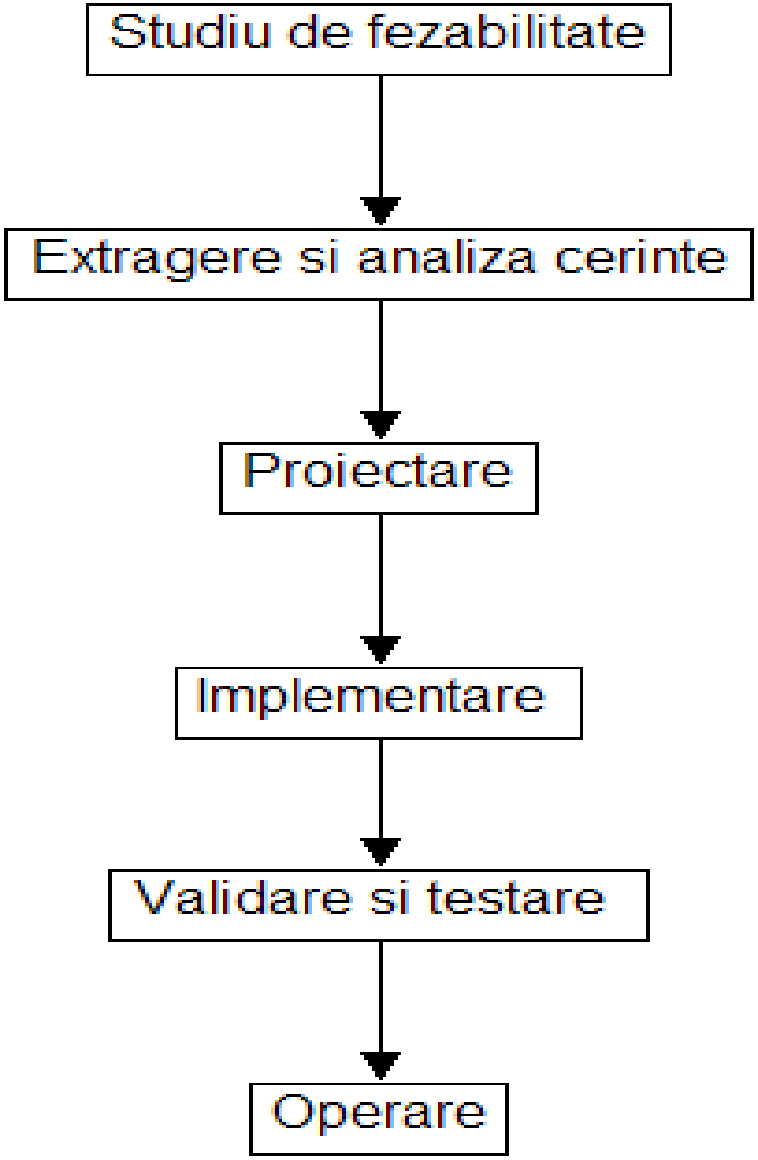


Figura 1. Ciclul de viaţă al unui sistem informaţional

Studiul de fezabilitate

- definirea cât mai precisă a costurilor diferitelor soluţii posibile
- stabilirea priorităţilor în crearea componentelor sistemului

Extragerea şi analiza cerinţelor

- definirea şi studiul proprietăţilor şi funcţionalităţii sistemului informaţional
- presupune interacţiunea cu clienţii pentru extragerea cerinţelor
- rezultatul este o descriere completă a datelor care intervin şi a operaţiilor care acţionează asupra datelor respective
- se stabilesc de asemenea necesităţile hardware şi software ale sistemului

Proiectarea

- *proiectarea bazei de date* - se stabilesc structura şi organizarea datelor
- *proiectarea operaţională* - se definesc caracteristicile programelor

Cei doi paşi sunt complementari şi se pot parcurge simultan sau consecutiv.

Descrierea datelor şi a programelor rezultate din această etapă este formală şi se referă la modele specifice.

Implementarea

- crearea sistemului informaţional în concordanţă cu structura şi caracteristicile precizate în etapa de proiectare
- se construieşte baza de date şi se scrie codul programelor

Validarea şi testarea

- verificarea funcţionării şi a calităţii sistemului informaţional
- testele trebuie să conţină, pe cât posibil, toate condiţiile de operare posibile

Operarea

- sistemul informaţional funcţionează îndeplinind scopurile pentru care a fost creat
- această activitate constă doar în managementul şi întreţinerea sistemului (presupunând ca nu există erori majore şi că nu este schimbată funcţionalitatea sistemului)

Observaţii

- în practică aceste etape nu sunt în general secvenţiale, deoarece în timpul uneia dintre activităţile menţionate este necesară revenirea la o etapă anterioară

- câteodată este necesară introducerea unei noi activităţi – *prototipizarea* – care constă în utilizarea unor unelte software specifice pentru crearea rapidă a unei versiuni simplificate a sistemului informaţional cu ajutorul căreia se testează funcţionalitatea acestuia

  - prototipul poate fi prezentat clienţilor pentru a verifica dacă au fost extrase şi modelate corect cerinţele acestora

6.1.2. Metodologii pentru proiectarea bazelor de date

O abordare structurată a proiectării bazelor de date constă în parcurgerea următoarelor etape:

- *descompunerea* activităţii de proiectare în paşi succesivi, independenţi unul de altul;
- o serie de *strategii* care trebuie urmate şi de *criterii* care permit o alegere în cazul în care există mai multe soluţii;
- *modele de referinţă* pentru a descrie intrările şi ieşirile diferitelor faze.

Proprietăţile care trebuie garantate de o anumită metodologie sunt:

- *generalitatea* – privitoare la aplicaţiile şi sistemele pe care rulează (şi astfel posibilitatea utilizării independent de o aplicaţie specifică şi de un sistem disponibil);
- *calitatea produsului*, care presupune acurateţe şi eficienţă;
- *uşurinţa utilizării*.

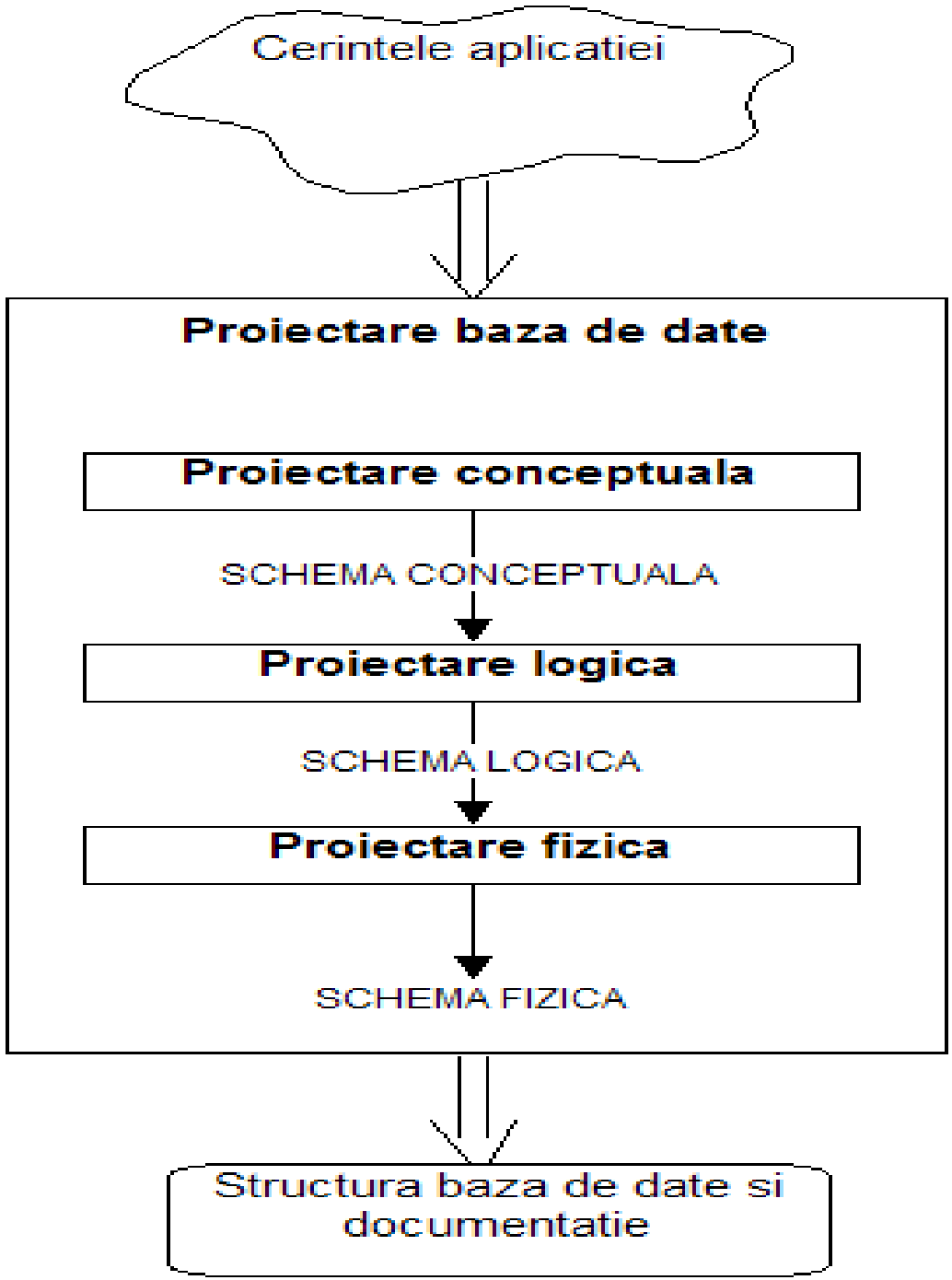


Figura 2. Fazele proiectării bazelor de date

Proiectarea conceptuală

- Scop: reprezentarea cerinţelor informale ale aplicaţiei în termenii descrierii complete şi formale dar independent de criteriul folosit pentru reprezentare în sistemul de management al bazei de date.

- Rezultat: *schema conceptuală* - *model de date conceptual* - permite descrierea organizării datelor la un nivel înalt de abstractizare fără a lua în considerare aspectele de implementare.

- Proiectantul trebuie să reprezinte *conţinutul* bazei de date fără a lua în considerare mijloacele prin care informaţiile respective vor fi implementate în sistem sau eficienţa programelor ce utilizează aceste informaţii.

Proiectarea logică

- Scop: Translarea schemei conceptuale într-un model de date disponibil pentru sistemul de gestiune al bazei de date

- Rezultat: *schema logică - model de date logic -* permite reprezentarea datelor într-o formă independentă încă de detaliile fizice deşi sistemul de gestiune al bazei de date folosit pentru implementare poate fi unul ce suportă acest model de date.

- Proiectantul trebuie să ia în considerare *criterii de optimizare*.

- Se utilizează frecvent tehnici formale pentru verificarea calităţii schemei logice.

- În cazul unui model de date relaţional, tehnica folosită este aceea a *normalizării*.

Proiectarea fizică

- Scop: schema logică se completează cu detalii de implementare fizică (organizarea fişierelor şi indecşi) puse la dispoziţie de SGBD.

- Rezultat: *schema fizică* - *model de date fizic* - depinde de sistemul de management al bazei de date şi ia în considerare criteriile pentru organizarea fizică a datelor.

Cerinţe ale aplicaţiei care sunt utilizate în cele trei faze ale proiectării BD

- *cerinţele de date* - conţinutul bazei de date
- *cerinţele operaţionale* - utilizarea bazei de date de clienţi sau programatori

În proiectarea conceptuală

- cerinţele de date furnizează majoritatea informaţiilor
- cerinţele operaţionale se folosesc doar pentru a verifica dacă schema conceptuală este completă

În proiectarea logică

- schema conceptuală (furnizată ca intrare), rezumă cerinţele de date
- cerinţele operaţionale se folosesc pentru a obţine schema logică

În proiectarea fizică

- schema logică şi cerinţele operaţionale se folosesc pentru optimizarea performanţelor sistemului
- trebuie luate în considerare caracteristicile SGBD-ului utilizat

Rezultatul procesului de proiectare a bazei de date constă în:

- schema conceptuală - furnizează o reprezentare de nivel înalt a bazei de date - folositoare pentru documentare.

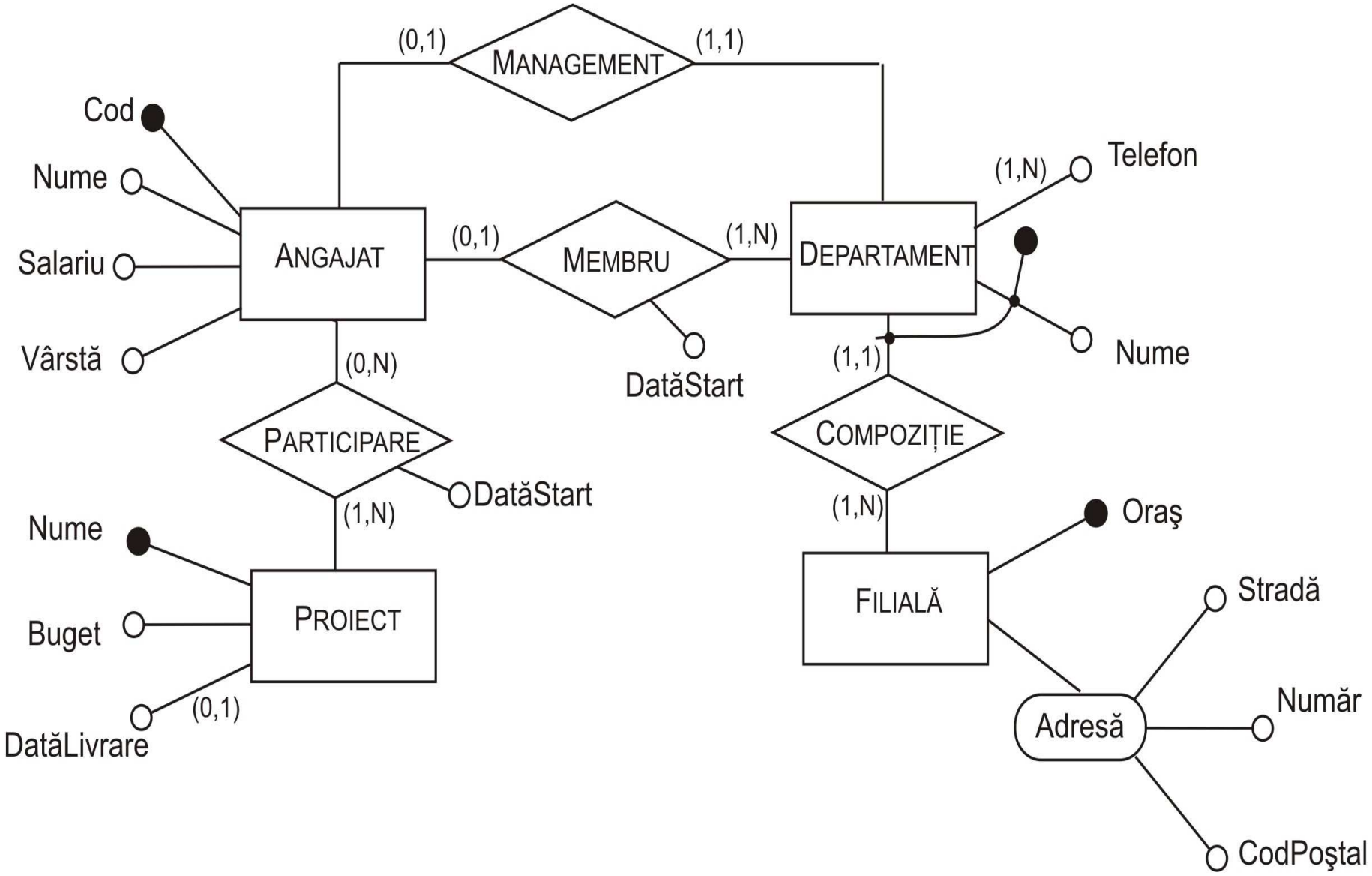
- schema logică - furnizează o descriere a conţinutului bazei de date şi, pe lângă aspectele de implementare, este utilă pentru realizarea interogărilor şi modificărilor bazei de date.

- schema fizică;

6.2. Modelul Entitate – Relaţie (Entity – Relantionship model)

- model de date conceptual ce pune la dispoziţie o serie de *construcţii*
  - descriu datele necesare unei aplicaţii într-o manieră uşor de înţeles şi independentă de criteriile de gestiune şi organizare a datelor din sistem
- construcţiile modelului E-R
  - definesc scheme ce descriu modul de organizare
  - dictează care *apariţii de date* (valori pe care baza de date le poate stoca la diferite momente de timp) sunt legale

Exemplu de model E-R



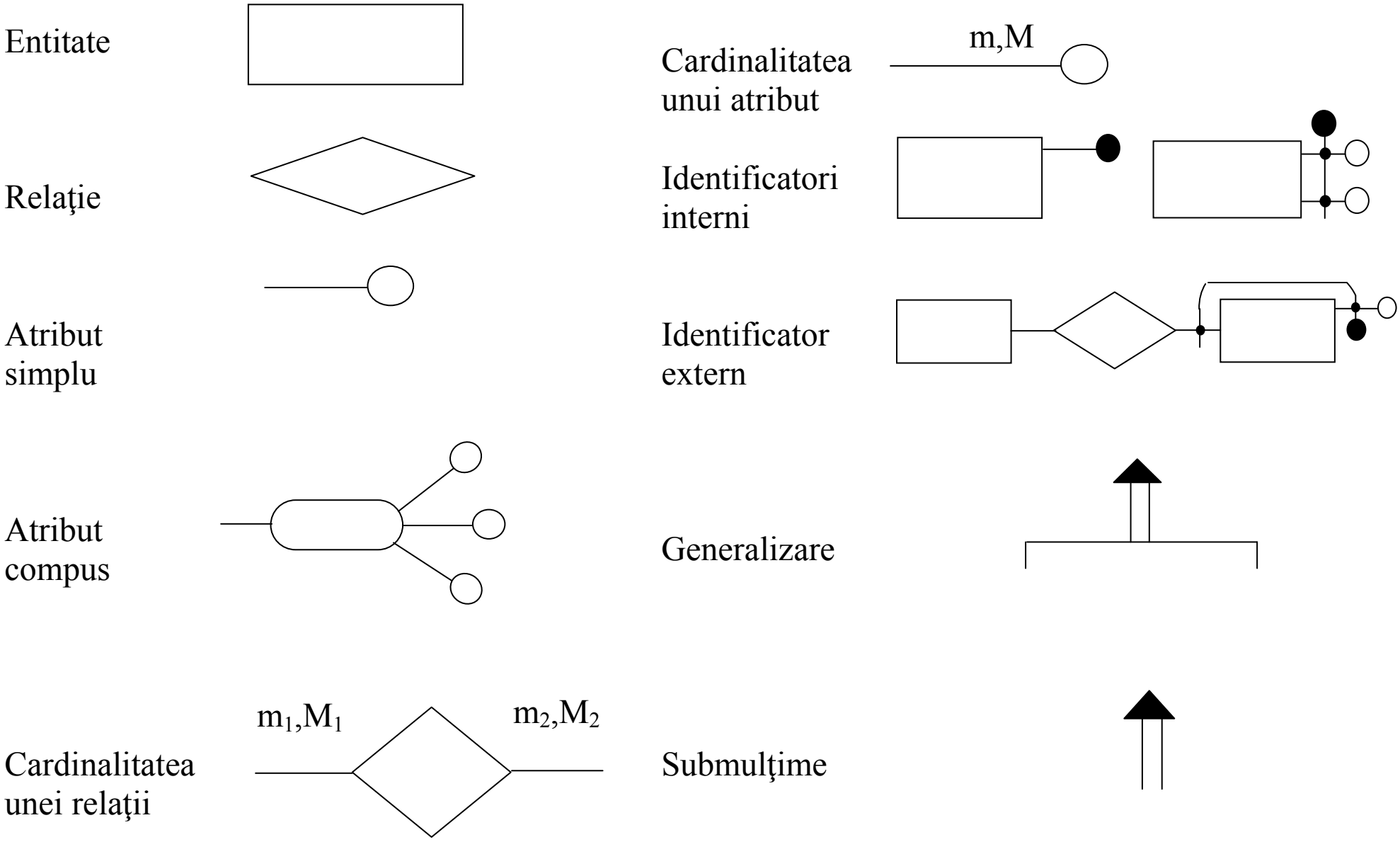


Figura 3. Construcţii ale modelului entitate-relaţie

Entităţi

- clase de obiecte ce au proprietăţi comune şi existenţă autonomă
- notaţia grafică - figura 4

  ORAŞ, DEPARTAMENT, ANGAJAT, VÂNZĂRI sunt exemple de entităţi într-o aplicaţie pentru o organizaţie comercială

- o apariţie a unei entităţi este un obiect al clasei reprezentate de entitatea respectivă

  oraşele Iaşi şi Bucureşti sunt apariţii ale entităţii ORAS

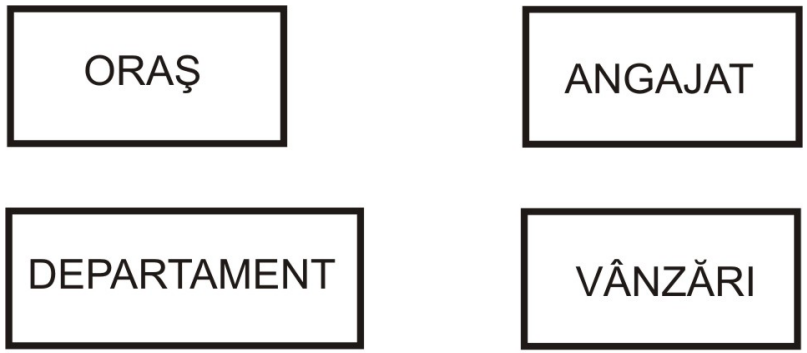


Figura 4. Exemple de entităţi

- într-o schemă, fiecare entitate are un nume unic.

Relaţii

- reprezintă legături logice între două sau mai multe entităţi

  REZIDENŢĂ este un exemplu de relaţie care poate exista între entităţile ORAŞ şi ANGAJAT

- o apariţie a unei relaţii este un n-tuplu format din apariţii ale entităţilor, câte o apariţie pentru fiecare entitate implicată

- exemplu - apariţii ale relaţiei EXAMEN între entităţile STUDENT şi CURS:

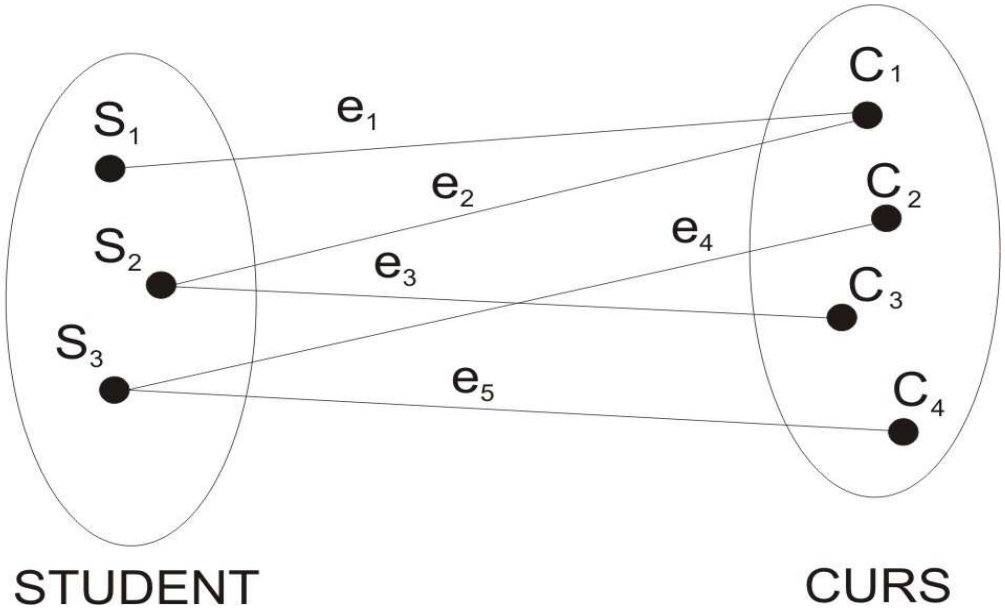


Figura 5. Exemple de apariţii ale relaţiei EXAMEN

- fiecare relaţie are un nume unic

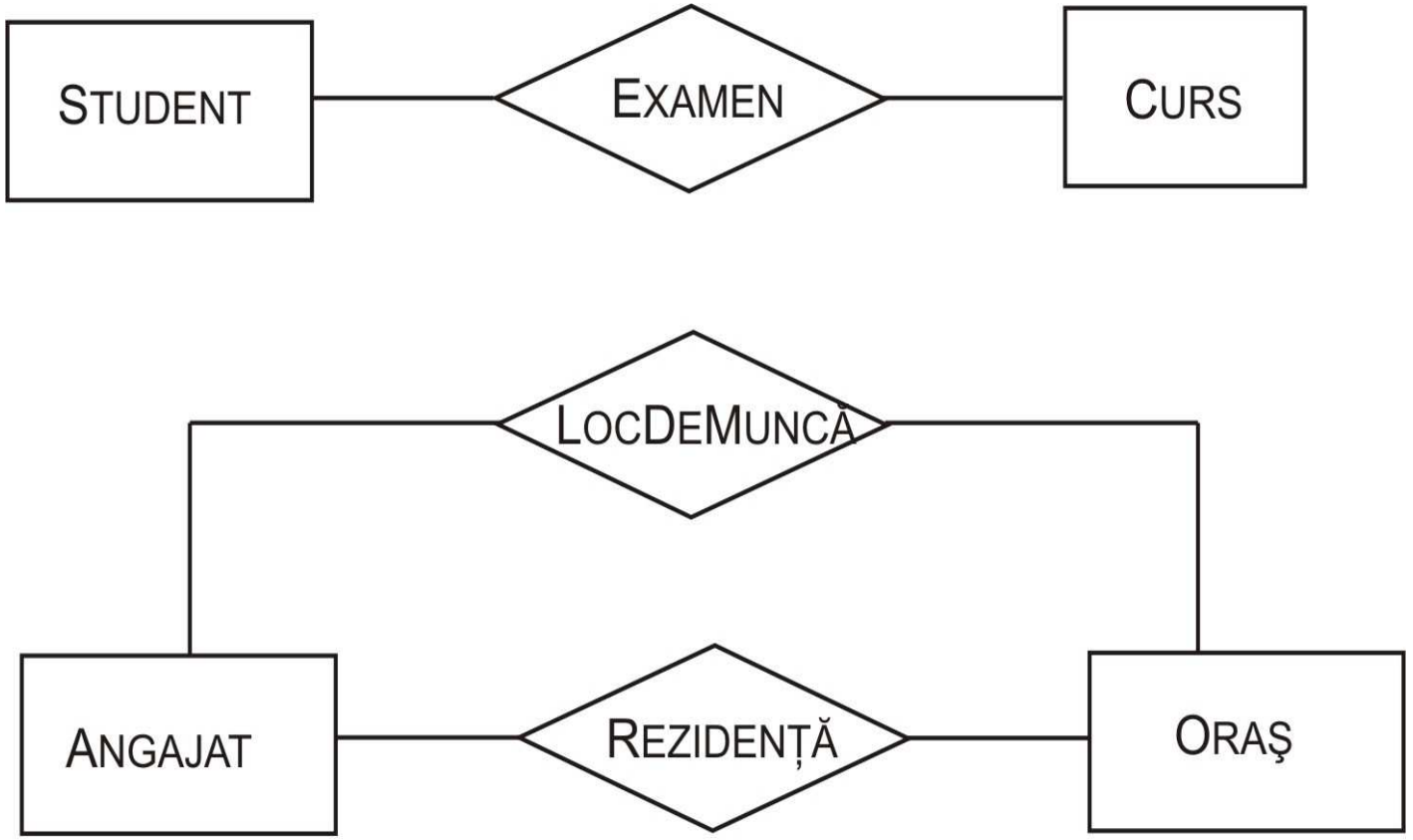


Figura 6. Exemple de relaţii

- între aceleaşi entităţi pot exista mai multe relaţii
- pentru numele relaţiilor este indicată folosirea substantivelor în locul verbelor, pentru a evita sugerarea unei „direcţii”

- mulţimea apariţiilor unei relaţii este o relaţie matematică între mulţimile de apariţii ale entităţilor implicate (este o submulţime a produsului cartezian al celor două mulţimi de apariţii ale entităţilor implicate)

  - este asigurat faptul că nici un n-tuplu nu va apărea de două ori pentru apariţiile unei relaţii

  - acest aspect are consecinţe importante:

    - relaţia EXAMEN din figura 6 nu are capacitatea de a raporta faptul că un student oarecare dă un examen de mai multe ori (deoarece aceasta ar trebui să producă perechi identice)

    - în acest caz, examenul trebuie reprezentat de o entitate legată de entităţile STUDENT şi CURS prin intermediul a două relaţii binare

- sunt posibile *relaţiile recursive* care reprezintă relaţii între o entitate şi ea însăşi

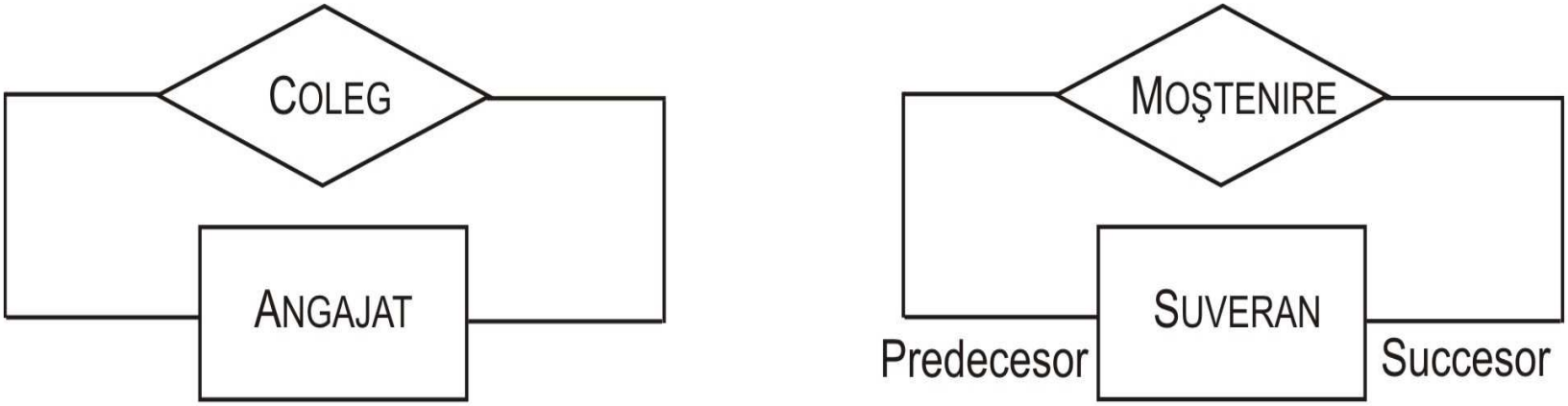


Figura 7. Exemple de relaţii recursive

- relaţia recursivă COLEG a entităţii ANGAJAT conectează perechi de oameni care lucrează împreună
- relaţia SUCCESIUNE a entităţii SUVERAN nu este simetrică $\Rightarrow$ este necesar să se asocieze identificatori liniilor din relaţia recursivă

- există relaţii care implică mai mult de două entităţi

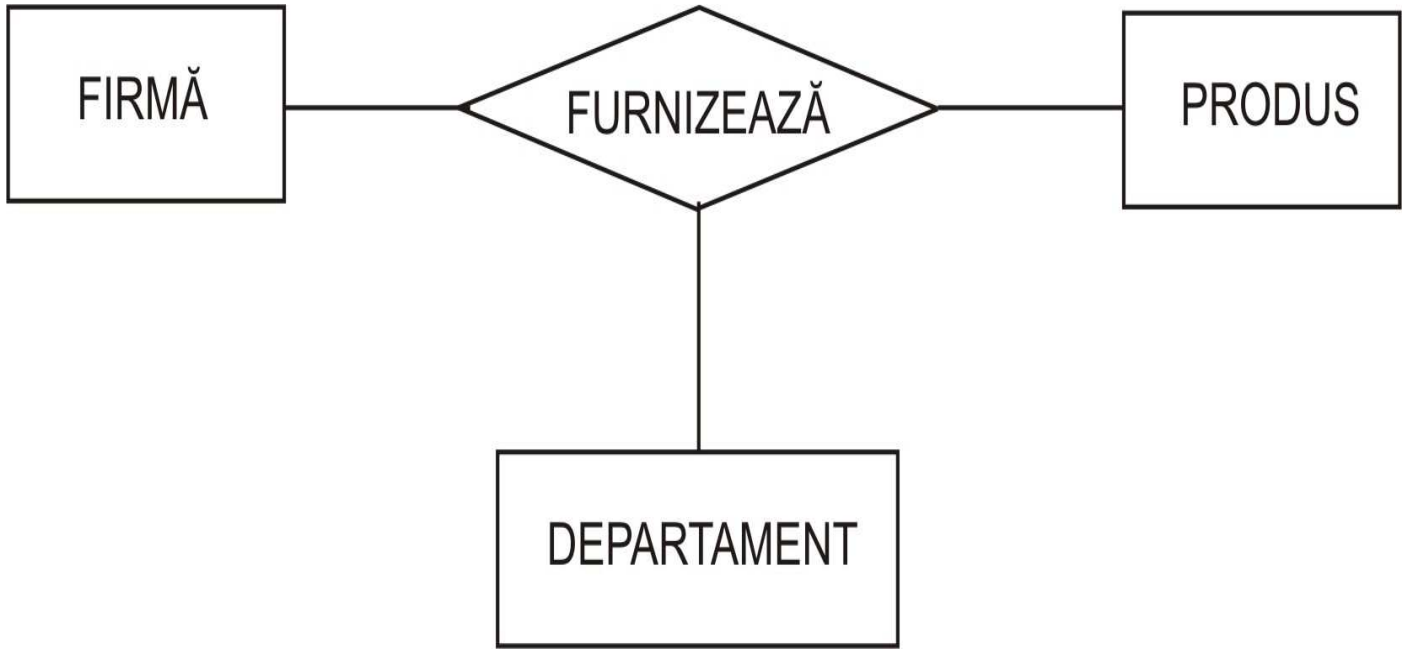


Figura 8. Exemplu de relaţie ce implică mai multe entităţi

- o apariţie a relaţiei FURNIZEAZĂ descrie faptul că o anumită firmă furnizează un anumit produs unui departament

Atribute

- descriu proprietăţile elementare ale entităţilor sau relaţiilor

  Ex.: Nume, Salariu, Vârstă sunt atribute pentru entitatea ANGAJAT iar Dată, Notă sunt atribute pentru relaţia EXAMEN

- un atribut asociază fiecărei apariţii a unei entităţi (sau relaţie) o valoare aparţinând unei mulţimi denumite *domeniul atributului*

- domeniul conţine valori admisibile pentru atribut

  Ex.: domeniul pentru atributul *Nume* poate fi orice şir de caractere de lungime 20, iar domeniul pentru atributul *Vârstă* poate fi orice număr între 18 şi 60

- domeniile nu sunt reprezentate grafic, ele fiind de obicei descrise în documentaţia asociată

- atributele pot fi

  simple

  compuse

Atribut compus - mulţime de atribute ale aceleaşi entităţi sau relaţii ale căror înţelesuri sau utilizări sunt strâns conectate

Ex.: atributele *Strada*, *NumărCasă*, *CodPoştal* ale entităţii PERSOANĂ pot fi grupate pentru a forma atributul compus *Adresă*

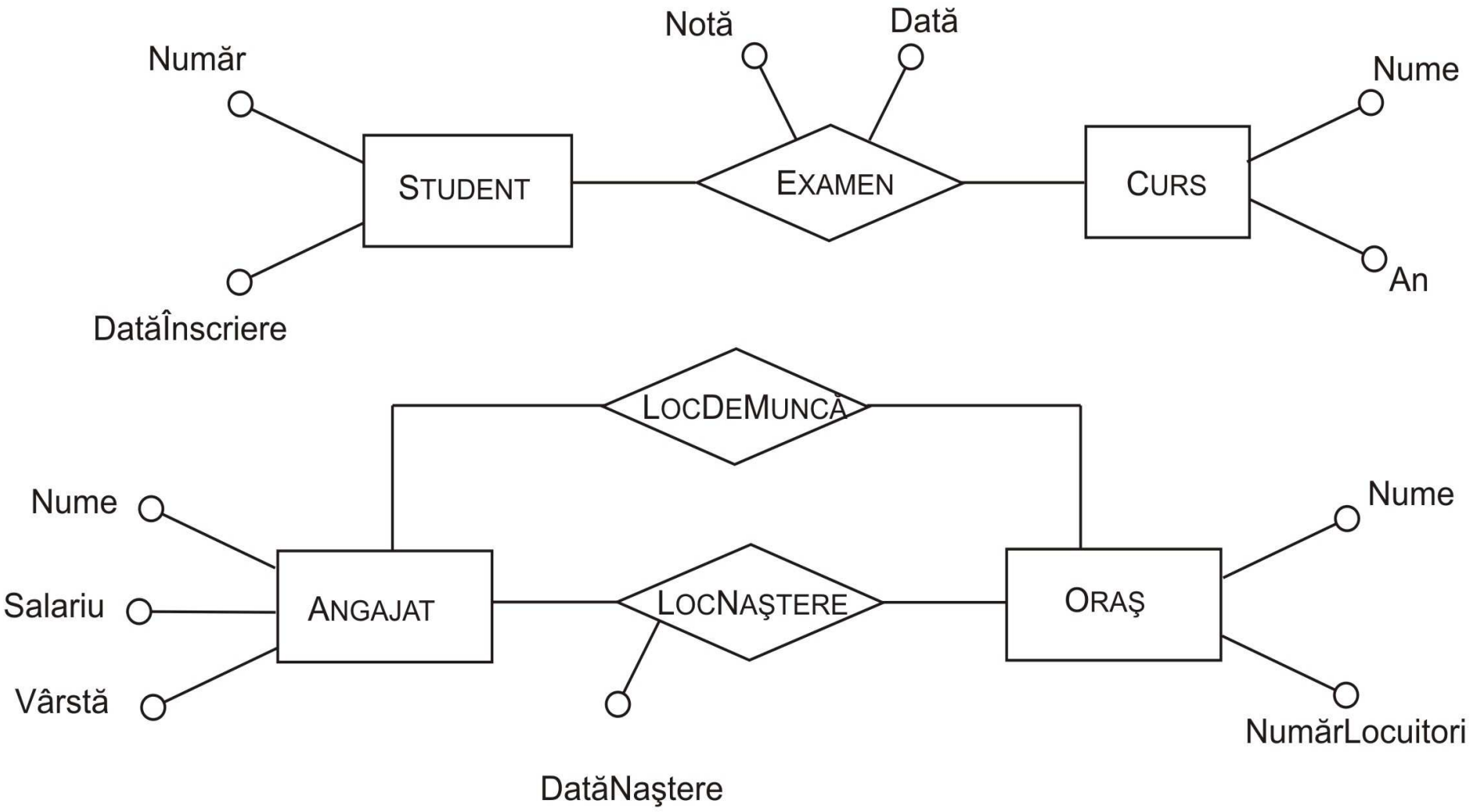


Figura 9. Schemă E-R cu relaţii, entităţi şi atribute

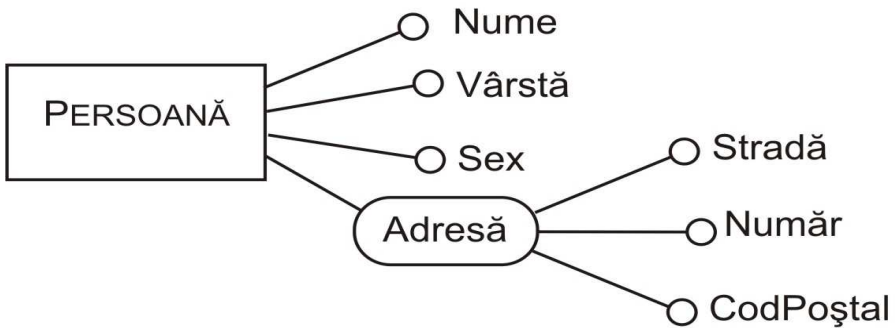


Figura 10. Exemplu de entitate cu un atribut compus

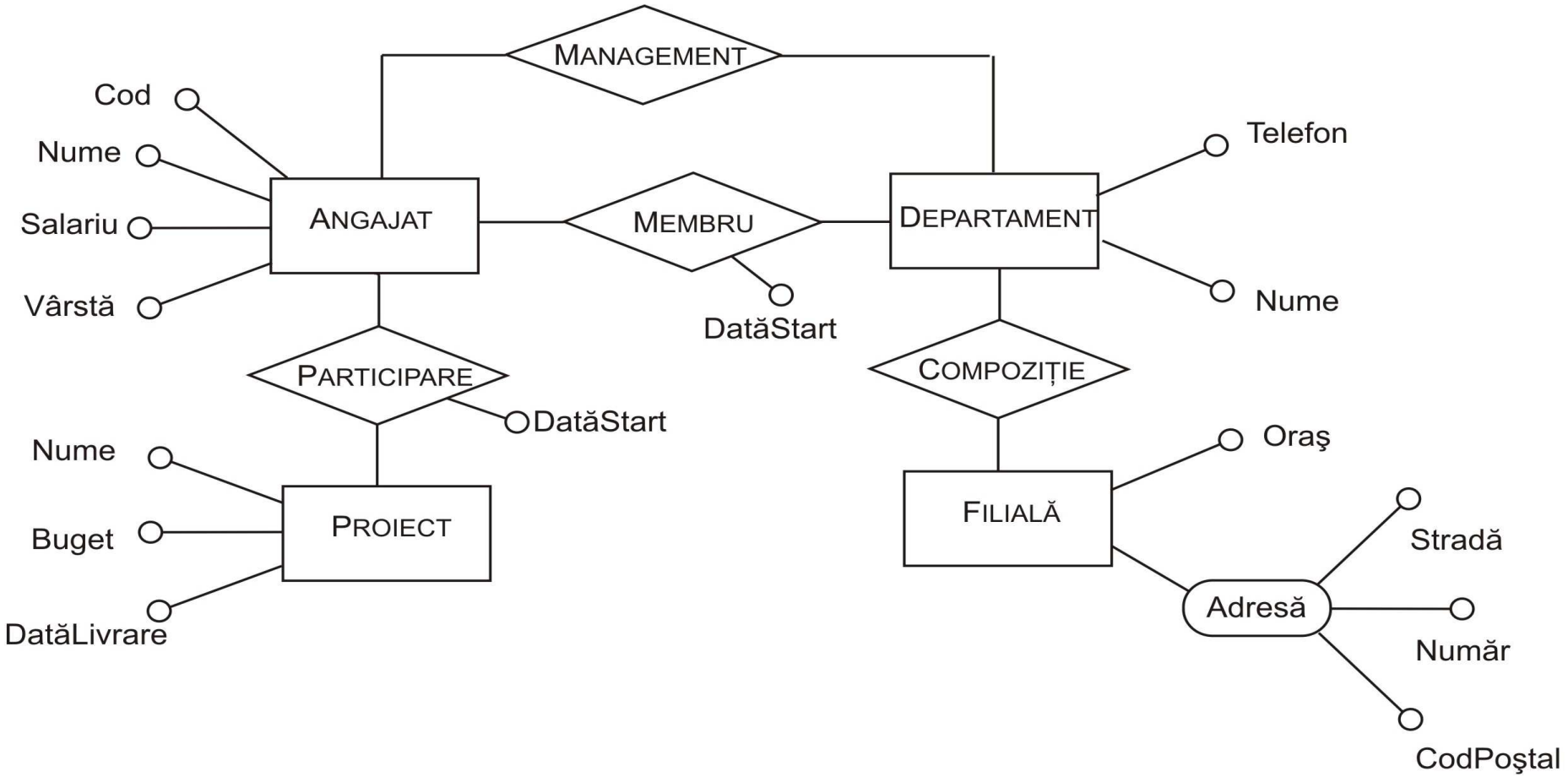


Figura 11. Exemplu de schemă Entitate-Relaţie

Cardinalitatea unei relaţii

- este specificată pentru fiecare entitate participantă la relaţie
- descrie numărul minim, respectiv maxim de apariţii ale relaţiei la care poate participa o apariţie a entităţii
- într-o schemă E-R cardinalităţile minime şi maxime ale participărilor entităţilor în relaţii sunt specificate în paranteze

Figura 12. Cardinalitatea unei relaţii într-un model E-R

- în principiu este posibil să atribuim orice valoare întreagă mai mare ca 0 cardinalităţii unei relaţii, singura cerinţă fiind ca valoarea minimă a cardinalităţii să fie mai mică decât cea maximă

- în majoritatea cazurilor este suficientă utilizarea a trei valori: 0, 1 sau N (N este o valoare întreagă mai mare ca 1):

  - *cardinalitatea minimă* 0 – participarea la relaţie este *opţională*
  - *cardinalitatea minimă* 1 – participarea la relaţie este *obligatorie*
  - *cardinalitatea maximă* 1 – fiecare apariţie a entităţii este asociată cel mult unei singure apariţii a relaţiei
  - *cardinalitatea maximă* N – fiecare apariţie a entităţii este asociată unui număr arbitrar de apariţii ale relaţiei

  - în funcţie de cardinalităţile maxime ale entităţilor implicate într-o relaţie, relaţiile se împart în:
    - *relaţii unu-la-unu*
    - *relaţii unu-la-mai-mulţi*
    - *relaţii mai mulţi-la-mai mulţi*

- ***relaţii unu-la-unu*** - definesc o corespondenţă unu la unu între entităţi

- ***relaţii unu-la-mai-mulţi*** - cardinalitatea maximă a unei entităţi este 1, iar cardinalitatea maximă a celeilalte entităţi este N

- ***relaţii mai mulţi-la-mai mulţi*** - cardinalităţile maxime sunt N pentru ambele entităţi

Observaţii

- este rar cazul în care participarea este obligatorie pentru toate entităţile implicate;
  - motiv: când se adaugă o nouă apariţie a entităţii, în general apariţiile corespunzătoare ale altor entităţi legate de aceasta nu sunt cunoscute încă sau nu există

    Exemplu

    Când se primeşte o nouă comandă, nu există încă o factură şi deci nu este posibilă construirea unei apariţii pentru relaţia ONORARE care conţine noua comandă.

- în relaţiile *n-are*, entităţile implicate au aproape întotdeauna cardinalitatea maximă egală cu N

Cardinalitatea atributelor

- descrie numărul minim, respectiv maxim de valori ale atributelor asociate fiecărei apariţii a unei entităţi sau relaţii.
- în general, cardinalitatea unui atribut este (1,1) şi este omisă pe schemă. În acest caz atributul este o funcţie ce asociază o singură valoare fiecărei apariţii a entităţii (relaţiei).
- valoarea unor atribute poate fi null sau pot exista valori diferite ale atributului asociate unei apariţii a entităţii

  Aceste situaţii pot fi reprezentate prin alocarea cardinalităţii minime egală cu 0 (în primul caz) respectiv cu cardinalitatea egală cu N (în al doilea caz).

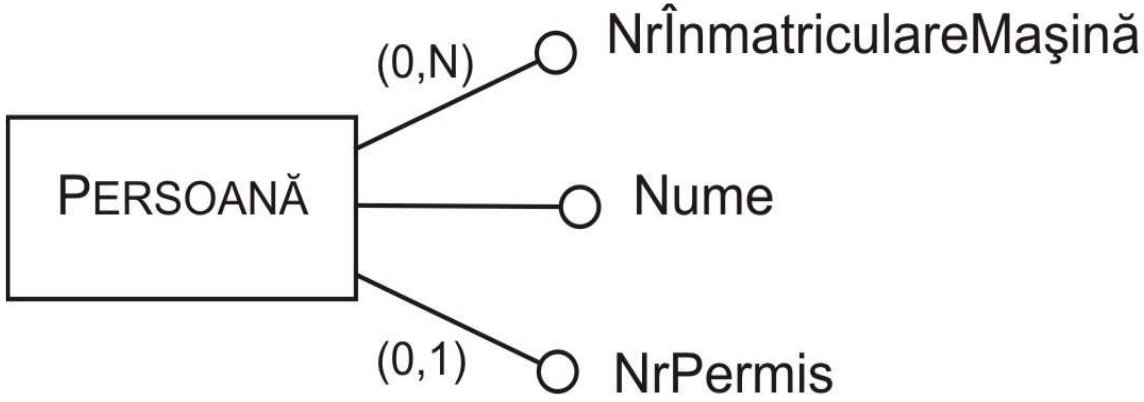


Figura 14. Exemplu de atribute cu cardinalitate

- Într-un mod similar participării unei apariţii a unei entităţi într-o relaţie, putem spune că:

  - un atribut cu cardinalitatea minimă egală cu zero este *opţional* pentru entitatea asociată (sau relaţia asociată)

  - un atribut este *obligatoriu* în cazul în care cardinalitatea minimă este unu.

  - un atribut este *multivaloare* dacă are cardinalitatea maximă N.

    Atributele multivaloare trebuie folosite cu precauţie deoarece ele reprezintă situaţii care pot fi modelate, câteodată, prin entităţi adiţionale legate prin relaţii *unu-la-unu* sau *mai mulţi-la-mai mulţi* cu entităţile la care se referă.

    Exemplu: Să presupunem că avem atributul multivaloare *Calificări* pentru entitatea PERSOANĂ (o persoană poate avea mai multe calificări).

    Calificarea este un concept ce poate fi atribuit mai multor persoane $\Rightarrow$ este naturală modelarea acestui concept cu ajutorul unei entităţi CALIFICARE legată de PERSOANĂ cu o relaţie *mai mulţi-la-mai mulţi*.

Identificatori - sunt specifici fiecărei entităţi din schemă şi descriu conceptele (atributele şi/sau entităţile) schemei ce permit identificarea unică a aparţiilor acelei entităţi

Identificatorii se clasifică în:

- *identificator intern* - format din unul sau mai multe atribute ale entităţii - este cunoscut sub numele de *cheie*

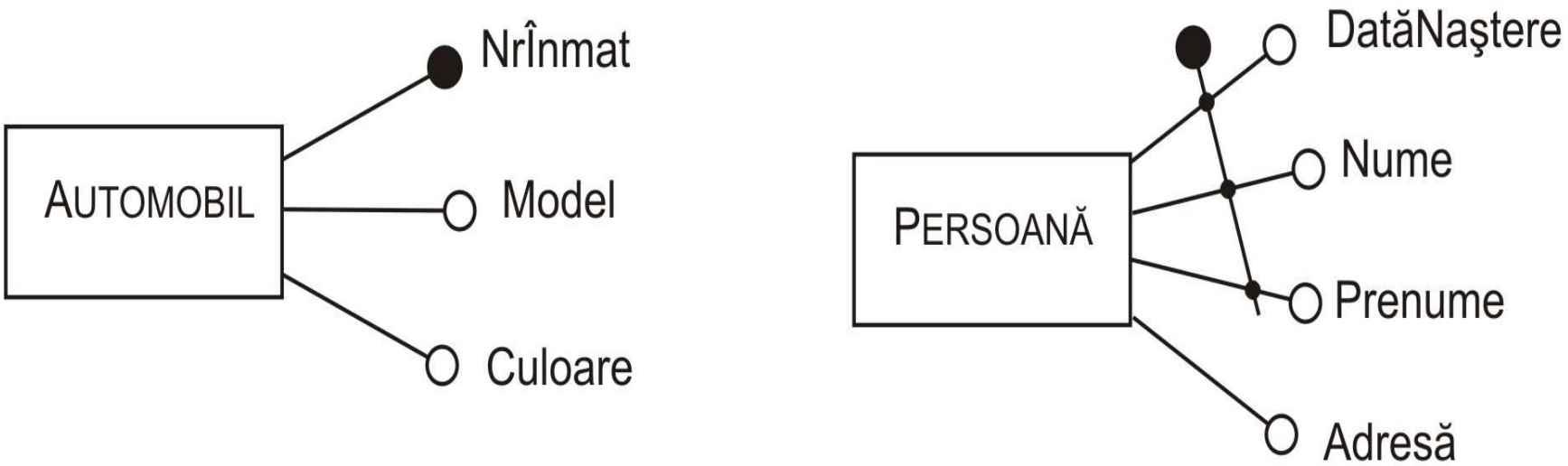
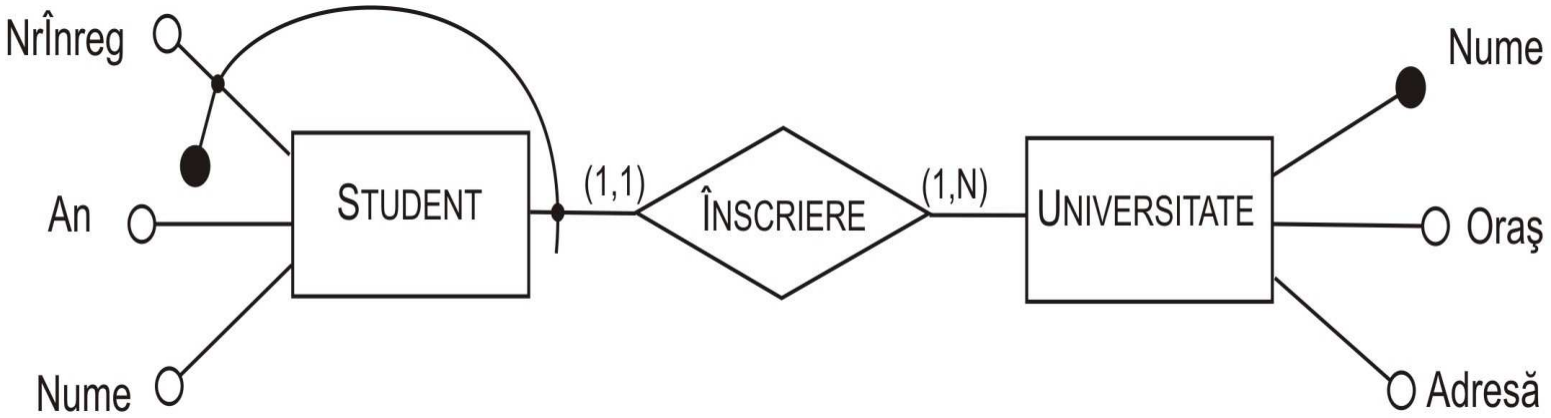


Figura 15. Exemple de identificatori interni

- *identificator extern*. Există cazuri când atribute ale unei entităţi nu sunt suficiente pentru a identifica în mod unic apariţiile entităţii.



  - Schema descrie studenţii înscrişi la diverse universităţi, doi studenţi din universităţi diferite putând avea acelaşi număr de înregistrare $\Rightarrow$ pentru a putea identifica un student în mod unic avem nevoie atât de numărul său de înregistrare, cât şi de universitatea de care acesta aparţine.
  - Un identificator corect pentru entitatea student este format din atributul *NrÎnreg* şi entitatea UNIVERSITATE - ***identificator extern***.
    - Se observă că identificarea este posibilă prin relaţia obligatorie *unu-la-mai mulţi* dintre entităţile UNIVERSITATE şi STUDENT, care asociază fiecare student cu o singură universitate.
  - **O entitate E poate fi identificată prin alte entităţi doar dacă fiecare astfel de entitate este implicată într-o relaţie în care E participă cu cardinalitatea (1,1)**

Observaţii

- un identificator poate implica unul sau mai multe atribute, cu condiţia ca fiecare dintre ele să aibă cardinalitate (1,1);

- un identificator extern poate implica una sau mai multe entităţi, cu condiţia ca fiecare dintre ele să fie într-o relaţie în care entitatea de identificat participă cu cardinalitatea (1,1);

- un identificator extern poate implica o entitate care este la rândul său identificată extern, atâta timp cât nu se generează cicluri;

- fiecare entitate trebuie să aibă un identificator (intern sau extern) dar poate avea mai mult de unul;

  - dacă există mai mult de un identificator atunci atributele şi entităţile implicate într-o identificare pot fi opţionale (cardinalitatea minimă egală cu zero).

În acest moment schema din figura 11 poate fi reexaminată, introducând cardinalităţi şi identificatori.

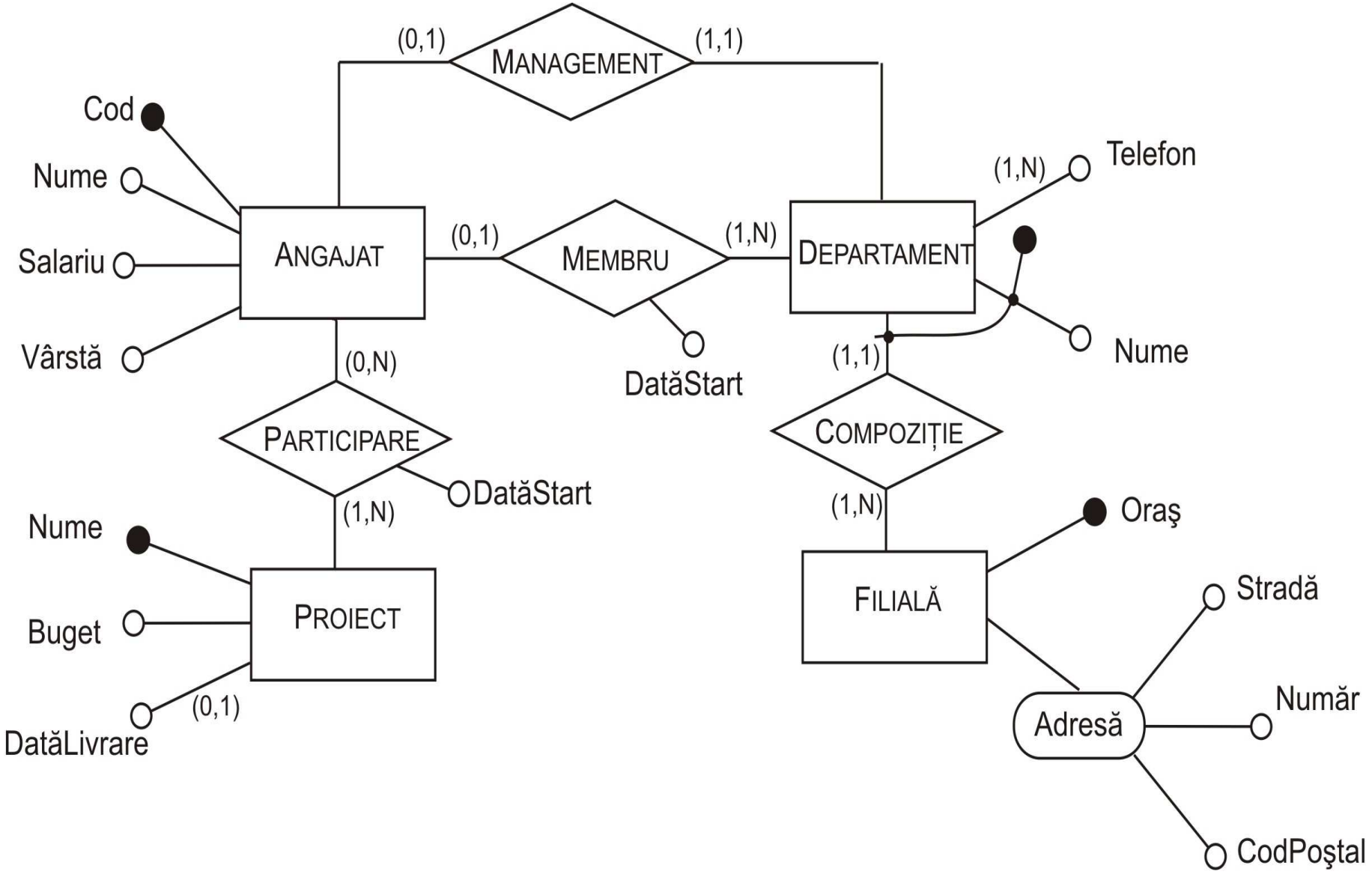


Figura 17. Schema din figura 11 completată cu identificatori şi cardinalităţi

Generalizări

- reprezintă legături logice între o entitate părinte E şi una sau mai multe entităţi copii, E_1, ..., E_n
- entitatea E este mai generală, în sensul că E_1, ..., E_n sunt cazuri particulare ale lui E $\Rightarrow$ E este o generalizare a lui E_1, ..., E_n, iar E_1, ..., E_n sunt particularizări ale entităţii E.
- Exemplu: PERSOANĂ este o generalizare pentru BĂRBAT şi FEMEIE

Proprietăţi

- fiecare apariţie a unei entităţi copil este apariţie a entităţii părinte
- fiecare proprietate a entităţii părinte (atribut, identificator, relaţie, generalizare) este de asemenea o proprietate a entităţii copil

  Exemplu: dacă entitatea PERSOANĂ are atributele Nume şi Vârstă, atunci entităţile BĂRBAT şi FEMEIE au de asemenea aceste atribute. Mai mult, identificatorul pentru PERSOANĂ este de asemenea un identificator valid pentru entităţile BĂRBAT şi FEMEIE. Această proprietate se numeşte ***moştenire***.

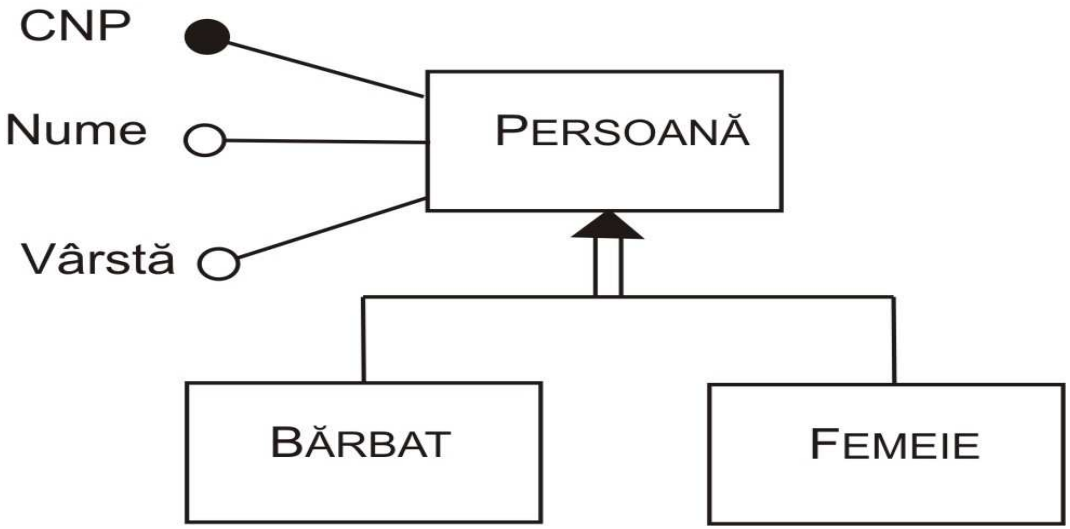


Figura 18. Exemplu de generalizare

Generalizările pot fi clasificate în modul următor:

- *totală* - fiecare apariţie a entităţii părinte este de asemenea o apariţie a unei entităţi copil; în caz contrar generalizarea este *parţială*.

- *exclusivă* - fiecare apariţie a entităţii părinte este cel mult o apariţie a unei entităţi copil; în caz contrar generalizarea este *suprapusă*.

- *Exemple*
  - Generalizarea PERSOANĂ pentru BĂRBAT şi FEMEIE este totală (persoanele pot fi numai bărbaţi sau femei) şi exclusivă (o persoană este fie bărbat fie femeie).
  - Generalizarea VEHICUL pentru AUTOMOBIL şi BICICLETĂ este parţială (există şi alte tipuri de vehicule) şi exclusivă.
  - Generalizarea PERSOANĂ pentru STUDENT şi ANGAJAT este parţială şi suprapusă (există studenţi care sunt şi angajaţi).

- Generalizarea suprapusă poate fi transformată uşor într-o generalizare exclusivă prin adăugare uneia sau mai multor entităţi copil pentru reprezentarea entităţilor ce sunt „intersecţia" între entităţile ce se suprapun.

  În ultimul exemplu prezentat se poate adăuga entitatea STUDENTANGAJAT pentru a obţine o generalizare exclusivă.

- În general, o entitate poate fi implicată în mai multe generalizări diferite.
- Există generalizări cu mai multe nivele, cunoscute sub numele de *ierarhii*.
- De asemenea, o generalizare poate avea un singură entitate copil, cunoscută sub numele de *submulţime*.

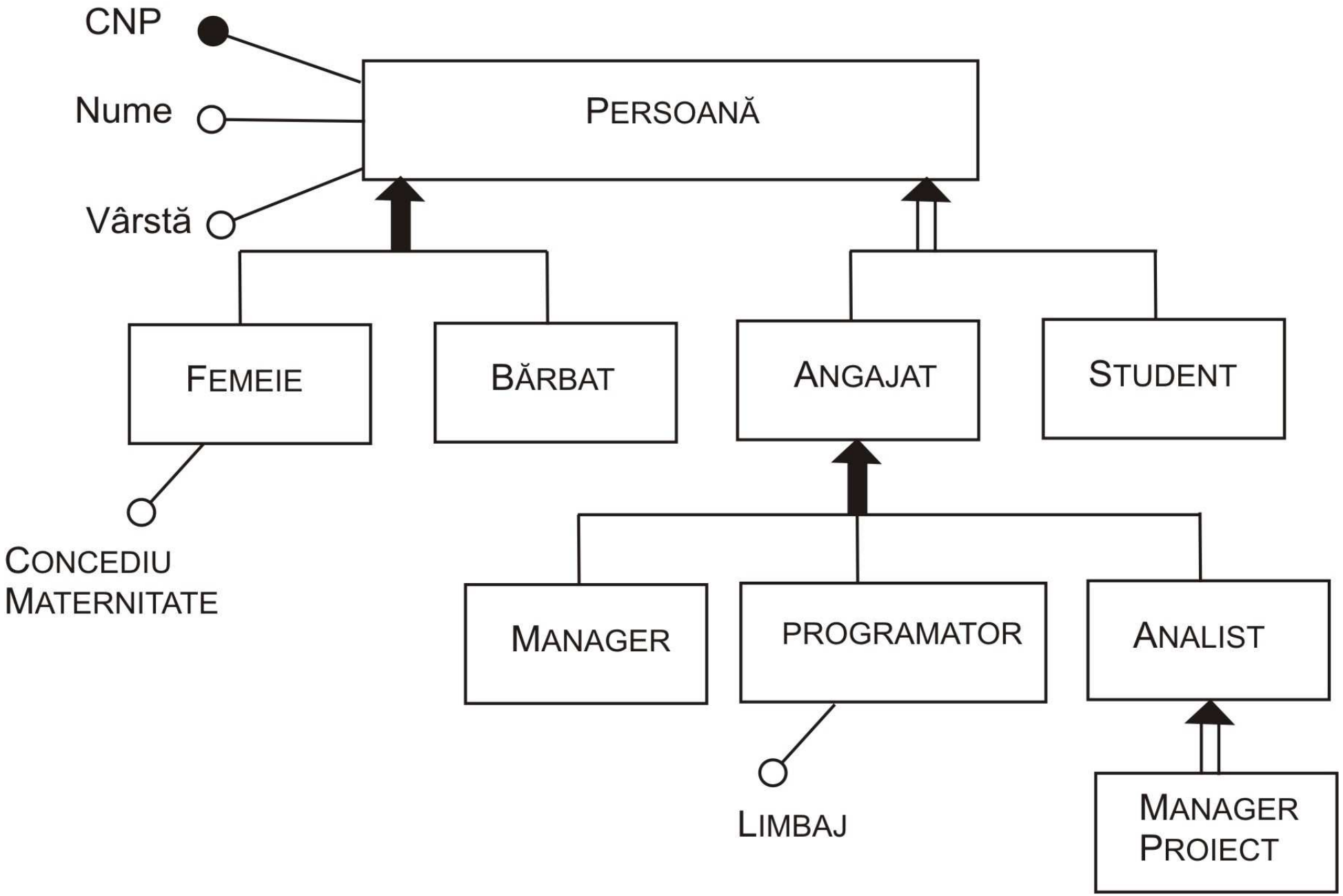


Figura 19. Exemplu de ierarhie a generalizărilor între entităţi

Observaţii finale asupra modelului E-R

Modelul E-R este realizat pe baza a două construcţii de bază: entitate şi relaţie.

- o entitate poate participa în mai multe relaţii sau în nici una;
- o relaţie implică două sau mai multe entităţi;
- participarea unei entităţi într-o relaţie are o cardinalitate minimă şi una maximă.

Modelul E-R mai are construcţiile atribut şi generalizare.

- un atribut are un nume şi o cardinalitate minimă şi una maximă şi aparţine unui concept de bază (entitate sau relaţie);
- atributele compuse sunt specializări ale atributelor şi sunt formate din unul sau mai multe atribute;
- o generalizare are o entitate părinte şi una (cazul submulţimilor) sau mai multe entităţi copil;
- o entitate poate fi părinte sau copil în mai multe generalizări (de asemenea în nici una);

Este esenţială folosirea de nume diferite în cazul construcţiilor de bază pentru a evita ambiguităţile. Două atribute pot avea acelaşi nume dacă aparţin unor construcţii de bază diferite

Există anumite *restricţii* în folosirea unor construcţii:

- **o ierarhie a unei generalizări nu poate conţine cicluri.**
- **cardinalitatea minimă trebuie să fie mai mică decât cea maximă.**

Schemele E-R furnizează o reprezentare abstractă a datelor unei aplicaţii şi pot fi utilizate nu numai pentru proiectarea bazelor de date:

- se pot folosi pentru documentaţii deoarece pot fi înţelese uşor de către nespecialişti;
- pot fi utilizate pentru descrierea datelor dintr-un sistem informaţional existent deja (spre exemplu pentru integrarea cu alte baze de date);
- se pot folosi în cazul modificării cerinţelor clienţilor.

6.3 Documentaţia pentru schemele E-R

- O schemă E-R este adeseori insuficientă pentru a descrie toate aspectele unei aplicaţii în detaliu.

  - În primul rând într-o schemă E-R sunt precizate doar numele conceptelor, acest fapt fiind insuficient pentru a explica semnificaţia acestora

    Spre exemplu, în figura 17, nu este clar dacă entitatea PROIECT se referă la un proiect intern al companiei sau la un proiect extern la care compania respectivă este parte.

  - Mai mult, dacă schema este complexă, se poate întâmpla să nu poată fi reprezentate toate proprietăţile conceptelor care apar într-un mod inteligibil.

    În schema din figura 17 ar fi greu de introdus şi alte atribute pentru ANGAJAT fără a reduce accesibilitatea schemei.

- Pe de altă parte, este imposibilă reprezentarea unor proprietăţi ale datelor prin intermediul schemelor E-R.

  Exemple:

  - un angajat poate fi manager doar în departamentul de care aparţine – nu se pot corela două relaţii
  - un angajat nu poate avea un salariu mai mare decât managerul departamentului de care aparţine.

  Ambele proprietăţi menţionate sunt de tipul constrângere de integritate.

Modelul E-R nu furnizează mijloace potrivite pentru reprezentarea constrângerilor complexe impuse datelor.

- Din motivele precizate mai sus, o schemă E-R trebuie însoţită de o documentaţie care:
  - facilitează interpretarea schemei
  - permite descrierea proprietăţilor care nu pot fi exprimate direct prin construcţiile puse la dispoziţie de modelul E-R.

Reguli de operare

Regulile de operare - unelte utilizate de analişti pentru a descrie proprietăţile unei aplicaţii care nu pot fi exprimate direct cu ajutorul modelului conceptual.

Această abordare permite specificarea regulilor unei aplicaţii.

Exemplu de regulă de operare: un angajat nu poate câştiga mai mult decât managerul său.

O regulă de operare poate fi o:

- *descriere a unui concept* relevant al aplicaţiei - o definire precisă a entităţilor, atributelor sau relaţiilor unui model E-R;
- *constrângere de integritate* aplicată datelor aplicaţiei;
- *derivare* - un concept care poate fi obţinut pe baza unei deducţii sau a unui calcul matematic din alte concepte ale schemei

  Exemplu: atributul *Cost* poate fi obţinut ca sumă a atributelor *Net* şi *Taxe*.

Regulile pentru descrierea conceptelor se exprimă în general în limbaj natural.

Regulile ce descriu constrângeri de integritate şi derivări folosesc **definiţii formale**.

O notaţie de tipul if <*condiţie*> then <*acţiune*> nu este potrivită pentru a exprima o regulă.

O structură mai potrivită pentru a exprima o regulă de operare sub forma unei aserţii poate fi:

<*concept*> trebuie/nu trebuie <*expresie a conceptelor*>

unde conceptele pot corespunde:

- unui concept a schemei E-R la care se referă
- unui concept derivat.

Exemplu: pentru a exprima constrângerile pentru schema din figura 17, se utilizează următoarele reguli de operare:

(RO1) *managerul departamentului* trebuie *să aparţină departamentului*;

(RO2) *un angajat* nu trebuie *să aibă salariul mai mare decât managerul departamentului căruia îi aparţine*;

(RO3) *un departament din filială Iaşi* trebuie *manageriat de un angajat cu mai mult de 10 ani vechime în companie*

Regulile de operare care descriu derivări pot fi exprimate prin specificarea operaţiilor (matematice sau de alt fel) care permit obţinerea conceptelor derivate.

O structură posibilă este următoarea:

<*concept*> este obţinut prin <*operaţii asupra conceptelor*>

Dacă în exemplul tratat până acum entitatea DEPARTAMENT are un atribut NumărDeAngajaţi, se poate introduce o regulă de forma:

(RO4) *numărul angajaţilor dintr-un departament* este obţinut prin *numărarea angajaţilor care aparţin departamentului respectiv*;

Tehnici de realizare a documentaţiei

Documentaţia pentru conceptele variate reprezentate într-o schemă pot fi organizate uşor sub forma unui *dicţionar de date*.

Acesta este format din două tabele:

- primul tabel descrie entităţile din schemă: numele lor, definiţii informale în limbaj natural, lista tuturor atributelor (cu o descriere a acestora) şi identificatorii posibili;
- al doilea tabel descrie relaţiile: numele lor, descriere informală, lista atributelor (cu descrieri posibile) şi lista entităţilor implicate împreună cu cardinalităţile de participare la relaţie.

Observaţii:

- Dicţionarul de date poate fi utilizat şi pentru documentarea unor constrângeri impuse datelor şi altor forme de reguli de operare.
- Se mai poate alcătui un tabel în care se listează regulile organizate după tip. Unele reguli pot fi exprimate în formele precizate în secţiunea anterioară.
- Este importantă reprezentarea tuturor regulilor care descriu constrângeri ce nu sunt exprimate în schemă.

- poate fi de asemenea folositoare reprezentarea regulilor deja reprezentate în schemă.

Entitate	**Descriere**	**Atribute**	**Identificator**
ANGAJAT	Angajaţi care lucrează în companie	CNP, Nume, Prenume, Vârstă	CNP
PROIECT	Proiectele companiei la care lucrează angajaţii	Nume, Buget, DatăLivrare	Nume
DEPARTAMENT	Departamentele filialei companiei	Telefon, Nume	Nume, FILIALĂ
FILIALĂ	Filiala companiei într-un oraş	Oraş, Adresă (Număr, Stradă, CodPoştal)	Oraş

Relaţie	**Descriere**	**Entităţi implicate**	**Atribute**
MANAGEMENT	Asociază un manager cu un departament	Angajat (0,1) Departament (1,1)	
MEMBRU	Asociază un angajat cu un departament	Angajat (0,1) Departament (1,N)	DatăStart
PARTICIPARE	Asociază angajaţii cu proiectele	Angajat (0,N) Proiect (1,N)	DatăStart
COMPOZIŢIE	Asociază un departament cu o filială	Departament (1,1) Filială (1,N)	

Figura 20. Dicţionarul de date pentru schema din figura 17

Constrângeri
(RO1) *managerul departamentului* trebuie *să aparţină departamentului*;
(RO2) *un angajat* nu trebuie *să aibă salariul mai mare decât managerul departamentului căruia îi aparţine*;
(RO3) *un departament din filială Iaşi* trebuie *manageriat de un angajat cu mai mult de 10 ani vechime în companie*
(RO4) *un angajat care nu aparţine unui departament* nu trebuie *să participe la nici un proiect.*
Derivări
(RO5) *bugetul unui proiect* este obţinut prin *înmulţirea sumei salariilor angajaţilor ce lucrează la el cu 3*

Figura 21. Reguli de operare pentru schema din figura 17

Probleme propuse.

1. Se consideră schema E-R din figura 22.

   a) Corectaţi schema, luând în considerare proprietăţile fundamentale ale generalizărilor.

   b) Schema reprezintă doar femeile care muncesc; modificaţi schema astfel încât să reprezinte toţi muncitorii, bărbaţi şi femei.

   c) Atributul *Judeţ* poate fi privit ca o sub-proprietate a atributului *Ţară*; restructuraţi schema în acest sens.

2. Adăugaţi cardinalităţile minime şi maxime şi identificatorii pentru schema obţinută după rezolvarea problemei 1; specificaţi dacă există constrângeri de integritate pe schemă care nu pot fi exprimate prin modelul entitate-relaţie.

3. Reprezentaţi următoarele informaţii printr-o schemă entitate-relaţie:

   o companie produce CD-uri cu un cod şi un titlu; pe fiecare CD au fost înregistraţi unul sau mai mulţi cântăreţi, fiecare având un nume şi o adresă şi câţiva dintre aceştia şi un nume de scenă.

4. Completaţi schema din problema 3 cu informaţii care vi se par că lipsesc.

5. Creaţi o schemă E-R pentru a reprezenta următoarele concepte, utilizând, dacă este cazul, construcţii de tip generalizare. Indicaţi atributele entităţilor implicate şi tipul generalizărilor, rezolvând eventualele suprapuneri.

Angajaţii unei companii se împart în manageri, programatori, analişti, şefi de proiect şi secretare. Există analişti care sunt de asemenea programatori. Şefii de proiect trebuie să fie manageri. Fiecare angajat are un cod, nume şi prenume. Fiecare categorie de angajaţi are un salariu de bază. Fiecare angajat (în afară de manageri) are fixat un număr de ore de muncă.

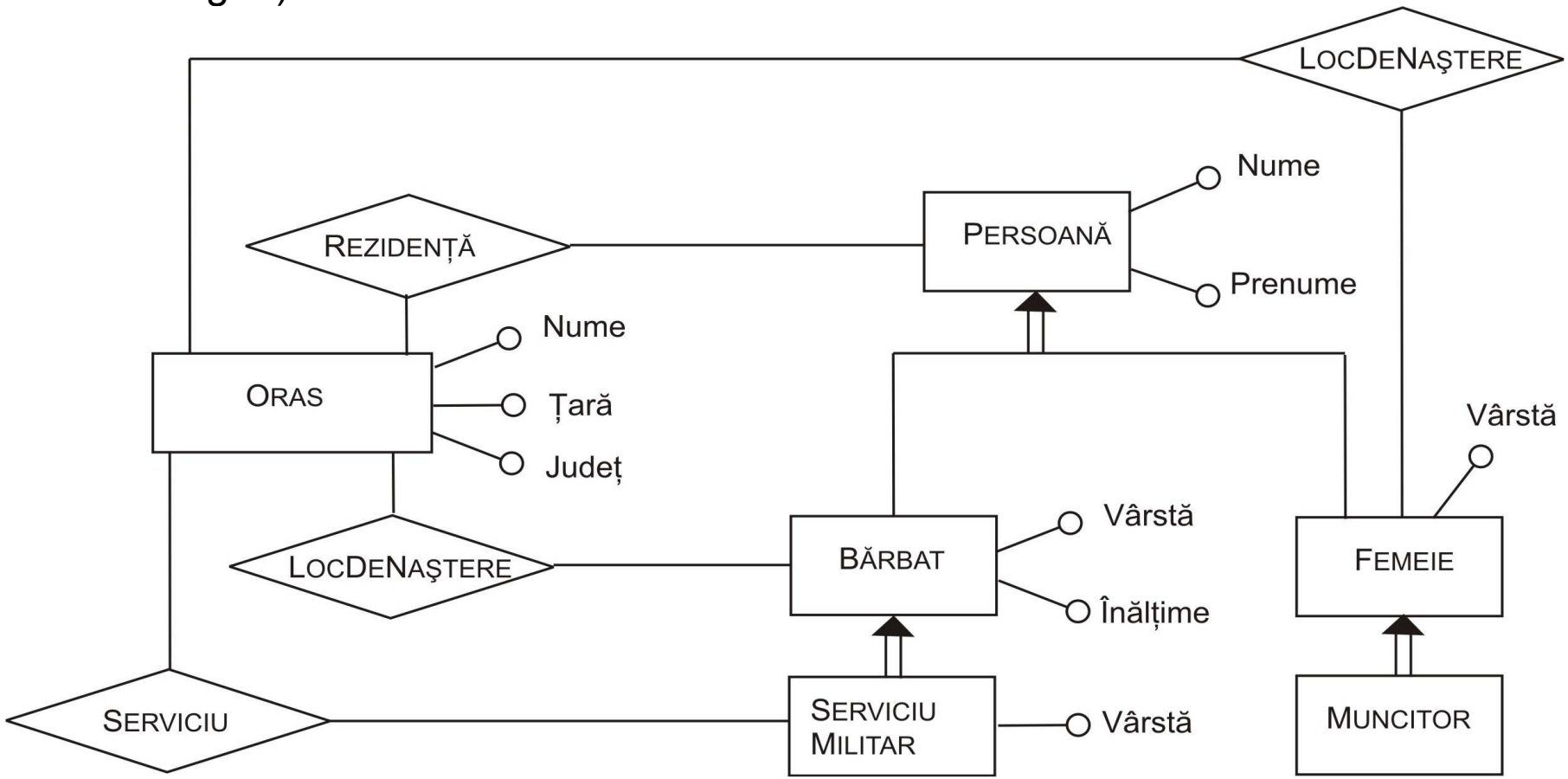


Figura 22. Schema E-R pentru problema 1