

# Ghid de proiectare a bazelor de date relaționale

## 0. Introducere

Proiectarea bazei de date este o muncă de colectiv care armonizează cerințele și posibilitățile beneficiarului pe de o parte și proiectantului de sistem pe de altă parte. Baza de date este prima treaptă spre o viziune sistemică, de ansamblu, unificatoare și generatoare de rezultate specifice corecte în același timp.

Vechiul sistem de proiectare a fost bazat pe fișiere. Limitele acestei concepții constau în separarea datelor, duplicarea datelor, dependența programului de concepția fișierului și incompatibilitatea formatelor de fișiere folosite în diverse limbaje. Aceste limitări implică pierdere de timp și spațiu și posibilitatea de inconsistență a datelor.

Sistemul de bază de date - în opoziție cu vechiul sistem - dă posibilitatea de a defini datele în afara programului și asigură controlul asupra manipulării datelor.

**Definiție: Bază de date:** Este o colecție partajată de date legate logic, proiectată pentru a satisface necesitățile unui sistem informatic.

Deci datele sunt strânse într-o colecție unică și sunt folosite simultan de mai mulți utilizatori. Redundanța datelor este controlată prin normalizare, ceea ce implică o redundanță minimă.

O astfel de bază de date are nevoie de un sistem de gestiune a bazei de date. Acesta este un sistem de programe care fac posibilă definirea, întreținerea și accesul controlat la baza de date. Un astfel de sistem trebuie să conțină limbajul de definire și limbajul de manipulare a datelor. Putem aminti următoarele limbaje: SQL sau QBE (limbaje generale), respectiv DBASE, FOX PRO, PROGRESS, PARADOX ș.a.m.d. (limbaje specifice).

Cum se stabilește structura bazei de date? Tocmai prin proiectarea de care ne ocupăm. În opoziție cu proiectarea bazei de date bazate pe fișiere, care pornea de la una sau mai multe aplicații ale beneficiarului, proiectarea bazei de date se rezolvă înaintea elaborării aplicației. De aceea această acțiune devine esențială pentru tot sistemul.

Proiectantul bazei de date trebuie să identifice datele relaționale, relațiile dintre ele și restricțiile asupra lor. Proiectarea constă din două faze: unul logic și unul fizic. În timpul proiectării logice este importantă implicarea viitorilor utilizatori în procesul de proiectare. În proiectarea fizică se decide cum va fi realizat practic modelul logic.

Avantajele și dezavantajele sistemului de baze de date:

Avantajele ar fi următoarele:

- controlul redundanței
- consistența datelor
- economia de spațiu pentru aceleași date
- controlul integrității datelor
- utilizarea standardelor
- dă posibilitatea răspunsului la cereri variate și cu exprimări parțial necunoscute la momentul proiectării.
- productivitate crescută
- concurență crescută
- posibilități crescute de recuperare în caz de eroare

Dezavantaje:

- complexitate crescută
- costul SGBD
- cost crescut rezultat din cerințe de hard
- costul trecerii de la un sistem la altul
- o eventuală defecțiune are un impact crescut, global

# 1. Modelarea ER (Entity-Relationship)

În acest capitol vom descrie următoarele obiective:

- Folosirea modelului conceptual de nivel înalt pentru proiectarea de baze de date.
- Conceptele modelului ER, model de nivel înalt.
- O tehnică grafică de reprezentare a modelului ER.
- Modul de identificare a problemelor ce pot apărea la crearea unui model ER.
- Limitele modelului ER primar și conceptele necesare pentru modelarea aplicațiilor complexe.
- Conceptele modelului EER (Enhanced Entity-Relationship), numite și specializare/generalizare.
- O tehnică grafică de reprezentare a specializării/generalizării în modelul EER.
- Utilizarea produsului ERwin produs de Logic Works, pentru proiectarea bazelor de date, ceea ce include și modelul ER.

Modelul ER (Entity-Relationship) este un model conceptual de nivel înalt dezvoltat de Chen în 1976, care facilitează crearea de baze de date relaționale. În acest capitol, vom descrie conceptele primare ale modelului ER, după care vom identifica problemele care pot apărea la un model ER (Howe, 1989). Vom discuta deasemenea, problemele reprezentării unor aplicații, folosind modelul ER (Schmidt și Swenson, 1975). Având în vedere că modelul ER are anumite limite în modelarea aplicațiilor, vom introduce un model mai complex, modelul EER, numit și specializare/generalizare.

## 1.1. Conceptele modelului Entity-Relationship

Conceptele primare ale modelului ER includ : **tip de entitate, tip de relație, atribut.**

### 1.1.1. Tipuri de entități

Definiție: **Tip de entitate:** Este un obiect sau un concept, identificat de o întreprindere, având o existență independentă.

Tipurile de entități reprezintă obiecte reale, din viața de zi cu zi, având proprietățile lor, sau obiecte conceptuale, abstracte.

Definiție: **Entitate:** Un obiect sau un concept ce se poate identifica unic.  
Un tip de entitate conține mai multe entități.

Un tip de entitate se identifică prin nume și listă de atribute. O bază de date conține în general mai multe tipuri de entități. Fiecare tip de entitate are propriile lui atribute. Putem clasifica aceste tipuri în tipuri slabe și tipuri tari.

Definiție: **Tip slab de entitate:** Este un tip de entitate, a cărui existență este dependentă de un alt tip de entitate.

Definiție: **Tip tare de entitate:** Este un tip de entitate, a cărei existență nu depinde de nici un alt tip de entitate.  
Entitățile slabe se mai numesc dependente sau cubordonate iar cele tari părinte sau dominante.

Reprezentarea grafică a entităților: **Entitățile tari se reprezintă printr-un dreptunghi, etichetate cu numele entității** iar cele slabe se reprezintă printr-un dreptunghi desenat cu linie dublă, etichetate deasemenea cu numele lor.

Exemplu: 

nume entitate tare
--------------------

nume entitate slabă
---------------------

### 1.1.2. Atribute

Definiție: **Atribute:** Proprietățile unui tip de entitate sau de relație.

Definiție: **Domeniul atributului:** Un set de valori ce se pot da acelui atribut.  
Domeniul unui atribut nu se poate defini totdeauna foarte exact. De exemplu, atributul *nume de familie* poate lua orice nume de familie existentă. Evident, acest atribut trebuie să fie un șir de caractere, dar oare ce caractere poate să conțină? Unele domenii se pot descompune în mai multe subdomenii. De exemplu *data nașterii* se poate descompune în subdomeniile: *an, lună, zi*.

Atributele se pot clasifica în *simple* sau *compuse*; cu o *singură valoare* sau cu *mai multe valori*; respectiv *derivate*.

Definiție: **Atribut simplu:** Atribut care are doar o singură componentă și o existență independentă.

Aceste atribute nu se pot diviza mai târziu în mai multe atribute distincte.

Definiție: **Atribut compus:** Atribut care are mai multe componente și o existență independentă.

Aceste atribute se pot diviza în mai multe atribute simple. De exemplu atributul **adresă se poate** descompune în atributele: *strada, număr, oraș, cod poștal și județ*. Decizia ca un atribut compus să se descompună în mai multe atribute simple este **dependentă de modul în care se va utiliza acel atribut: separat pe componente, sau întregul atribut.**

Definiție: **Atribut cu o singură valoare:** Atribut care poate lua o singură valoare pentru fiecare entitate.

Majoritatea atributelor sunt atribute cu o singură valoare, ceea ce este indicat în proiectarea bazelor de date.

Definiție: **Atribut cu mai multe valori:** Atribut care poate lua mai multe valori pentru fiecare entitate.

Atributele cu mai multe valori, trebuie să aibă totdeauna o limită inferioară și una superioară.

Definiție: **Atribut derivat:** Atribut a cărei valoare se poate calcula din unul, sau mai multe alte atribute, care nu sunt neapărat atributele entității în cauză.

De exemplu atributul *vârsta* este derivată din atributul *data nașterii* și data zilei în care se utilizează acest atribut. Alt exemplu ar fi atributul *numărul total de entități*, ceea ce se poate calcula, numărând entitățile înregistrate.

### Chei:

Intuitiv, o cheie este un atribut, care determină unic o entitate dintr-un tip de entitate. În continuare, dăm o definiție riguroasă a cheii.

**Definiție: Cheie candidat:** Un atribut, sau un set de attribute, care identifică unic o entitate dintr-un tip de entitate.

**Definiție: Cheie primară:** O entitate poate să aibă una sau mai multe chei candidat, din care doar una selectată este și primară.

Decizia referitoare la care din chei candidate să fie cheie primară este dependentă de lungimea cheii. Cheia primară este de obicei cea mai scurtă dintre cheile candidat.

**Definiție: Cheie compusă:** O cheie candidat care conține cel puțin două attribute.

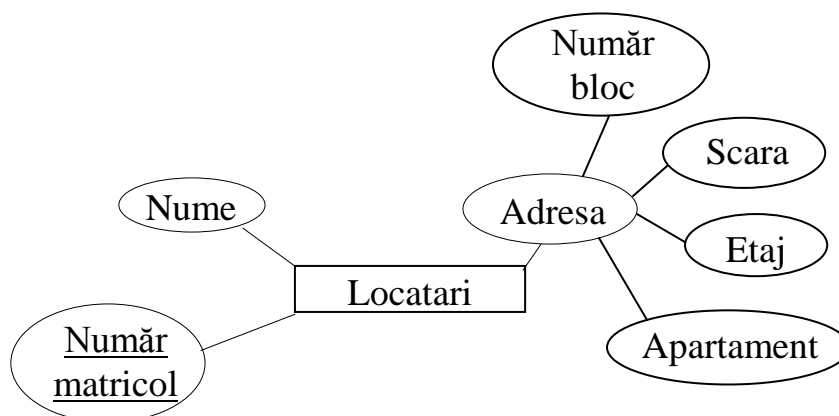
### Reprezentarea grafică a atributelor:

Un atribut se reprezintă printr-o elipsă, legată de entitatea de care aparține cu o linie și etichetată cu numele atributului. Elipsa este punctată, dacă atributul este un atribut derivat; respectiv dublată, dacă poate lua mai multe valori.

Dacă un atribut este compus, atunci componentele atributului se reprezintă în elipse legate printr-o linie de atributul compus.

Atributurile care intră în componența cheii primare, se notează prin sublinierea numelui atributului.

### Exemplu:



În acest exemplu, entitatea *Locatari*, fiind o entitate tare, are următoarele attribute: *Număr matricol*, *Nume* și *Adresa*. Dintre aceste attribute, atributul *Număr matricol* este cheie primară; atributul *Adresa* este atribut compus, care se descompune în *Număr bloc*, *Scara*, *Etaj* și *Apartament*.

### 1.1.3. Tipuri de relații

Definiție: **Tip de relație:** Asociere între tipuri de entități.

Definiție: **Relație:** Asociere între entități, când asocierea include un tip de entitate dintre toate tipurile participante.

Reprezentarea grafică a relațiilor: Relațiile se reprezintă printr-un romb etichetat cu numele relației. Rombul se desenează cu linie dublă, dacă relația leagă o entitate slabă de entitatea tare de care aparține.

Definiție: **Gradul relației:** Numărul entităților participante în relație.

Entitățile dintr-o relație se numesc **participanți**. Numărul lor dă gradul relației. Dacă într-o relație sunt doi participanți, atunci relația se numește **binară**.

Definiție: **Relație recursivă:** Relație în care aceleași entități participă în roluri diferite.

În cazul relațiilor recursive cele două arce de la entitate la relație și înapoi, primesc diferite etichete, care sunt importante în înțelegerea corectă a relației.

### 1.1.4. Atributele relațiilor

Atributele descrise mai sus, se pot asocia și relațiilor. Aceste atribute se reprezintă grafic, ca și atributele entităților, cu deosebirea că legătura nu este cu o entitate, ci cu o relație. Adică linia leagă elipsa de romb ce semnifică relația.

## 1.2. Structuralitatea

Să analizăm acum restricțiile ce pot apărea la includerea unei entități participante într-o relație. Avem două tipuri mari de restricții: **cardinalitatea și participarea**.

### 1.2.1. Cardinalitatea

Definiție: **Cardinalul** este numărul relațiilor posibile pentru o entitate participantă.

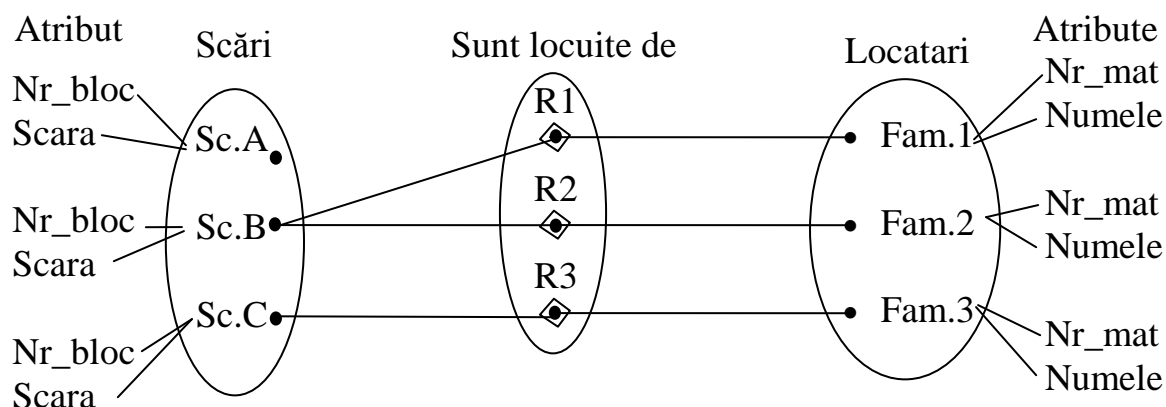
Majoritatea relațiilor au gradul doi, care pot fi: unu-la-unu (1:1), unu-la-multe (1:M), sau multe-la-multe (M:N).

Relațiile unu-la-unu:

În relațiile unu-la-unu, o entitate este legată de cel mult o entitate din partea cealaltă a relației. Această relație se reprezintă grafic prin etichetarea arcului dintre relație și entitate cu cardinalul relației, adică cu 1.

Relațiile unu-la-multe:

Relația de tip unu-la-multe are cardinalul 1 în stânga și N în dreapta. Deci o entitate participantă este legată în relația respectivă de 0, 1, sau mai multe entități. Exemplificăm acest tip de relație prin relația "Sunt locuite" dintre tipurile de entități "Scări", respectiv "Locatari".



**Figura 1.1.** Exemplu de relație 1:M

În reprezentarea grafică a relațiilor, relația unu-la-multe se notează cu etichetarea arcului cu 1 în partea stângă și cu N în partea dreaptă. Din exemplul de mai sus observăm că dacă relația directă este de unu-la-multe, atunci relația inversă este de unu-la-unu.

Relațiile multe-la-multe:

Acest tip de relație se deosebește de relația unu-la-multe prin faptul că relația inversă nu este de unu-la-unu, ci de unu-la-multe. Deci, dacă și relația directă și relația inversă este de tipul unu-la-multe, atunci relația este de tipul multe-la-multe și se notează cu (N:M).

### 1.2.2. Participarea

**Definiție:** **Participarea** determină dacă existența unei entități depinde sau nu de relația cu o altă entitate.

Participarea poate fi de mai multe tipuri: **totală și parțială**. În cazul participării totale, toate entitățile participă în relația dată. În caz contrar participarea se numește parțială. În diagrama ER aceste tipuri de relații se reprezintă prin arc cu linie dublă pentru participarea totală, respectiv cu linie simplă pentru participarea parțială. Pentru participarea parțială, există un mod de notație prin care se etichetează arcele relației cu perechea de numere ce reprezintă minimul, respectiv maximul entităților participante la relație.

## 1.3. Problemele modelului ER

În acest capitol vom examina numeroasele probleme ce pot apărea la modelarea unei baze de date. Aceste probleme se referă la capcanele care pot apărea la definirea relațiilor între tipuri de entități. Amintim două tipuri mari de capcane: de **tip labirint (fan trap)**, respectiv de **tip prăpastie (chasm trap)**.

Definiție: **Fan trap** (capcană de tip labirint): Acest tip de capcană reprezintă cazul în care modelul reprezintă o relație între două tipuri de entități, dar calea dintre cele două membrii este ambiguă.

Cu alte cuvinte, pornind de la o entitate, nu se știe ca după parcurgerea căii relațiilor, la ce entitate se ajunge în partea cealaltă. Această capcană se poate elimina prin rearanjarea ordinii relațiilor dintre cele două tipuri de entități.

Celălalt tip de capcană este:

Definiție: **Chasm trap** (capcană de tip prăpastie): Acest tip de capcană se descrie prin faptul că modelul sugerează existența relației între cele două tipuri de entitate, dar calea dintre tipurile de entități nu există.

Această problemă provine din participarea parțială a unor entități la una dintre relații. Problema se poate rezolva prin introducerea unei relații directe.

#### **1.4. Modelul Enhanced Entity-Relationship (EER)**

Modelul ER, discutat în capitolele de mai sus, este adecvat pentru descrierea multor scheme de baze de date. Din 1980 s-au dezvoltat foarte mult aplicațiile care folosesc baze de date. Modelul ER nu mai era suficient pentru reprezentarea complexelor scheme de baze de date. De aceea s-au introdus alte concepte în modelul ER, dezvoltând modelul EER.

Modelul EER include toate conceptele modelului ER, plus alte completări devenite necesare. Aceste concepte sunt nou introduse sunt: specializarea/generalizarea, categorisirea și încapsularea. În acest capitol vom descrie conceptele de bază a modelului EER și anume, specializare/generalizare.

Specializarea/generalizarea este în strânsă legătură cu modul de descriere a tipurilor de entități în superclase și subclase, precum și de moștenirea atributelor. De aceea vom descrie aceste concepte.

##### **1.4.1. Superclasele și subclasele tipurilor de entități**

Vom descrie aceste concepte, având ca exemplu tipul de entitate “Locatari”.

Definiție: **Superclasă:** Superclasa este un tip de entitate care include subclase distincte reprezentate într-un model de date.

Definiție: **Subclasă :** Subclasa este un tip de entitate care are un rol distinct și este membru al unei superclase.

De exemplu superclasa “Locatari” se divizează în două subclase și anume: “Șef de scară” și “Familii”. Relația dintre o superclasă și o subclasă se numește relație superclasă/subclasă. Această relație este de tip unu-la-unu. Relația “Locatari/Familii” de exemplu, este o astfel de relație.

Fiecare membru al unei subclase este membru și în superclasă, dar are un rol diferit. Există subclase care se suprapun, adică există membrii într-o subclasă care apar și în cealaltă subclasă. Putem observa că exemplul de mai sus este exact de acest tip, pentru că șeful de scară poate să fie și cap de familie.



De ce se folosește clasificarea în subclase? Răspunsul este simplu de observat direct din exemplul dat. Dacă nu am clasifica entitățile din tipul de entitate “Locatari”, atunci ar trebui să memorăm informațiile specifice șefului de scară și capului de familie pentru fiecare locatar. Deoarece majoritatea locatarilor nu aparține niciunei categorii, pentru acestea ar fi informații nefolositoare, dar care însă ar ocupa mult spațiu pe suportul magnetic.

#### 1.4.2. Moștenirea atributelor

Fiecare subclasă are ca atribute comune, atributele superclasei în care aparține. Deci fiecare membru al unei subclase se caracterizează prin atributele superclasei și alte atribute specifice subclasei. De exemplu subclasa *Familii* se caracterizează prin toate atributele superclasei *Locatari* (*Număr matricol*, *Număr bloc*, *Scara*, *Etaj*, *Apartament* și *Nume*), pe lângă care mai are și alte atribute specifice (*Număr persoane*, *Număr persoane prezente*, etc.)

Fiecare subclasă poate să aibă subclase, creîndu-se un arbore de clase, care se numește **ierarhie de tipuri**. Această ierarhie de tipuri include alte denumiri de ierarhii și anume: **ierarhia de specializări** (de exemplu, tipul de entitate *Familii* este o specializare a tipului de entitate *Locatari*) și **ierarhia de generalizări** (de exemplu, *Locatari* este o generalizare a tipului de entitate *Familii*).

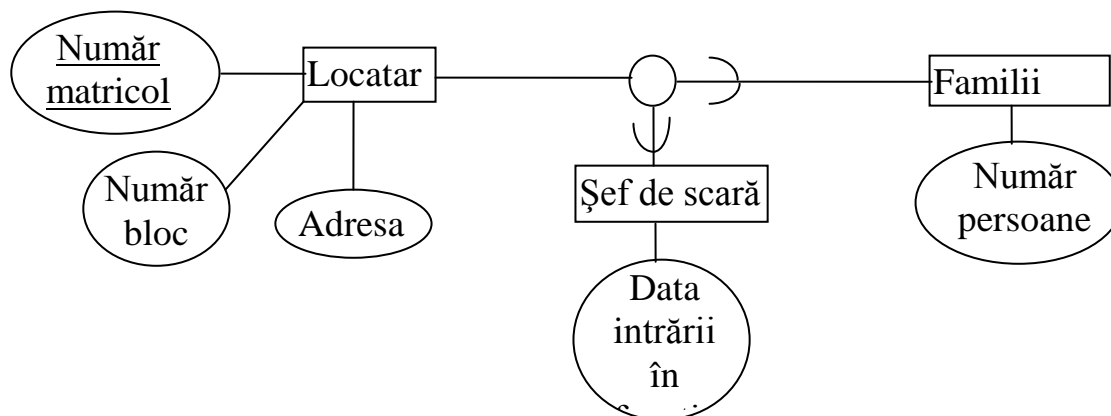
#### 1.4.3. Specializarea

**Definiție: Specializare** este un proces de maximizare a diferențelor unor membrii ale unei entități, identificând caracteristicile distincte.

Specializarea este o aproximare top-down a definirii unor superclase, împreună cu subclasele lor. Relația superclasă/subclasă se reprezintă grafic printr-un arc de la superclasă la un cerc, care la rândul lui este legată cu un arc de subclasă. Pe arcele de la subclase se desenează un semn de incluziune de la subclasă la superclasă. Aceste semn de incluziune indică direcția relației superclasă/subclasă.

Atributele specifice doar subclasei sunt atașate direct la dreptunghiul care reprezintă subclasa.

Toate aceste notații se exemplifică în figura de mai jos:



În exemplul de mai sus tipul superclasa *Locatari* se compune din mai multe subclase: *Şef de scară* şi *Familii*. Superclasa *Locatari* va avea ca membrii pe toţi locatarii unei scări. Între aceşti locatari, unii sunt capi de familie iar alţii şefi de scară. Aceşti membrii ai tipului de entitate *Locatari* vor apărea ca membrii şi în subclasele *Şef de scară*, respectiv *Familii*.

**Definiţie:** **Subclasă partajată** se numeşte subclasa care are mai multe superclase.

Membrii din subclasa partajată sunt membrii în fiecare dintre superclase. De aceea ei moştenesc attributele fiecărei superclase o astfel de moştenire se numeşte **moştenire multiplă**.

În cazul relaţiilor, dacă o subclasă este în relaţie cu un alt tip de entitate, această relaţie se referă doar la subclasa respectivă, nu şi la superclasa lui.

#### 1.4.4. Generalizarea

**Definiţie:** **Generalizare** se numeşte procesul de minimizare a diferenţelor dintre entităţi, pentru identificarea caracteristicilor comune.

Procesul de generalizare este o aproximare bottom-up a superclaselor, din subclasele originale. Deci generalizarea este inversa specializării. De exemplu dacă privim tipurile de entităţi *Familii* şi *Şef de scară*, vom observa că unele attribute ale lor caracterizează ambele tipuri. De aici rezultă necesitatea creării unei superclase care să conţină toate attributele comune celor două tipuri.

#### 1.4.5. Regurile specializării şi generalizării

În acest capitol vom discuta despre regulile ce trebuie aplicate în cazul specializării sau al generalizării.

Prima regulă se numeşte **regula disjuncţiei**. Această regulă specifică dacă subclasele unei clase sunt disjuncte, adică un membru al superclasei aparţine cel mult uneia dintre subclase. O specializare disjunctă se reprezintă cu un “**d**” înscris în cercul care leagă subclasele de superclase.

Dacă subclasele nu sunt disjuncte, adică un membru al superclasei poate să aparţină la mai multe subclase, atunci subclasele respective se numesc **non disjuncte** şi se notează cu “**o**” în cercul care leagă subclasele de superclase. În exemplul nostru subclasele *Şef de scară* şi *Familii* - care sunt subclasele clasei *Locatari* - sunt non disjuncte.

A doua regulă a specializării este **regula participării**, care poate fi totală sau parţială. Participarea este totală dacă toţi membrii superclasei sunt şi membrii subclaselor. Pentru reprezentarea participării totale, arcul de la superclasă la cercul dintre superclasă şi subclasă se dublează.

În cazul participării parţiale nu toţi membrii superclasei iau parte într-o subclasă. Acest tip de participare se reprezintă cu linie simplă între superclasă şi cerc. În exemplul nostru avem o participare parţială în cazul superclasei *Locatari*, cu subclasele *Şef de scară* şi *Familii*.

Cele două reguli se aplică distinct la procesele de specializare, sau generalizare. De aceea putem avea patru tipuri de specializări: disjunctă totală, disjunctă parțială, non disjunctă totală, sau non disjunctă parțială.

### **1.5. Descrierea cerințelor sistemului**

#### **“Asociație de locatari” - Crearea unui model EER**

În acest capitol vom demonstra crearea unui model EER. Vom descrie cerințele sistemului informatic pentru asociația de locatari, după care vom identifica entitățile, relațiile, cardinalitatea și participarea relațiilor și atributele asociate entităților. După toate aceste identificări vom crea diagrama EER asociată aplicației.

##### **1.5.1. Descrierea sistemului informatic “Asociația de locatari”**

- (1) O asociație de locatari se compune din una sau mai multe scări, aflate în una sau mai multe blocuri indentificate prin număr bloc și număr scară. Pe fiecare scară locuiesc locatari dintre care se alege un șef de scară. Fiecare dintre scări se compune din apartamente, având informații prin care se poate face relația la scări, precum și alte informații constând din: număr apartament, suprafața, numărul cutiilor poștale, a prezelor tv., a legăturilor la coșul de fum și altele.
- (2) Locatarii sunt identificați printr-un număr matricol unic în toată asociația de locatari. Informațiile despre ei sunt memorate în număr bloc, scara, etaj, apartament - reprezentând adresa - și nume. Unii dintre locatari pot fii șefi de scară si/sau capi de familii, ceea ce implică alte informații în plus pentru aceste persoane. Aceste informații sunt: pentru șef de scară, data intrării în funcție iar pentru capul de familie, numărul de persoane, numărul de persoane prezente, numărul de chei, valoarea fondului de rulment, a fondului de reparații, a altor fonduri, precum și plata curentă și restanța.
- (3) În fiecare lună se primesc facturi de la diferiți furnizori, reprezentând cheltuielile asociației de locatari. Aceste cheltuieli sunt identificați unic de un număr de cheltuială și au în componența informațiilor o informație prin care se face legătura cu un furnizor, tipul de cheltuială, numărul, data și valoarea facturii și scadența. Aceste scheltuieli pot fii pe una sau mai multe scări sau pe una sau mai multe familii.
- (4) Cheltuielile se vor plăti de asociația de locatari ori prin virament, ori prin casă.
- (5) Locatarii își plătesc datoriile la asociația de locatari cu bani gheață, primind ca dovadă a plății, chitanțe. Despre plăți se memorează o legătură la capul de familie care a efectuat plata, numărul, data și valoarea chitanței.

- (6) Furnizorii de la care vin facturile la asociația de locatari, se identifică unic printr-un cod local, sau după codul fiscal. Pe lângă aceste informații mai memorăm informații despre denumire, contul, banca și adresa sediului.
- (7) Calculul plăților pentru locatari se va face la începutul fiecărei luni, pentru luna anterioară, memorând data efectuării calculului, numărul matricol al capului familiei pentru relație, valoarea și restanța pentru luna aceea. După listarea plăților nu se mai admit modificări.
- (8) Fiecare scară este întreținută de un personal identificat de un număr matricol (altă decât la locatari) și având informațiile: legătura la scara unde lucrează, numele, data nașterii și a angajării și meseria.
- (9) În asociația de locatari pot fi mijloace fixe și obiecte de inventar, identificate prin numărul de inventar și având între informații legătura la scara unde sunt amplasate. Se mai poate memora despre ei denumirea, valoarea și valoarea amortizată.

#### 1.5.2. Crearea modelului EER

În acest capitol vom demonstra crearea modelului EER. Vom crea modelul EER pentru “Asociația de locatari”, folosind descrierea de mai sus.

##### *Identificarea tipurilor de entități:*

La început identificăm tipurile de entități din specificațiile de mai sus:

Scari	Cheltuieli
Apartamente	Achitări
Locatari	Patrimoniu
Șef de scară	Personal
Familii	Obiecte de inventar
Plăți	Furnizori
Chitanțe	

##### *Identificarea tipurilor de relații:*

Vom identifica tipurile principale de relații. Relațiile sunt reprezentate de verbe lor în tabela de mai jos.

Tip de entitate	Tip de relație	Tip de entitate
Scări	<i>sunt locuite de</i> <i>se compun din</i> <i>sunt întreținute de</i> <i>se folosește</i>	Locatari Apartamente Personal Patrimoniu
Familiiile	<i>trebuie să plătească</i>	Plăți
Furnizorii	<i>Primesc</i> <i>Provoacă</i>	Chitanțe Cheltuieli

Cheltuielile	<i>se calculează pentru</i> <i>se calculează pentru</i> <i>se achită prin</i>	Scări Familii Achitări
--------------	---	------------------------------

*Determinarea cardinalității și a participării în tipurile de relații:*

Considerăm relația Scări-le sunt locuite de Locatari. O singură scară este locuită de mai mulți locatari. Deci până acum cardinalitatea este de 1:M, însă trebuie să verificăm și relația inversă - Locatarii locuiesc pe Scări - unde se observă că un locatar locuiește pe o singură scară. Deci în final, cardinalitatea relației este 1:M. În continuare fie relația Cheltuielile se calculează pentru Scări. Observăm că putem avea mai multe scări la care să se refere o cheltuială, precum și mai multe cheltuieli pentru o scară. Deci cardinalul relației este de N:M.

Să analizăm acum participarea la relație a membrilor tipurilor de entități:

Fie relația Scări-le sunt locuite de Locatari. Putem avea cazul în care o scară să nu fie locuită deloc de locatari. În cazul relației inverse nu putem avea cazul în care un locatar să nu locuiască pe nici o scară. Deci relația directă este parțială, iar cea inversă este totală.

Cardinalitatea și participarea celorlalte relații este arătată în următorul tabel:

Tip de entitate	Tip de relație	Tip de entitate	Card.	Participarea	
				Directă	Inversă
Scări	<i>sunt locuite de</i>	Locatari	1:M	parțială	totală
	<i>se compun din</i>	Apartamente	1:M	totală	totală
	<i>sunt întreținute de</i>	Personal	1:M	parțială	totală
	<i>se folosesc</i>	Patrimoniu	1:M	parțială	totală
Familiiile	<i>trebuie să plătească</i>	Plăți	1:M	parțială	parțială
	<i>primesc</i>	Chitanțe	1:M	parțială	parțială
Furnizorii	<i>provoacă</i>	Cheltuieli	1:M	parțială	totală
Cheltuielile	<i>se calculează pentru</i>	Scări	M:N	totală	parțială
	<i>se calculează pentru</i>	Familii	M:N	totală	parțială
Cheltuielile	<i>se achită prin</i>	Achitări	1:M	totală	totală

Am observat că relația Cheltuielile sunt de tipul Tip cheltuială are cardinalul M:1. De aceea având în vedere convenția că relațiile se denumesc în direcția 1:M vom schimba numele relației în Tip cheltuială este tipul unei Cheltuieli.

*Identificarea atributelor asociate entităților:*

Vom identifica attributele entităților, attribute ce descriu caracteristicile entităților. Aceste attribute sunt trecute în următorul tabel:

Tip entitate	Atribute	Tip entitate	Atribute
Scări	Nr_bloc	Șef de scară	Nr_mat

Apartamente	Scara	Familii	Nr_Bloc
	Lift		Scara
	Apartament		Etaj
	Nr_bloc		Apartament
Locatari	Scara	Familii	Nume
	Suprafata		Data_intrare_func
	Cutii_poștale		Nr_Mat
	Nr_prize_tv		Nr_Bloc
Plăți	Nr_mat	Familii	Scara
	Nr_Bloc		Etaj
	Scara		Apartament
	Etaj		Nume
Furnizori	Apartament	Chitanțe	Nr_pers
	Nume		Nr_pers_prezente
	Data		Nr_chei
	Nr_mat		Fond_rulment
Furnizori	Valoare	Chitanțe	Fond_reparații
	Restanță		Alte_fonduri
	Cod_Furnizor		Nr_Chit
	Denumire		Nr_Mat
Furnizori	Cod_fiscal	Cheltuieli	Valoare
	Cont		Data
	Banca		Nr_Crt
	Strada		Cod_Cheltuială
Personal	Nr	Cheltuieli	Cod_Furnizor
	Bl		Nr_factură
	Sc		Data_factură
	Ap		Valoare_factură
Personal	Localitate	Achitări	Nr_crt
	Județ		Nr_Doc
	Nr_matricol		Tip_Op
	Nr_bloc		Valoare_Achit
Personal	Scara	Patrimoniul	Data
	Nume		Nr_inventar
	Data_nașterii		Nr_bloc
	Meseria		Scara
Personal	Data_angajării	Patrimoniul	Denumire
			Inv_Fix
			Valoare

*Determinarea cheilor candidat și cheilor primare:*

Vom analiza cheile candidat al fiecărei tip de entitate și vom alege una din ele să fie cheie primară. De exemplu în tipul de entitate Locatari avem o cheie candidat, numit Nr\_mat. Deci această cheie va fi cheia primară. În general cheia cea mai simplă este aleasă în funcția de cheie primară.

În tabela următoare specificăm cheile primare și alternante ale fiecărei tip de entitate:

Tip entitate	Cheie primare	Cheie alternante
Scări	Nr_bloc, Scara	
Apartamente	Nr_bloc, Scara	
Locatari	Nr_Mat	
Şef de scară	Nr_Mat	
Familii	Nr_Mat	
Plăți	Data, Nr_Mat	
Chitanțe	Nr_Chit	
Cheltuieli	Nr_Crt	
Furnizori	Cod_Furnizor	Cod_fiscal
Achitări	Nr_Doc, Tip_Op	
Personal	Nr_matricol	
Patrimoniu	Nr_inventar	

*Specializarea sau generalizarea tipurilor de entități:*

În final putem lua decizia de a specializa o clasă. Dacă avem tipuri de entități care au aceleași atribute, atunci putem generaliza aceste tipuri de entități. De exemplu din tabelul în care apar atributele tipurilor de entități, putem observa că tipurile de entități *Locatari*, *Şef de scară* și *Familii* au multe atribute comune. Deci putem generaliza tipurile de entități *Şef de scară* și *Familii* la tipul de entitate *Locatari*. După generalizare vom avea o relație de specializare/generalizare, care nu este disjunctă și este parțială.

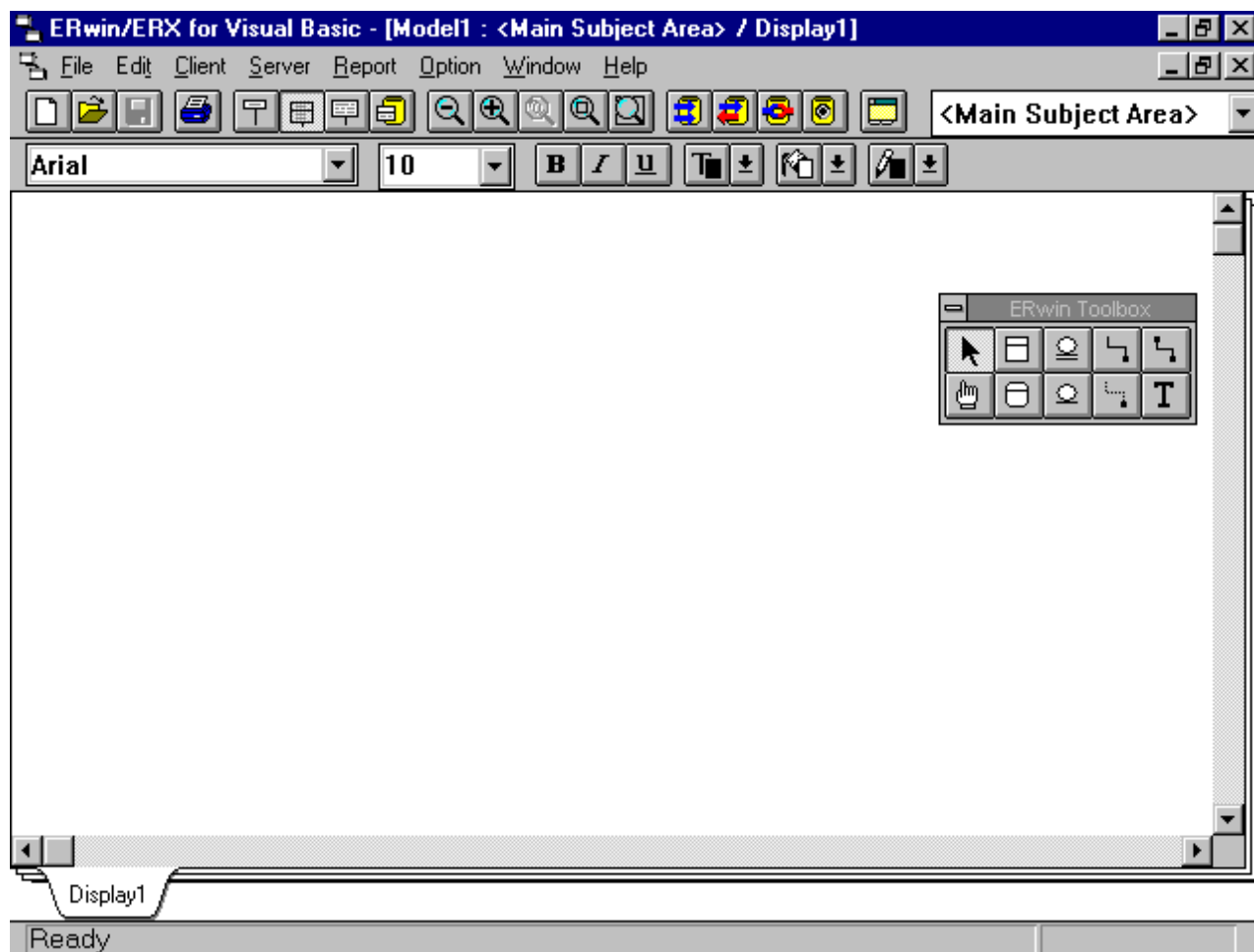
*Desenăm diagrama EER:*

Folosind toate informațiile descrise mai sus, desenăm diagrama EER a sistemului informatic al “Asociației de locatari”.



**Scule CASE pentru crearea modelului ER.**

În acest capitol vom descrie produse CASE (Computer Aided Software Engineering) care ne ajută la proiectarea bazelor de date relaționale. La elaborarea prezentei lucrări am folosit produsul ERwin produs de Logic Works.

La intrare în program fereastra va arăta în modul următor:



Pentru crearea unui model ER se folosesc butoanele din fereastra ERwin Toolbox. Cu aceste butoane se poate proiecta baza de date, desenând tabelele și relațiile dintre ele.

Cum creăm o entitate? Pentru a crea o entitate, selectăm butonul  și îl poziționăm undeva în pagină. Va apărea un dreptunghi, care are delimitat două secțiuni: cheia principală (deasupra liniei orizontale) și atributele asociate entității (sub linia orizontală). Deasupra dreptunghiului apare numele tipului de entitate. Dacă entitatea este dependentă, atunci se selectează butonul .

Exemplu de entități:

Locatari

Nr_Mat
Etaj
Nr_Bloc (FK)
Apartament
Scara (FK)
Nume

Plati

Data
Nr_Mat (FK)
Valoare
Restanta




Numele acestei entităților sunt Locatari - entitate independentă - și Plăți - entitate dependentă. În cazul entității Locatari, secțiunea chei principale se compune din câmpul Nr\_Mat iar câmpurile de sub linie sunt atributele asociate entității.

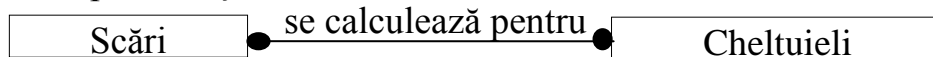
În teoria de mai sus aceste informații sunt reprezentate grafic în următorul mod:


- tip de entitate se reprezintă cu un dreptunghi etichetat cu numele entității
- atributele se reprezintă cu o elipsă etichetată cu numele atributului
- atributele din componența chei principale se subliniază

Cum notăm cheile candidate? La editarea numelui atributului care este conținut într-o cheie candidat, se trece în continuarea numelui textul: (AKn), unde n reprezintă numărul chei candidat de care aparține atributul. În exemplul nostru *Nume* este cheie candidat.

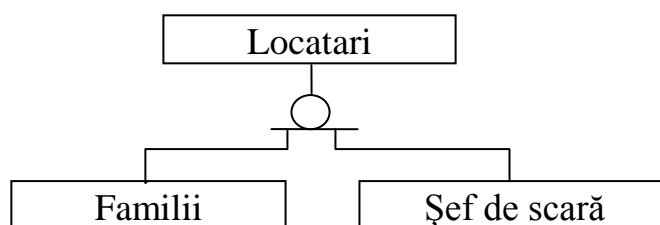
Cum evidențiem un tip de relație? Selectăm butonul corespunzător tipului de relație dorit dintre tipurile , reprezentând relație “identificare” de cardinal 1:M, N:M respectiv relație “neidentificare”. Diferența între relațiile “identificare” și “neidentificare” este că în cazul primului tip cheia entității părinte migrează în secțiunea de cheie principală a entității fiu iar în cazul al doilea migrează în secțiunea de atribute.

Exemplu de tip de relație:



Pentru a specializa/generaliza tipurile de entități, folosim butoanele , care reprezintă specializare cu participare totală respectiv parțială. În cazul specializării, cheia principală de la tipul de entitate părinte, migrează în secțiunea cheii principale a tipului de entitate fiu.

O astfel de relație este între tipurile de entitate *Locatari* și *Familii*:



Simbolul de specializare de mai sus reprezintă, că nu toate entitățile din tipul de entitate *Locatari* iau parte în una din tipurile de entități *Familii* sau *Șef de scară*. Deci participarea este parțială.

Suportul produsului ERwin pentru normalizarea bazei de date:

ERwin are suport pentru normalizarea bazei de date, dar nu are implementat un algoritm de normalizare. Ajutorul acordat proiectantului bazei de date constă în avertizarea acestuia în cazul unor erori la normalizare.

#### Suportul Formei Normale Unu:

În modelul ER attributele sunt identificate prin nume. ERwin acceptă orice nume pentru attribute, cu următoarele excepții:

- Avertizează în cazul utilizării unei denumiri de tip de entitate de două ori (depinde de setarea opțiune “Nume unică”).
- Avertizează în cazul folosirii de două ori a aceluiași nume de atribut, cu excepția numelui de rol.

ERwin nu distinge dacă un atribut conține sau nu mai multe informații. De exemplu se acceptă și denumirea “Numele copiilor” ceea ce poate reprezenta un tablou în acel atribut.

#### Suportul pentru Forma Normală Doi și Trei:

ERwin nu cunoaște dependențele funcționale din baza de date, dar poate ajuta la proiectarea bazei de date în forma normală doi sau trei. De exemplu avertizează în cazul în care o cheie care migrează (în cazul unui relații), apare de două ori în tipul de entitate. Acesta poate să apară în cazul în care sunt mai multe relații între două tipuri de entități. Potem opta pentru păstrarea cheii de două ori, dând nume diferite câmpurilor, sau pentru unificarea atributelor ce compun cheia.

Baza de date se poate proiecta pe view-uri, ERwin generând automat imaginea întregii baze de date. Se poate selecta modul de reprezentare a bazei de date, opțiunile fiind: nivel de entitate, la nivel de chei primare, sau la nivel de attribute. Toate aceste opțiuni fiind disponibile pentru nume logice și fizice.

După proiectarea bazei de date, aceasta se poate exporta în mai multe formate disponibile, dintre care amintim: FoxPro, Acces, Oracle, Ingres, AS/400, Informix, Progress și altele.

Baza de date a sistemului de gestiune a Asociației de locatari proiectat cu ERwin este următorul:



## 2. Normalizarea bazei de date

În acest capitol vom discuta despre:

- Necesitatea normalizării
- Problemele asociate cu informațiile redundante în rânduri.
- Identificarea tipurilor variate de anomalii ce pot apărea la actualizare: inserare, ștergere și modificare.
- Procesul de normalizare.
- Conceptul de dependență funcțională, modalitatea de a grupa attributele în relații
- Cum utilizăm dependențele funcționale, pentru a grupa attributele în relații care sunt în forme normale cunoscute?
- Cum definesc relațiile, formele normale?
- Cum derulăm procesul de normalizare?
- Cum identificăm prima (1NF), a doua (2NF) și a treia (3NF) formă normală, respectiv forma normală Boyce-Codd (BCNF)?

### 2.1. Necesitatea normalizării

Când proiectăm o bază de date, dorința noastră este de a reprezenta cât mai corecte informațiile și să diminuăm cât mai mult posibilitatea de a ajunge la informații eronate (dacă nu putem elimina de tot acest neajuns). Pentru a ajunge la această performanță, trebuie să folosim **normalizarea**.

Definiție: Normalizare: O tehnică de generare a unor relații cu proprietățile dorite, în scopul memorării corecte a datelor unei întreprinderi.

Procesul de normalizare prima dată a fost introdus de E. F. Codd (1972). Inițial s-au propus trei forme normale, numerotate de la unu la trei. Mai târziu s-a inclus încă o formă normală, numită Boyce-Codd, după numele celor care l-au introdus: R. Boyce și E. F. Codd.

Formele normale cele mai folosite sunt: forma normală 3 și forma normală Boyce-Codd. Există și forme normale mai tari - forma normală 4 (4NF) și forma normală 5 (5NF) - dar acestea se folosesc foarte rar.

Procesul de normalizare este o metodă formală de identificare a relațiilor bazate pe chei primare (sau chei candidat în cazul BCNF) și dependențele funcționale cu attributele asociate. Normalizarea dă posibilitatea proiectantului bazei de date, pentru a efectua o serie de teste pe baza de date, toate aceste teste ducând la prevenirea posibilității de a apărea anomalii la actualizarea bazei de date.

Pentru a ilustra procesul de normalizare, vom folosi exemple din sistemul informatic *Asociacie de locatari*.

## 2.2. Redundanța în informații și anomalii la actualizare

Partea cea mai importantă a proiectării bazei de date este de a grupa atributele în relații cu scopul de a minimiza redundanța în informații și spațiul ocupat de fișiere pe suportul magnetic.

Fie relația *Furnizori\_Cheltuieli* exemplificată mai jos. La exemple vom simplifica atributele asociate entităților.

Cod Furn.	Denumire	Cod fiscal	Nr. Crt.	Cod Chelt.	Denumire Cheltuială	Valoare
F100	Romgaz	R1234567	1	C15	Chelt pt. încălzire	1500000
F100	Romgaz	R1234567	2	C16	Chelt pt. bucătărie	500000
F110	Renel	R7654321	3	C10	Chelt cu iluminatul	3000000
F110	Renel	R7654321	4	C11	Chelt pt. func. liftului	200000

**Figura 2.1** Relația *Furnizori\_Cheltuieli*

Dependențele funcționale pentru relația *Furnizori\_Cheltuieli* de mai sus sunt următoarele:

Furnizori                    (Cod.Furn., Denumire, Cod fiscal)

Cheltuieli                (Nr.Crt., Cod chelt., Cod.furn., Valoare)

Tip\_cheltuiala        (Cod.Chelt., Denumire chelt.)

Furnizori\_Cheltuieli   (Nr.Crt., Cod chelt., Cod furn., Denumire chelt., Denumire, Cod fiscal, Valoare)

În acest exemplu, informația despre furnizor devine redundantă. Detaliile despre furnizor se repetă la fiecare introducere a unei cheltuieli noi. În dependențele funcționale specificate pentru entitatea *Cheltuieli*, apare doar codul furnizorului. Analog, denumirea cheltuielii, apare și ea în plus față de entitatea *Cheltuieli*.

O altă problemă serioasă datorată redundanței bazei de date, sunt problemele de actualizare a informației stocate. Aceste probleme se pot clasifica în anomalii de inserare, modificare, sau ștergere.

### 2.2.1. Anomalii de inserare

Anomaliile de inserare se pot clasifica în două tipuri mari:

- Pentru a adauga detaliile despre o cheltuială către un furnizor, în relația *Furnizori\_Cheltuieli* trebuie obligatoriu adăugate și detaliile despre furnizorul în cauză, chiar dacă el există deja în baza de date. Această anomalie poate duce la apariția aceluiași furnizor, având introduse detalii diferite în înregistrări diferite.
- Pentru a adăuga detaliile unui furnizor nou în relația *Furnizori\_Cheltuieli*, trebuie neapărat adăugată și o cheltuială pentru asociația de locatari către acel furnizor. În cazul în care încă nu a sosit factura de la furnizor, nu pot înregistra nici o cheltuială

și deci trebuie introduse date vide în locul cheltuieli. Cum *Nr.Crt.* este cheia primară în relația *Furnizori\_Cheltuieli*, introducerea de date vide în acest câmp va strica integritatea entității.

### 2.2.2. Anomalii de ștergere

În cazul ștergerii ultimei cheltuieli a asociației de locatari către un furnizor, se va șterge și furnizorul. Deci toate detaliile introduse despre acel furnizor vor fi pierdute, ceea ce duce la obligativitatea reintroducerii datelor la o nouă folosire al acelui furnizor. În cazul entității separate pentru cheltuieli și furnizori, dacă se șterge și ultima cheltuială către un furnizor, datele despre furnizor rămân totuși în entitatea *Furnizori*.

### 2.2.3. Anomalii de modificare

Dacă în relația *Furnizori\_Cheltuieli* dorim să schimbăm valoarea unui atribut al unui furnizor, va trebui să schimbăm datele la fiecare apariție a acelui furnizor. De exemplu dacă dorim să schimbăm codul fiscal al furnizorului cu codul F100, va trebui să schimbăm acest atribut în două locuri. Dacă în timpul modificărilor, care poate dura destul de mult, intervine și un incident în uema căruia vor rămâne înregistrări nemodificate, atunci baza de date devine inconsistentă. Această anomalie se poate evita prin folosirea a două entități distincte precum *Cheltuieli* și *Furnizori*. În acest caz dacă trebuie să modific un atribut al unui furnizor, va trebui s-o modific doar într-un singur loc: în entitatea *Furnizori*.

## 2.3. **Dependențe funcționale**

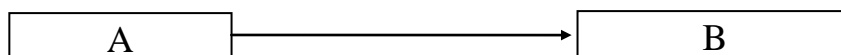
Unul din cele mai importante concepte asociate normalizării este dependența funcțională. Dependența funcțională descrie relația dintre atribute. În acest capitol vom descrie conceptul de dependență funcțională, urmând ca în capitolele următoare să descriem procesul de normalizare a bazei de date.

### 2.3.1. Definiția dependenței funcționale

**Definiție: Dependență funcțională:** Descrie relația dintre atribute. De exemplu dacă atributul A este în relația R cu atributul B, atunci atributul B este dependent funcțional de atributul A (notat:  $A \rightarrow B$ ), dacă orice valoarea al lui A este asociat prin relația R cu exact o valoare a atributului B.

Dependența funcțională este o proprietate a semanticii atributelor în relații. Semantica indică atributele care sunt în relație și specifică dependențele funcționale dintre atribute.

Considerăm o relație între două atribute A și B, unde atributul B este dependent funcțional de atributul A (atributele A și B pot fi compuse din mai multe atribute). Cu alte cuvinte, dacă știm valorile câmpului A, atunci analizând valorile lui B în relația cu A, vom considera că pentru valori diferite ale lui A - care fiind cheie, trebuie să aibă valori diferite - avem valori diferite și pentru câmpul B. Dependența funcțională dintre două câmpuri se reprezintă grafic în următorul mod:



**Figura 2.2** Diagrama dependenței funcționale

**Definiție: Determinant:** Numim determinantul unei relații funcționale, atributul, sau mulțimea atributelor din partea stângă a săgeții.

În acest capitol vom ignora dependențele funcționale triviale, adică acele relații de tipul  $A \rightarrow B$ , în care B este dependent de un subset de atribute al lui A. Pentru a exemplifica dependențele funcționale, considerăm următorul exemplu:

**Exemplu:** Considerăm atributele *Cod furnizor* și *Denumire*. Pentru un anumit cod de furnizor, putem determina denumirea acelui furnizor. Adică denumirea este dependent funcțional de cod furnizor. Dependența inversă nu este adevărată pentru că pot exista aceleași denumiri cu coduri diferite.

Relația dintre cod furnizor și denumire este de 1:1, iar relația inversă este de 1:M.

Să identificăm dependențele funcționale din relația *Furnizori\_Cheltuieli* descrisă mai sus:

Nr. Crt., Cod Furn., Cod Chelt.	$\rightarrow$ Denumire
Nr. Crt., Cod Furn., Cod Chelt.	$\rightarrow$ Cod fiscal
Nr. Crt., Cod Furn., Cod Chelt.	$\rightarrow$ Denumire cheltuială
Nr. Crt., Cod Furn., Cod Chelt.	$\rightarrow$ Valoare
Cod Furnizor	$\rightarrow$ Denumire
Cod Furnizor	$\rightarrow$ Cod fiscal
Cod fiscal	$\rightarrow$ Denumire
Cod fiscal	$\rightarrow$ Cod Furnizor
Cod Chelt.	$\rightarrow$ Denumire cheltuială

În această relație avem 9 dependențe funcționale, care au determinantele (Nr. Crt., Cod Furn., Cod Chelt.), Cod fiscal, Cod Furn. și Cod Chelt. Putem evidenția aceste dependențe și într-un alt format:

Nr.Crt., Cod Furn., Cod Chelt. → Denumire, Cod fiscal, Denumire cheltuială, Valoare

Cod Furn. → Denumire, Cod fiscal

Cod fiscal → Denumire, Cod Furn.

Cod Chelt. → Denumire cheltuială

Pentru a identifica cheile candidate, trebuie să identificăm atributul, sau grupul de attribute care determină unic fiecare înregistrare al acestei relații. Dacă avem mai mult de o cheie candidat, atunci va trebui să identificăm și o cheie primară. Toate attributele care nu aparțin cheii primare, depind funcțional de cheia primară. În această ipostază este grupul de attribute (Nr. Crt., Cod Furn., Cod Chelt.). Attributele Cod Furn., Cod fiscal și Cod Chelt. Sunt determinanți, dar nu sunt chei candidat.

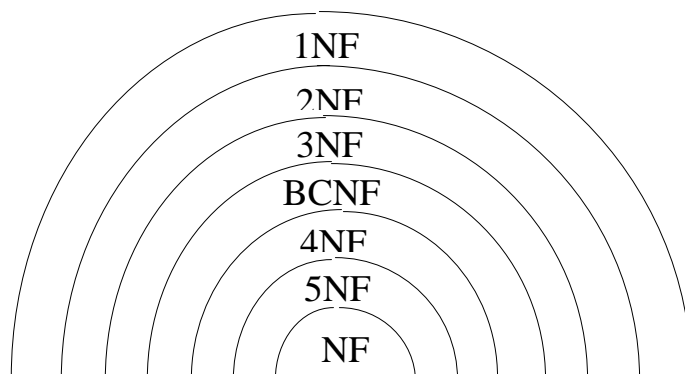
Conceptul de dependență funcțională este conceptul central al normalizării, ceea ce vom descrie în capitolele următoare.

## **2.4. Procesul de normalizare**

Normalizarea este un proces formal de analiză a relațiilor bazate pe chei primare (sau pe baza cheilor candidat în cazul BCNF). Normalizarea presupune urmarea unor reguli prin care baza de date se poate normaliza până la un anumit grad. Dacă o cerință nu este satisfăcută, relația trebuie dezcompusă în mai multe relații, care individual satisfac cerințele formei normale.

Normalizarea se execută trecând prin toate formele normale, până la forma normală cerută. La proiectarea unei baze de date trebuie să ajungem cel puțin la forma normală unu, dar ca să evităm toate anomaliile descrise mai înainte, este recomandat aducerea bazei de date la BCNF.

În figura următoare evidențiem relația dintre diversele forme normale. Observăm că unele din relațiile în 1NF este și în 2NF, dintre care unele în 3NF și așa mai departe.





**Figura 2.3.** Ilustrarea grafică a relației dintre formele normale.

## **2.5. Forma Normală Unu (1NF)**

Înainte să discutăm despre forma normală unu, vom da definiția stadiului dinainte de această formă normală.

**Definiție: Formă Nenormalizată (UNF):** O tabelă care conține una sau mai multe grupuri repetitive.

**Definiție: Forma Normală Unu (1NF):** Relație în care la intersecția oricărei linii cu oricare coloană găsim un câmp care conține exact o valoare.

În acest capitol începem procesul de normalizare. Pentru început transferăm datele de la sursă într-o tabelă, care va fi o tabelă nenormalizată. Pentru a transforma această tabelă în forma normală unu, identificăm și ștergem grupurile repetitive din tabelă. Grupurile repetitive sunt attribute, sau grup de attribute, pentru care avem diferite valori pentru aceeași valoare a cheii, ceea ce este în contradicție cu definiția cheii. Eliminarea acestor grupuri repetitive se poate realiza în două moduri:

Conform primei modalități, eliminăm grupurile repetitive, prin crearea altor înregistrări, în care să fie introduse valorile din aceste grupuri, împreună cu celelalte valori ai atributelor din înregistrarea la care se lucrează. Tabele astfel rezultată va fi în formă normală unu.

În a doua modalitate, fiecare valoare a grupurilor repetitive le copiem într-o nouă relație împreună cu cheia primară din tabela inițială. Putem avea mai multe grupuri repetitive. În acest caz creăm mai multe relații noi. Aceste relații noi, precum și tabela normalizată vor fi în formă normală unu.

De exemplu să luăm relația Furnizori\_Cheltuieli:

<b>Cod Furn.</b>	<b>Denumire</b>	<b>Cod fiscal</b>	<b>Nr. Crt.</b>	<b>Cod Chelt.</b>	<b>Denumire Cheltuială</b>	<b>Valoare</b>
F100	Romgaz	R1234567	1	C15	Chelt pt. Încălzire	1500000
			2	C16	Chelt pt. Bucătării	500000
F110	Renel	R7654321	3	C10	Chelt cu iluminatul	3000000
			4	C11	Chelt pt. Func. liftului	200000

**Figura 2.4.** Tabela nenormalizată Furnizori\_Cheltuieli.

În această tabelă observăm că pentru furnizorul “Romgaz” avem două tipuri de cheltuieli. Pentru a transforma această tabelă în 1NF, trebuie să avem o singură valoare la fiecare intersecție linie coloană.

În cazul primei modalități, scriem repetițiile pe diferite rânduri iar coloanele care nu conțin repetiții, vor fi copiate pe fiecare rând. După despărțirea repetițiilor pe mai multe rânduri, identificăm o nouă cheie.

<b>Cod Furn.</b>	<b>Denumire</b>	<b>Cod fiscal</b>	<b>Nr. Crt.</b>	<b>Cod Chelt.</b>	<b>Denumire Cheltuială</b>	<b>Valoare</b>
F100	Romgaz	R1234567	1	C15	Chelt pt. Încălzire	1500000
F100	Romgaz	R1234567	2	C16	Chelt pt. Bucătării	500000
F110	Renel	R7654321	3	C10	Chelt cu iluminatul	3000000
F110	Renel	R7654321	4	C11	Chelt pt. Func. liftului	200000

**Figura 2.5.** Tabela Furnizori\_Cheltuieli în 1NF,  
creată prin prima modalitate de transformare.

În tabela normalizată, noua cheie va fi (Cod Furn., Nr. Crt., Cod Chelt.).

Normalizând tabela inițială după a doua modalitate, vom crea o a doua tabelă cu informațiile care nu se repetă, împreună cu cheia primară din tabela inițială. Deci cele două tabele vor fi următoarele:

Furnizori (Cod Furn., Denumire, Cod fiscal)

Cheltuieli (Cod Furn., Cod Chelt., Nr. Crt., Denumire cheltuială, Valoare)

Cele două tabele astfel create sunt în 1NF:

<b>Cod Furn.</b>	<b>Nr. Crt.</b>	<b>Cod Chelt.</b>	<b>Denumire Cheltuială</b>	<b>Valoare</b>
F100	1	C15	Chelt pt. Încălzire	1500000
F100	2	C16	Chelt pt. Bucătării	500000
F110	3	C10	Chelt cu iluminatul	3000000
F110	4	C11	Chelt pt. func. Liftului	200000

<b>Cod Furn.</b>	<b>Denumire</b>	<b>Cod fiscal</b>
F100	Romgaz	R1234567
F100	Romgaz	R1234567
F110	Renel	R7654321
F110	Renel	R7654321

**Figura 2.6.** Relația Cheltuială și Furnizor, create prin metoda a doua de normalizare.

Pentru a demonstra trecerea la forma normală doi și mai mari, vom folosi relația Furnizori\_Cheltuieli, evidențiată în figura 2.5.

## 2.6. Forma Normală Doi (2NF)

Forma normală doi se bazează pe conceptul de dependență funcțională totală, ceea ce vom descrie în continuare.

### 2.6.1. Dependența funcțională totală

**Definiție: Dependența funcțională totală:** Dacă A și B sunt attributele unei relații, atunci B este total dependent funcțional de atributul A, dacă B este dependent funcțional de A, dar nu este dependent funcțional de nici un subset al lui A.

De exemplu să luăm următoarea dependență funcțională:

Nr. Crt., Cod Chelt. → Denumire cheltuială

Dependența funcțională este corectă, pentru că fiecare valoare al lui (Nr. Crt., Cod Chelt.) determină unic denumirea cheltuielii, însă dependența nu este totală, pentru că *Denumire cheltuială* depinde funcțional și de un subset al lui (Nr. Crt., Cod Chelt.) și anume atributul *Cod Chelt.*

### 2.6.2. Definiția Formei Normale Doi

Forma normală doi trebuie verificată doar la relațiile care au cheie compusă pe poziție de cheie primară. Relațiile la care cheia primară se compune dintr-un singul atribut, este în 2NF. Relațiile care nu sunt în forma normală doi, pot suferi de anomalii prezentate în capitolul 2.2. De exemplu dacă în cazul relației Furnizori\_Cheltuieli vream să actualizăm datele despre furnizorul F100, trebuie să actualizăm două înregistrări.

**Definiție: Forma Normală Doi (2NF):** O relație este în forma normală doi, dacă este în forma normală unu și fiecare atribut care nu aparține cheii primare, este total dependent funcțional de cheia primară.

Vom demonstra aducerea la forma normală doi, folosind relația Furnizori\_Cheltuieli evidențiată în figura 2.5. Putem trece la forma normală doi prin ștergerea atributelor care nu depind total de cheia primară și trecerea lor într-o altă tabelă împreună cu determinantul lor. După efectuarea acestor transformări, vom avea următoarele relații:

#### Relatia Cheltuieli

<b>Cod Furn.</b>	<b>Nr. Crt.</b>	<b>Cod Chelt.</b>	<b>Valoare</b>
F100	1	C15	1500000
F100	2	C16	500000
F110	3	C10	3000000
F110	4	C11	200000

#### Relația Furnizori

<b>Cod Furn.</b>	<b>Denumire</b>	<b>Cod fiscal</b>
F100	Romgaz	R1234567
F100	Romgaz	R1234567
F110	Renel	R7654321
F110	Renel	R7654321

#### Relația Tip Cheltuială

<b>Cod Chelt.</b>	<b>Denumire Cheltuială</b>
C15	Chelt pt. încălzire
C16	Chelt pt. bucătării
C10	Chelt cu iluminatul
C11	Chelt pt. func. liftului

**Figura 2.7.** Relațiile rezultate după trecerea la 2NF a relației Furnizori\_Cheltuieli.

Relațiile rezultate au următoarea formă:

Furnizori (Cod Furn., Denumire, Cod fiscal)

Tip cheltuială (Cod Chelt., Denumire cheltuială)

Cheltuieli (Nr. Crt., Cod Furn., Cod Chelt., Valoare)

## 2.7. Forma Normală Trei (3NF)

Forma normală doi chiar dacă nu conține atâta redundanță ca forma normală unu, totuși există cazuri în care pot apărea anomalii la actualizare. Aceste anomalii apar din cauza redundanței generate de dependența tranzitivă.

### 2.7.1. Dependența tranzitivă

**Definiție: Dependență tranzitivă:** Dacă attributele A, B, C sunt în relațiile  $A \rightarrow B$  și  $B \rightarrow C$ , atunci spunem că atributul C este dependent tranzitiv de atributul A, via B.

### 2.7.2. Definiția Formei Normale Trei

**Definiție: Forma Normală Trei (3NF):** O relație care este în formă normală doi și nu există nici un atribut care să nu aparțină cheii principale și care să fie tranzitiv dependentă de cheia principală.

În cazul existenței dependenței tranzitive, ștergem coloanele care sunt tranzitiv dependente de cheia primară și creăm o relație nouă cu aceste coloane, împreună cu determinantul lor, adică cheia primară.

Examinând relațiile în forma normală de mai sus, observăm că nu există dependențe tranzitive. Deci relațiile sunt în formă normală trei.

## 2.8. Forma Normală Boyce-Codd (BCNF)

Baza de date trebuie proiectată astfel încât să nu conțină dependențe parțiale sau tranzitive, pentru că altfel ne putem confrunta cu anomaliile descrise în cap. 2.2.

### 2.8.1. Definiția Formei Normale Boyce-Codd

Forma normală Boyce-Codd este bazată pe cheile candidat din relație. O relație cu o singură cheie candidat în formă normală trei este și în formă normală Boyce-Codd.

**Definiție: Forma normală Boyce-Codd (BCNF):** O relație este în forma normală Boyce-Codd dacă și numai dacă orice determinant din relație este cheie candidat.

Să căutăm determinanții din exemplul de mai sus:

Cod Furn.	→ Denumire, Cod fiscal
Cod Chelt.	→ Denumire cheltuială
Nr. Crt., Cod Furn., Cod Chelt.	→ Valoare

Toți cei trei determinanți sunt și chei candidat. Deci exemplul de mai sus este în formă normală Boyce-Codd. Relațiile în formă normală trei sunt în general și în formă normală Boyce-Codd. În cazul în care relația nu este în formă normală Boyce-Codd, trecerea la BCNF se realizează prin ștergerea din relația inițială a atributelor care sunt asociate unui determinant care nu este cheie candidat și crearea unei noi relații cu aceste attribute și determinantul lor.

Există situații când este foarte greu de descompus attributele, ca să ajungem la BCNF. În aceste situații este indicat rămânerea la forma normală trei.

## **2.9. Să recapitulăm procesul de normalizare (de la 1NF la BCNF)**

### **Forma Normală Unu (1NF)**

Trebuie să căutăm toate intersecțiile de linii și coloane, unde există repetiții. Aceste repetiții se pot elimina prin două modalități:

- Crearea a noi înregistrări pentru fiecare valoare a repetiției, după care se caută o nouă cheie primară.
- Ștergerea atributelor care conțin repetiții și crearea a unor noi relații care vor conține attributele șterse, precum și cheia principală din relația inițială.

### **Forma Normală Doi (2NF)**

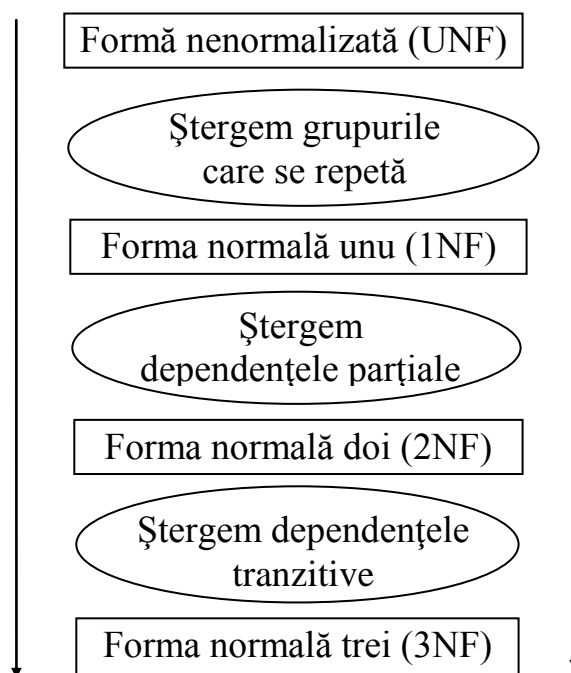
Se caută dependențele parțiale de cheia principală, adică toate attributele care depind funcțional de un subset de attribute a cheii primare. Dacă cheia primară este compusă dintr-un singur atribut, atunci relația este în forma normală doi. Dacă există dependențe parțiale, vom șterge attributele care depind parțial de cheia principală și creăm o relație nouă care să se compună din attributele șterse împreună cu determinantul lor.

### **Forma Normală Trei (3NF)**

Pentru a trece la forma normală trei, trebuie să eliminăm dependențele tranzitive. Eliminarea se realizează prin ștergerea câmpurilor dependente tranzitiv de cheia primară din relația inițială și crearea unei noi relații cu aceste attribute și determinantul lor.

### **Forma Normală Boyce-Codd (BCNF)**

Cerința la forma normală Boyce-Codd este ca fiecare determinant din relație să fie cheie candidat. În cazul în care nu este îndeplinită această cerință, vom șterge attributele dependente funcțional de determinantul care nu este cheie candidat și creăm o nouă relație în care să avem attributele șterse și determinantul lor. În unele cazuri trecerea la forma normală Boyce-Codd complică foarte mult baza de date, caz în care este de preferat rămânerea la forma normală trei.



### 3. Metodologie - Proiectarea bazei de date

În acest capitol vom învăța despre:

- Scopul metodologiei de proiectarea a bazelor de date.
- Proiectarea bazelor de date are două faze mari: proiectarea logică și proiectarea fizică.
- Utilizatorii sistemului informatic au un rol important în proiectarea bazelor de date.
- Cum documentăm procesul de proiectare a bazelor de date?
- Cum descompunem scopul proiectării în vederi (view-uri) specifice utilizatorilor întreprinderii?
- Cum utilizăm modelarea Entity-Relationship (ER), pentru a crea un model conceptual local, bazat pe informațiile adunate despre view-urile utilizatorilor?
- Cum proiectăm un model local conceptual într-un model local logic de date?
- Cum creăm relațiile pentru un model conceptual local?
- Cum validăm modelul logic de date, utilizând tehnica normalizării?
- Cum compunem modelele logice locale de date, bazate pe view-urile specifice utilizatorilor, într-un model logic global de date al întreprinderii?
- Cum ne asigurăm că modelul global este o reprezentare adevărată și totală a părții întreprinderii pe care vrem să o modelăm?

Metodologia proiectării bazei de date se compune din două etape mari: proiectarea logică, în care proiectantul decide asupra structurii logice a bazei de date, și proiectarea fizică, în care proiectantul decide cum se va implementa structura logică în sistemul de gestiune a bazelor de date (SGBD).

În acest capitol vom prezenta pas cu pas, crearea unei baze de date. Vom utiliza modelarea ER descrisă în capitolul 1, pentru proiectarea bazei de date; vom valida acest model prin normalizare, prezentat în capitolul 2; iar în final vom compune diferitele view-uri într-un model global al întreprinderii.

În acest capitol vom utiliza cuvintele “entitate” și “relație” în locul expresiilor “tip de entitate” și “tip de relație”. Unde această utilizare ar putea crea confuzii, vom specifica “tip de entitate”, respectiv “tip de relație”.

#### 3.1. Metodologia proiectării bazelor de date

##### 3.1.1. Ce este o metodologie de proiectare?

**Definiție: Metodologie de proiectare:** O aproximare structurată, care utilizează proceduri, tehnici, instrumente și documentații pentru a facilita procesul de proiectare.

Metodologia de proiectare se compune din etape, care la rândul lor se compun din pași, care orientează proiectantul la fiecare nivel al creării bazei de date.



### 3.1.2. Proiectarea logică și fizică a bazei de date

**Definiție: Proiectare logică:** Procesul de construcție a unui model de informații folosite într-o întreprindere, bazată pe modelul de date, dar independent de particularizările sistemului de gestiune a bazei de date și a altor considerente fizice.

Proiectarea logică începe cu crearea modelului conceptual al bazei de date, care este independent de implementarea într-un SGBD. Modelul conceptual este apoi proiectat pe un model logic, care va influența mai târziu modelul de date în care se va implementa.

**Definiție: Proiectare fizică:** Este procesul de descriere a implementării bazei de date într-un SGBD.

În această etapă a proiectării este creată baza de date într-un SGBD, împreună cu procedurile de actualizare. În această etapă există un feedback între proiectarea fizică și cea logică, pentru că deciziile luate la implementarea fizică pot afecta baza de date logice.

### 3.2. Prezentarea metodologiei

În acest capitol vom prezenta o descriere a metodologiei de proiectare a bazelor de date. Prima dată să vedem care sunt pașii de urmat în proiectare:

#### Pasul 1. Proiectarea logică a bazei de date relaționale:

Crearea unui model conceptual local, pentru vederile utilizatorilor.

Pasul 1.1. Identificarea tipurilor de entități.

Pasul 1.2. Identificarea tipurilor de relații.

Pasul 1.3. Identificarea și atribuirea de attribute la tipurile de entități și tipurile de relații.

Pasul 1.4. Determinarea domeniilor de definiție a atributelor.

Pasul 1.5. Determinarea atributelor care compun cheile candidate și primare.

Pasul 1.6. Specializare/generalizare (pas opțional).

Pasul 1.7. Desenarea diagramei entity-relationship.

Pasul 1.8. Verificarea modelului conceptual local cu ajutorul utilizatorului.

#### Pasul 2. Crearea și validarea modelului logic local.

Pasul 2.1. Proiectarea modelului conceptual local pe un model logic local.

Pasul 2.2. Crearea relațiilor pentru modelul logic local.

Pasul 2.3. Validarea modelului, utilizând normalizarea.

Pasul 2.4. Validarea modelului din nou, utilizând tranzacțiile.

Pasul 2.5. Desenarea diagramei ER.

Pasul 2.6. Definirea regulilor de integritate a bazei de date.

Pasul 2.7. Verificarea modelului logic local cu ajutorul utilizatorului.

#### Pasul 3. Crearea și validarea modelului logic global de date.

- Pasul 3.1. Compunerea medelelor logice locale într-un model logic global.
- Pasul 3.2. Validarea modelului logic global.
- Pasul 3.3. Verificarea posibilității de a completa baza de date în viitor.
- Pasul 3.4. Desenarea diagramei ER finale.
- Pasul 3.5. Verificarea modelului logic global cu ajutorul utilizatorului.
- Pasul 4. **Proiectarea fizică și implementarea bazei de date relaționale:**
  - Translatarea modelului logic global în SGBD.
  - Pasul 4.1. Proiectarea relațiilor de bază în SGBD.
  - Pasul 4.2. Crearea regulilor de integritate în SGBD.
- Pasul 5. Proiectarea și implementarea reprezentării fizice.
  - Pasul 5.1. Analizarea tranzacțiilor.
  - Pasul 5.2. Alegerea organozării fișierelor.
  - Pasul 5.3. Alegerea indexșilor secundari.
  - Pasul 5.4. Introducerea unei redundanțe comntrolate.
  - Pasul 5.5. Estimarea spațiului pe disc.
- Pasul 6. Proiectarea și implementarea unui mecanism de securitate.
  - Pasul 6.1. Crearea view-urilor pentru utilizatori.
  - Pasul 6.2. Proiectarea regulilor de acces la baza de date.
- Pasul 7. Verificarea sistemului operațional.

Proiectarea logică a bazei de date se divide în trei pași mari. Primul pas are ca obiectiv, descompunerea proiectării sistemului informatic în vederi, care se pot discuta cu utilizatorii sistemului. Modelul de date astfel creat, se validează prin normalizare și prin tranzacții în pasul doi. În final, se generează modelul global al întreprinderii, cars este la rândul lui validat și verificat cu ajutorul utilizatorului sistemului.

*Factori critici pentru succesul proiectării logice:*

- Lucrul interactiv cu utilizatorul sistemului.
- Folosirea unei metodologii structurate pentru procesul de proiectare a bazei de date.
- Încorporarea regulilor de integritate în modelul logic de date.
- Combinarea validării conceptuale, prin normalizare și prin tranzactii, la proiectarea modelului logic de baze de date.
- Utilizarea diagramelor pentru a reprezenta cât mai multe modele logice posibile.
- Crearea dicționarului de date, ca supliment al modelului de date.

### **3.3. Metodologia de proiectare a bazei de date logice**

În acest capitol vom prezenta pas cu pas, proiectarea unei baze de date.

## **Pasul 1. Crearea modelului conceptual local, pentru utilizatori.**

Obiectivul: Crearea unui model conceptual local, pentru view-urile utilizatorilor.

Primul pas în proiectarea bazei de date este de a colecta datele necesare pentru realizarea sistemului, ceea ce putem culege, discutând cu viitorii tilizatori ai bazei de date. Acrastă discuție presupune o despărțire în vederi, a bazei de date, vederi care pot luca separat.

Despărțirea în vederi se poate realiza în mai multe moduri. O modalitate ar fi analiza datelor globale și găsirea de părți relativ independente. O altă modalitate ar fi analiza rapoartelor, procedurilor cerute și/sau observarea sistemului existent în lucru.

Modelele conceptuale locale trebuie să conțină:

- tipuri de entități,
- tipuri de relații,
- attribute,
- domeniile atributelor,
- cheile candidat,
- chei primare

Pașii din prima etapă a proiectării logice sunt:

- Pasul 1.1. Identificarea tipurilor de entități.
- Pasul 1.2. Identificarea tipurilor de relații.
- Pasul 1.3. Identificarea și atribuirea de attribute la tipurile de entități și tipurile de relații.
- Pasul 1.4. Determinarea domeniilor de definiție a atributelor.
- Pasul 1.5. Determinarea atributelor care compun cheile candidate și primare.
- Pasul 1.6. Specializare/generalizare (pas opțional).
- Pasul 1.7. Desenarea diagramei entity-relationship.
- Pasul 1.8. Verificarea modelului conceptual local cu ajutorul utilizatorului.

### *Pasul 1.1. Identificarea tipurilor de entități*

Obiectivul: Identificarea tipurilor de entități principale în vederile utilizatorilor.

Primul pas în proiectarea bazei de date este identificarea entităților din datele furnizate de utilizatori. De exemplu, dacă avem informațiile Nr\_Mat, Nr\_Bloc, Scara, Etaj, Apartament și Nume, putem identifica entitatea Locatari. În general putem identifica entitățile în mai multe moduri. De exemplu în locul entității

Locatari, am putea crea o entitate Locatari cu attributele Nr\_Mat și Nume, iar celelalte informații în entitatea ProprietateLocatari.

Există cazuri când entitățile sunt greu de identificat, pentru că modul de prezentare a viitorilor utilizatori necesită explicații. Utilizatorii descriu aceste entități, folosind sinonime și omonime, ceea ce îngreunează identificarea entităților. Sinonimele prin care se descrie aceeași entitate, se pot considera sinonime și la crearea modelului logic, evidențiind aceste sinonime ca diverse aliasuri ai entităților.

#### *Documentarea tipurilor de entități*

După identificarea entităților, le dăm câte un nume, iar aceste nume le vom evidenția în dicționarul de date, împreună cu explicațiile despre entități, precum și posibilele aliasuri.

### *Pasul 1.2. Identificarea tipurilor de relații*

Obiectivul: Identificarea relațiilor importante dintre entități.

După identificarea entităților, va trebui să identificăm și relațiile importante dintre aceste entități. Relațiile se descriu printr-un verb al relației. De exemplu:

- Scările *sunt Locuite de* Locatari
- Furnizorii *Provoacă* Cheltuieli

La identificarea relațiilor vom lua în considerare doar relațiile care ne interesează. Degeaba există și alte relații care să se poată identifica, dacă nu prezintă importanță pentru problema noastră, atunci nu le luăm în considerație.

În cele mai multe din cazuri, relațiile sunt binare, adică se realizează între exact două entități. Există și relații mai complexe, care se realizează între mai multe entități, sau relații recursive, care pune în relație o singură entitate cu ea însăși.

#### *Determinarea cardinalității și a participării la tipurile de relații*

După identificarea tipurilor de relații, trebuie să determinăm cardinalitatea lor, alegând dintre posibilitățile: unu-la-unu (1:1), unu-la-multe (1:M), sau multe-la-multe (M:N). Dacă se cunosc valori specifice ale cardinalităților, aceste valori se scriu la documentarea relațiilor. În continuare determinăm și participarea la relație, care poate fi total, sau parțial.

#### *Documentarea tipurilor de relații*

După identificarea tipurilor de relații, le denumim și le introducem în dicționarul de date, împreună cu cardinalitatea și participarea lor.

#### *Utilizarea modelării ER*

Pentru vizualizarea sistemelor complicate, utilizăm diagrama ER, pentru că este mult mai ușor de a cuprinde toate informațiile. Vă propunem ca să utilizați întotdeauna diagrama ER, pentru o mai bună vizualizare a datelor.

### *Pasul 1.3. Identificarea și asocierea de attribute la tipurile de entități și tipurile de relații*

Obiectivul: Asocierea de attribute la tipurile de entități și la tipurile de relații.

Următorul pas în metodologie este identificarea atributelor. Aceste attribute se identifică în aceeași mod ca și entitățile. Pentru o mai ușoară identificare, trebuie să luăm entitățile și relațiile ra rând și să ne punem următoarea întrebare: *Ce informații deținem despre această ... ?* Răspunsul la această întrebare ne va da attributele căutate.

#### *Attribute simple sau compuse*

Este important să notăm dacă un atribut este simplu sau compus. Conform acestei informații va trebui să luăm decizii referitoare la acel atribut. Dacă un atribut este compus, atunci putem opta pentru descompunerea sa, dacă este necesară prelucrarea separată a detelor compuse, sau putem să-l lăsăm compus în caz contrar.

De exemplu, atributul Adresă conține informațiile (Nr\_Bloc, Scara, Etaj, Apartament). Noi va trebui să prelucrăm aceste informații separat, deci vom descompune acest atribut în cele patru attribute simple.

Putem avea cazuri în care attributele simple să le compunem. De exemplu în cazul atributelor Nume\_Familie și Prenume, neavând nevoie de aceste informații separat, le vom compune în atributul Nume.

#### *Attribute derivate (calculate)*

Sunt acele attribute, care se pot calcula din alte attribute existente în baza de date. De exemplu numărul locatarilor de pe o scară se poate număra în tipul de entitate Locatari. Deci acest atribut este atribut derivat.

În general aceste attribute nu trebuie incluse în modelul de date, pentru că în cazul în care se modifică atributul din care se calculează atributul derivat, trebuie să se modifice și acesta. În cazul în care nu se modifică, baza de date devine inconsistentă. De aceea este important de a menționa dacă un atribut este sau nu derivat.

Dacă identificăm un atribut care să nu-l putem asocia nici unei entități sau relații, ne întoarcem la pașii anteriori, identificând noua relație sau entitate la care să asociem atributul respectiv.

În cazul în care putem asocia același atribut la mai multe entități, atunci va trebui să decidem dacă generalizăm sau nu aceste entități, proces care este descris la pasul 1.6.

#### *Documentarea atributelor*

După identificarea atributelor, le asociem un nume, și le înregistrăm în dicționarul de date, împreună cu următoarele informații:

- numele și descrierea atributului,
- toate aliasurile și sinonimele prin care este cunoscut atributul,
- tipul de date și lungimea,
- valorile inițiale ale atributelor (dacă există),
- dacă atributul acceptă sau nu valoarea nulă,

- dacă atributul este sau nu compus, și dacă este atunci atributele simple care le compun,
- dacă atributul este sau nu derivat și atributul din care se derivă,
- dacă atributul acceptă sau nu mai multe valori.

#### *Pasul 1.4. Determinarea domeniului atributelor*

Obiectivul: Determinarea domeniului atributelor în modelul conceptual local.

Domeniul atributului este o mulțime de valori pe care o poate lua. Pentru a controla în totalitate domeniul atributelor, se poate evidenția următoarele:

- setul de valori admisibile pentru un atribut,
- operațiile admisibile asupra unui atribut,
- ce atribute se pot compara cu atributul respectiv, în combinațiile cu alte atribute,
- mărimea și formatul câmpului atributului.

#### *Documentarea domeniilor atributelor*

Actualizăm dicționarul de date cu domeniul de definiție a fiecărui atribut.

#### *Pasul 1.5. Determinarea atributelor care compun cheile candidat și primare*

Obiectivul: Identificarea cheilor candidat pentru fiecare entitate și alegerea cheilor primare în cazul în care sunt mai multe chei candidat.

#### *Identificarea cheilor și selectarea cheilor primare*

O cheie candidat este un atribut, sau un grup de atribute care identifică unic fiecare înregistrare din tipul de entitate. Putem identifica una, sau mai multe chei candidat. În acest caz trebuie să alegem dintre ele o cheie primară. Cheile candidat care nu sunt primare, se vor numii chei alternante. Pentru alegerea unei chei ca fiind cheie primară, vom ține cont de următoarele:

- cheia candidat, care are un număr minimal de atribute,
- cheia candidat, care își va schimba cel mai rar valoarea,
- cheia candidat, care este cel mai puțin probabil să sufere modificări în viitor,
- cheia candidat, care este compusă din cele mai puține caractere (în cazul atributelor de tip caracter),
- cheia candidat, care este cel mai ușor de folosit din punctul de vedere al utilizatorului.

Prin procesul de identificare a cheilor primare, deducem și dacă o entitate este entitate “tare”, sau entitate “slabă”. Dacă reușim să identificăm o cheie primară, atunci entitatea este tare, altfel este slabă. O entitate slabă nu poate exista fără o entitate tare, care să-i fie “părinte”. Cheia primară a entității slabe este derivată parțial sau total din cheia primară a entității tari.

#### *Documentarea cheilor primare și alternante*

Înscriem cheile primare și pe cele alternante în dicționarul de date.

### *Pasul 1.6. Specializarea/generalizarea tipurilor de entități (pas opțional)*

Obiectivul: Identificarea entităților subclasă respectiv superclasă, între entitățile apropiate.

În acest pas putem opta pentru a continua modelarea ER, folosind procesul de generalizare sau specializare. Dacă alegem procesul de specializare, vom încerca să definim unul, sau mai multe subclase ai entității respective. Dacă însă alegem procesul de generalizare, vom căuta superclase pentru acea entitate.

Un exemplu pentru procesul de generalizare ar fi entitățile Șef\_de\_scară și Familii. Ambele entități au atribuite următoarele atribute: Nr\_mat, Nr\_bloc, Scara, Etaj, Apartament și Nume. Pe lângă aceste atribute, entitatea Șef\_de\_scară mai are asociate atributul Data\_intrare\_func; iar entitatea Familii, attributele Nr\_pers, Nr\_pers\_prezente și Nr\_chei. Deci, cele două entități având atribute în comun, le putem generaliza în entitatea Locatari, care va conține atributele comune, și entitățile Șef\_de\_scară și Familii, conținând doar atributele diferite - particularizările față de superclasă.

### *Pasul 1.7. Desenarea diagramei ER.*

Obiectivul: Desenarea unei diagrame ER. care va fi reprezentarea conceptuală a vederilor utilizatorilor despre întreprindere.

În momentul acesta suntem în măsură să prezentăm o diagramă completă a modelului bazat pe vederile utilizatorilor despre întreprindere.

### *Pasul 1.8. Verificarea modelului conceptual local cu ajutorul utilizatorului*

Obiectivul: Verificarea modelului conceptual local cu ajutorul utilizatorului, pentru a vedea dacă modelul este o reprezentare adevărată a vederii utilizatorului despre întreprindere.

Înainte de a termina pasul 1, trebuie verificat modelul conceptual elaborat. Acest model include diagrama ER și documentația anexată. În cazul în care apare orice fel de anomalie, repetăm procesul de mai înainte și remediem problema.

## **Pasul 2. Crearea și validarea modelului logic local**

Obiectivul: Crearea unui model logic local bazată pe modelul conceptual al utilizatorilor asupra întreprinderii și validarea ei prin procesul de normalizare și prin tehnica tranzațiilor cerute.

În acest pas verificăm modelul conceptual creat în pasul anterior și eliminăm din el structurile care sunt dificil de realizat într-un SGBD. Dacă la sfârșitul acestui proces modelul alterat, vom corecta aceste probleme și ne vom referii în continuare

la modelul rezultat, ca fiind modelul logic local. Vom valida modelul logic prin procesul de normalizare și a tranzacțiilor.

Activitățile din acest pas sunt:

- Pasul 2.1. Proiectarea modelului conceptual local pe un model logic local.
- Pasul 2.2. Crearea relațiilor pentru modelul logic local.
- Pasul 2.3. Validarea modelului, utilizând normalizarea.
- Pasul 2.4. Validarea modelului din nou, utilizând tranzacțiile.
- Pasul 2.5. Desenarea diagramei ER.
- Pasul 2.6. Definirea regulilor de integritate a bazei de date.
- Pasul 2.7. Verificarea modelului logic local cu ajutorul utilizatorului.

### *Pasul 2.1. Proiectarea modelului conceptual local pe modelul logic local*

Obiectivul: Verificarea modelului conceptual local pentru eliminarea structurilor care se pot implementa greu într-un SGBD și proiectarea modelului rezultat la modelul logic local.

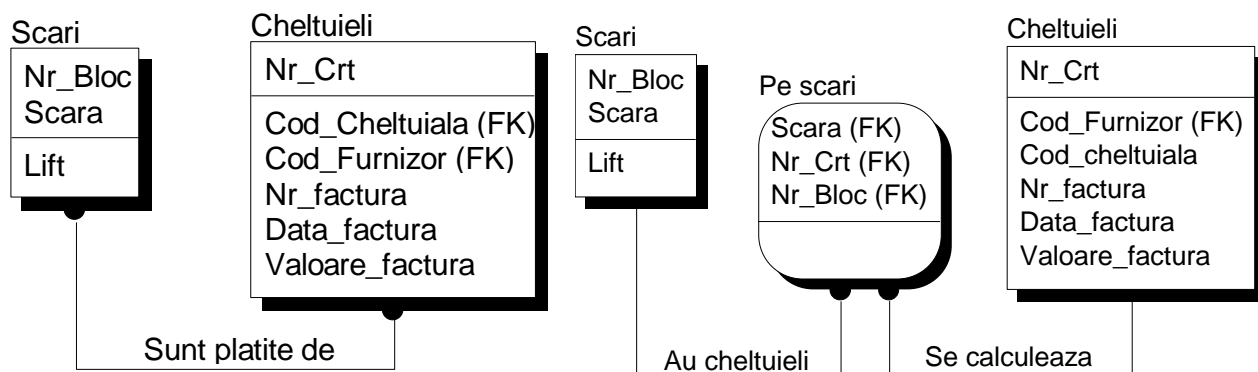
În pasul întâi am pregătit un model conceptual bazat pe informațiile date de utilizator. Acest model trebuie prelucrat, pentru a putea fi ușor de prelucrat de sistemul de gestiune a bazelor de date. Obiectivele acestui pas sunt:

- (1) Eliminarea relațiilor M:N.
- (2) Eliminarea relațiilor complexe.
- (3) Eliminarea relațiilor recursive.
- (4) Eliminarea relațiilor cu attribute.
- (5) Reexaminarea relațiilor 1:1.
- (6) Eliminarea relațiilor redundante.

#### *(1) Eliminarea relațiilor multe-la-multe*

Dacă în modelul de date apar relații de tipul multe-la-multe (M:N), trebuie descompuse în două relații unu-la-multe (1:M) prin adăugarea unei noi entități suplimentare.

De exemplu, pot exista mai multe cheltuieli pentru o scară, dar și o cheltuială (factură) poate să se refere la mai multe scări. Deci relația dintre entitatea Scări și entitatea Cheltuieli este de M:N, ceea ce este evidențiat în figura 3.1.(a).





**Figura 3.1.** (a). Relație de tipul N:M. (b). Relație transformată în două relații 1:M.

Această relație se poate elimina, prin crearea unui tip de entitate suplimentar, care să facă legătura dintre scări și cheltuieli. Diagrama acestor relații se vede în figura 3.1.(b).

Notăm, că tipul de entitate nou creat - Pe\_scări - este tip de entitate slabă, pentru că depinde atât de tipul de entitate Scări, cât și de tipul de entitate Cheltuieli.

*(2) Eliminarea relațiilor complexe*

O relație complexă este o relație între mai mult de două tipuri de entități. Dacă în modelul conceptual apar relații complexe, acestea trebuie eliminate. Se pot elimina prin crearea unui nou tip de entitate, care să fie în relație de tipul 1:M cu fiecare tip de entitate din relația inițială, partea cu M a relației fiind spre tipul de entitate nou creat.

*(3) Eliminarea relațiilor recursive*

Relațiile recursive sunt niște relații particulare, prin care un tip de entitate este în relație cu el însuși. Dacă apare o astfel de relație în modelul conceptual, ea trebuie eliminată. Eliminarea se poate rezolva prin crearea unei noi entități unde să se evidențieze fiecare entitate care este legată de entitatea din tipul de entitate inițial. În acest caz vom avea o relație de tipul 1:M între tipul de entitate inițial și tipul de entitate nou creat și o relație de tipul 1:1 între tipul de entitate nou creat și tipul de entitate inițial.

*(4) Eliminarea relațiilor cu attribute*

Dacă în modelul conceptual avem relații cu attribute, putem descompune această relație, identificând un nou tip de entitate în care să înregistrăm attributele necesare.

*(5) Reexaminarea relațiilor de tipul 1:1*

În modelul conceptual putem avea entități între care să avem relație de tipul 1:1. Se poate întâmpla ca aceste entități să fie de fapt aceeași entitate cu nume diferite. Dacă suntem în acest caz, unim cele două entități, cheia primară devenind cheia primară al uneia dintre entități.

*(6) Eliminarea relațiilor redundante*

O relație este redundantă dacă se poate ajunge de la un tip de entitate la alt tip de entitate pe mai multe drumuri. Vă amintim că noi vrem să ajungem la un model minimal și deci relațiile redundante nu sunt necesare. Decizia ca o relație este redundantă trebuie să fie precedată de o analiză amănunțită a relațiilor care compun cele două drumuri dintre cele două entități, pentru că pot apărea situații, când o relație este aparent redundantă, dar în realitate este nevoie de ea.

În finalul acestui pas putem spune că am eliminat din modelul conceptual acele structuri care sunt dificil de implementat și deci este mai corect ca în continuare să ne referim la acest model ca fiind un **model logic local de date**.

## *Pasul 2.2. Crearea de relații peste modelul logic local*

**Obiectivul:** Crearea de relații peste modelul logic.

Relația pe care un tip de entitate o are cu alt tip de entitate este reprezentată prin mecanismul cheie primară/cheie străină. Cheia străină pentru o entitate este reproducerea cheii primare altei entități. Pentru a decide entitățile unde vom include copia cheii primare a altei entități, trebuie înainte să identificăm entitățile “părinte” și “fiu”. Entitățile “părinte” se referă la acele entități ale căror chei primare se vor copia în entitățile “fiu”.

Pentru descrierea relațiilor vom folosi un limbaj de definire a bazei de date (Database Definition Language - DDL). În acest limbaj, specificăm prima dată numele entității, urmat de atributele asociate între paranteze. După aceea identificăm cheia primară și toate cheile alternante, precum și/sau cheile străine.

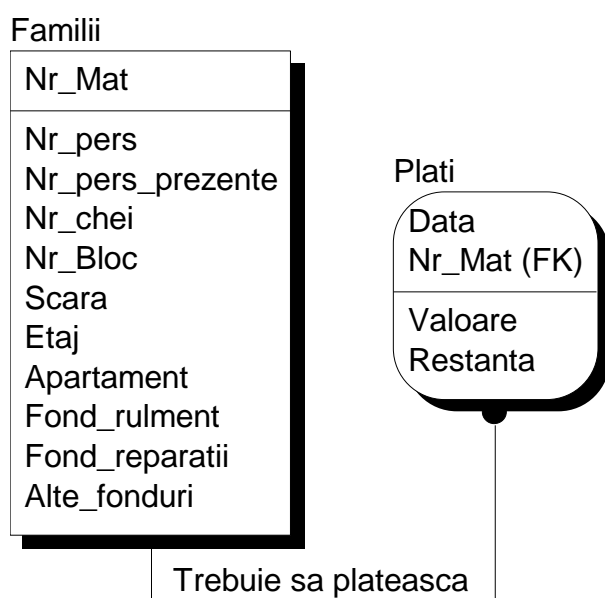
### *Tipuri de entitate tari*

Pentru tipurile de entități tari în modelul logic crearea unei relații include toate atributele entității. Pentru atributele compuse al unei entități, vom include numai atributele simple din compunerea atributului compus în descrierea entității.

De exemplu tipul de entitate Familii, prezentată în figura 3.2. se descrie în următorul mod.

**Familii** (Nr\_mat, Nr\_bloc, Scara, Etaj, Apartament, Nume, Nr\_pers,  
Nr\_pers\_prezente, Nr\_chei)

**Cheie primară:** Nr\_mat



**Figura 3.2.** Exemplu de model logic.

### *Tipurile de entități slabe*

Descrierea tipurilor de entități slabe se face la fel ca și tipurile de entități tari, cu o completare și anume, evidențierea cheii străine. De exemplu descrierea tipului de entitate de mai sus se descrie astfel:

**Plăți** (Data, Nr\_mat, Valoare, Restanță)

**Cheie primară:** Data, Nr\_mat

**Cheie străină:** Nr\_mat **se referă la** Familii(Nr\_mat)

Menționăm că în cazul acesta cheia străină este și în compunerea chei primare. Deci înainte de introducerea cheii străine, cheia primară nu identifica unic o entitate. La terminarea acestui pas, putem identifica cheile primare pentru toate tipurile de entități din modelul logic.

*Tipurile de relații binare de tipul unu-la-unu (1:1)*

Pentru fiecare tip de relație binară de tipul 1:1 între două tipuri de entitate E1 și E2 găsim o copie a cheii primare a tipului de entitate E1 în compunerea tipului de entitate E2, sub denumirea de cheie străină. Identificarea tipului de entitate “părinte” și a tipului de entitate “fiu” depinde de participarea entităților la relație. Tipul de entitate care participă parțial la relație este desemnat ca fiind “părinte” iar cel cu participare totală “fiu”. Dacă ambele tipuri de entitate participă parțial sau total la relație, atunci tipurile de entități se pot evidenția ca fiind “părinte” sau “fiu” arbitrar. În cazul în care participarea este totală, putem încerca să combinăm cele două tipuri de entități într-una singură. Această compunere poate să fie posibilă în cazul în care nici unul dintre cele două tipuri de entități nu mai ia parte la altă relație.

*Tipurile de relații binare de tipul unu-la-multe 1:M*

Pentru toate relațiile 1:M între două entități E1 și E2 în modelul logic de date, vom avea copia cheii primare a entității E1 în compunerea entității E2. Totdeauna entitatea de partea unu a relației este considerată entitate “părinte”, iar cealaltă entitate “fiu”.

*Atributele cu mai multe valori*

Pentru fiecare atribut A care permite mai multe valori din entitatea E1 în modelul logic de date, creăm o nouă relație care va conține atributul A împreună cu cheia primară a entității E1 pe post de cheie străină. Cheia primară a noii relații va fi atributul A, sau dacă este necesar, compunerea ei cu cheia primară al lui E1.

*Relațiile superclasă/subclasă*

Pentru fiecare relație superclasă/subclasă vom identifica superclasa ca fiind entitatea “părinte”, iar subclasa entitatea “fiu”. Există multe moduri de a reprezenta relația aceasta. De exemplu să luăm relația prezentată în figura 3.3.

(Opțiunea 1)

**Locatari** (Nr\_mat, Nr\_bloc, Scara, Etaj, Apartament, Nume,  
Data\_intrare\_func, Nr\_pers, Nr\_pers\_prez, Nr\_chei)

**Cheie primară:** Nr\_mat

(Opțiunea 2)

**Familii** (Nr\_mat, Nr\_bloc, Scara, Etaj, Apartament, Nume,  
Nr\_pers, Nr\_pers\_prez, Nr\_chei)

**Cheie primară:** Nr\_mat

**Șef\_de\_scară** (Nr\_mat, Nr\_bloc, Scara, Etaj, Apartament, Nume,  
Data\_intrare\_func)

**Cheie primară:** Nr\_mat

(Opțiunea 3)

**Locatari** (Nr\_mat, Nr\_bloc, Scara, Apartament, Nume)

**Cheie primară:** Nr\_mat

**Șef\_de\_scară** (Nr\_mat, Data\_intrare\_func)

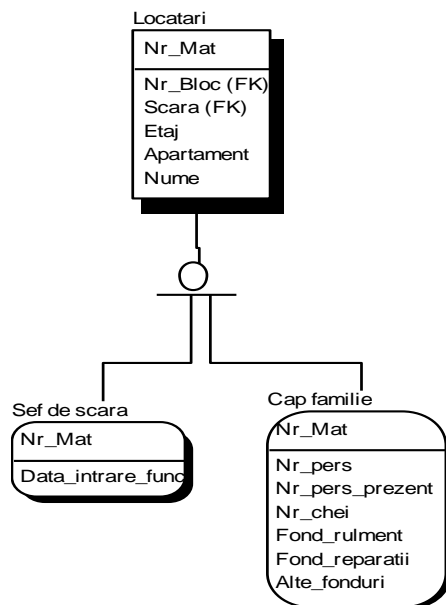
**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_mat **referire la** Locatari(Nr\_mat)

**Familii** (Nr\_mat, Nr\_pers, Nr\_pers\_prez, Nr\_chei)

**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_mat **referire la** Locatari(Nr\_mat)



**Figura 3.3.** Exemplu de relație superclasă/subclasă

*Documentarea relațiilor și a atributelor din cheile străine*

Actualizăm dicționarul de date, introducând fiecare atribut nou introdus în compunerea unei chei la acest pas.

*Pasul 2.3. Validarea, utilizând normalizarea*

Obiectivul: Validarea modelului logic, utilizând tehnica normalizării.

Examinăm procesul de normalizare după cum am descris în capitolul 2. Prin normalizare trebuie să demonstrăm că modelul creat este consistent, conține redundanță minimală și are atabilitate maximă.

Normalizarea este procesul prin care se decide dacă atributele pot sau nu să rămână împreună. Conceptul de bază a teoriei relațiilor este că atributele sunt grupate împreună pentru că există o relație logică între ele. Câteodată baza de date nu este cea mai eficientă. Acesta se argumentează prin următoarele:

- Proiectarea normalizată organizează datele în funcție de dependențele funcționale. Deci acest proces este situat undeva între proiectarea conceptuală și cea fizică.
- Proiectul logic nu este un proiect final. El ajută proiectantul să înțeleagă natura datelor în întreprindere. Proiectul fizic poate fi diferit. Există posibilitatea ca unele tipuri de entități să se denormalizeze. Totuși normalizarea nu este un timp pierdut.
- Proiectul normalizat este robust și independent de anomalii de actualizare prezentate în capitolul 2.
- Calculatoarele moderne au mult mai multă putere de calcul, ca cele de acum câțiva ani. Din această cauză, câteodată este mai convenabil implementarea unei baze de date cu puțină redundanță, decât suportarea cheltuielilor pentru procedurile adiționale.
- Normalizarea produce o bază de date care va fi ușor extensibil în viitor.

Procesul de normalizare include următoarele etape mari:

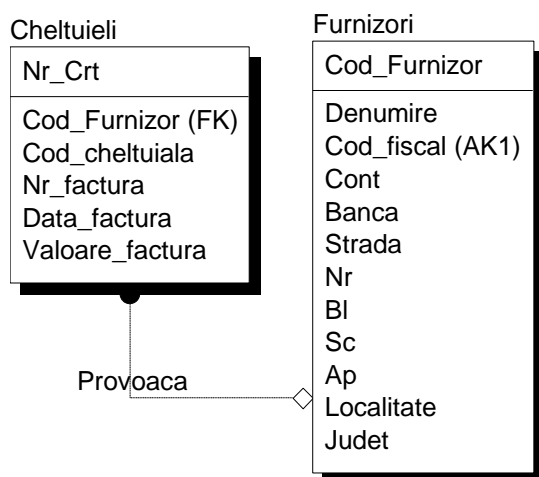
- Forma Normală Unu (1NF), eliminarea grupurilor repetitive;
- Forma Normală Doi (2NF), eliminarea dependențelor parțiale de cheia primară;
- Forma Normală Trei (3NF), eliminarea dependențelor tranzitive de cheia primară;
- Forma Normală Boyce-Codd (BCNF), eliminarea anomaliilor care au mai rămas.

#### *Pasul 2.4. Validarea modelului prin tranzacții*

Obiectivul: Verificarea ca modelul de date suportă toate tranzacțiile cerute de utilizator.

În acest pas se validează baza de date prin verificarea tranzacțiilor ce se cer de utilizator. Luând în considerare tipurile de entități, relațiile, cheile primare și străine, încercăm să rezolvăm manual cerințele utilizatorilor. Dacă reușim să rezolvăm fiecare tranzacție cerută, atunci înseamnă că modelul creat este valid. Dacă nu putem rezolva o tranzacție, atunci este foarte posibil să fi omis un atribut, o relație, sau o entitate din modelul de date.

Trebuie să examinăm în două posibilități, dacă baza de date suportă tranzacțiile cerute. Una dintre ele ar fi prin rezolvarea de tranzacții. De exemplu să luăm relația dintre Furnizori și Cheltuieli exemplificată în figura 3.4.



**Figura 3.4.** Exemplu de relație pentru validarea prin tranzacții.

#### *Inserarea de detalii despre un nou furnizor.*

Cheia primară al acestei entități este Nr\_furnizor. Deci prima dată verific dacă numărul introdus nu există deja; după care în caz că nu există acest cod, inserez detaliile despre furnizor. Dacă există deja codul, nu admit inserarea. Pot verifica și existența codului fiscal, care pentru această entitate este cheie alternantă.

#### *Ștergerea detaliilor despre un furnizor*

Se caută codul furnizorului de șters. Dacă există se șterge furnizorul, actualizându-se și cheia străină în entitatea Cheltuieli. Dacă nu există codul cerut, atunci apare un mesaj de eroare și nu este șters nici un furnizor.

A doua modalitate de verificare trebuie folosită în cazul în care avem entități care nu iau parte direct la nici o tranzacție. Pentru verificarea acestor entități, vom verifica niște interogări pergătie special.

#### *Pasul 2.5. Desenarea diagramei ER.*

**Obiectivul:** Desenarea diagramei Entity-Relationship, care este reprezentarea logică a vederilor utilizatorilor aspra întreprinderii.

#### *Pasul 2.6. Identificarea regulilor de integritate.*

**Obiectivul:** Identificarea regulilor de integritate pentru vederile utilizatorilor asupra întreprinderii.

Regulile de integritate sunt importante pentru a proteja baza de date împotriva posibilelor inconsistențe. Dacă este necesar, putem produce un proiect fizic de bază de date, pentru a putea vedea mai ușor ce reguli sunt necesare.

Vom considera cinci tipuri de reguli de integritate:

- necesitatea datelor,
- reguli asupra domeniului atributelor,

- integritatea entităților,
- integritatea referințelor,
- regulile înțepinderii.

#### *Necesitatea datelor*

Există atribute care nu pot conține valoarea nulă, ci trebuie să aibă totdeauna o valoare. De exemplu fiecare cheltuială înregistrată trebuie să aibă asociat un furnizor al serviciilor sau al obiectelor ce se plătesc prin acea cheltuială.

Aceste reguli deja le-am identificat, când am documentat atributele în pasul 1.3.

#### *Reguli asupra domeniului atributelor.*

Unele atribute au un domeniu de definiție bine definit. Aceste domenii trebuie respectate. Domeniile de definiție au fost deja identificate când am documentat domeniile atributelor în pasul 1.4.

#### *Integritatea entităților.*

Cheia primară a entităților nu poate lua valori nule. De exemplu fiecare furnizor trebuie să aibă un cod diferit de zero.

Aceste reguli au fost deja identificate, când am documentat cheile primare în pasul 1.5.

#### *Integritatea referințelor*

Cheia străină din tipul de entitate “fiu” face ligătura cu o entitate din tipul de entitate “părinte”. Deci, dacă cheia străină conține o valoare, ea trebuie neapărat să se regăsească și în tipul de entitate “părinte”. De exemplu tipul de entitate Cheltuieli conține cheia străină Cod\_furnizor, care se referă la Furnizori(Cod\_furnizor). Dacă această cheie nu este nulă, atunci trebuie să găsim un furnizor care să aibă acel cod.

Prima întrebare pe care trebuie să ne-o punem este: Poate fii cheia străină nulă? În cazul exemplului nostru asta ar însemna că există cheltuieli care nu se plătesc nimănui. Aceste cazuri nu sunt admise de lege, deci nu putem avea valoare nulă în cheia străină din tipul de entitate Cheltuieli.

În general dacă participarea tipului de entitate “fiu” este totală, atunci nu putem avea valoare nulă în cheia străină. În caz contrar putem avea valoare nulă.

Pentru a demonstra diferitele cazuri la definirea acestor reguli, folosim relația de mai sus dintre Furnizori și Cheltuieli, care este de tipul 1:M. Considerăm următoarele cazuri:

*Cazul 1: Inserarea unei entități în tipul de entitate “fiu” (Cheltuieli):* Pentru a verifica integritatea referinței, verificăm dacă atributele din componența cheii străine (Cod\_furnizor) sunt vide, sau să existe entitate în tipul de entitate “părinte” care să aibă valoare cheii primare egală cu valoare cheii străine.

*Cazul 2: Ștergerea unei entități din tipul de entitate “fiu” (Cheltuieli):*  
Ștergerea unei entități din tipul de entitate “fiu” nu cauzează probleme cu privință la integritatea referințelor.

*Cazul 3: Actualizarea cheii străine în tipul de entitate “fiu” (Cheltuieli):*  
Acest caz este similar cazului 1.

*Cazul 4: Ștergerea unei entități din tipul de entitate “părinte” (Furnizori):*  
Dacă se șterge o entitate din tipul de entitate “părinte”, integritatea referințelor se strică în cazul în care există entități în tipul de entitate “fiu”, care se referă la entitatea ștearsă. Există strategii severe de a rezolva integritatea referințelor:

- **FĂRĂ ACȚIUNE** Neacceptarea ștergerii unei entități din tipul de entitate părinte, dacă acesta este referit de o entitate din tipul de entitate fiu. În cazul nostru, nu se acceptă ștergerea furnizorului, dacă el are o factură de încasat.
- **CASCADĂ** Dacă o entitate din tipul de entitate părinte este ștearsă, se șterg automat toate entitățile din tipurile de entități fiu. Dacă tipurile de entități fiu au și ei la rândul lor alte tipuri de entități fiu, se va efectua ștergerea în cascadă în toate tipurile de entități fiu, până la ultimul nivel. Cu alte cuvinte, dacă se șterge un furnizor, atunci automat se șterge fiecare factură pe carea are de încasat acest furnizor.
- **SETARE LA NUL** Dacă o entitate din tipul de entitate părinte se șterge, atunci se vor seta la valoare nulă toate cheile străine ai tipurilor de entități fiu în cascadă până la ultimul nivel. În exemplul nostru, dacă se șterge un furnizor, atunci se vor seta la valoare nulă toate referințele la acest furnizor în tipul de entitate Cheltuieli. Acesta înseamnă că nu vom ști ca anumite cheltuieli la ce furnizor trebuie plătite.
- **SETARE IMPLICITĂ** Este analog cazului de setare la nul, cu diferența că aici se setează cheia străină la o valoare implicită în loc de valoare nulă. În exemplul nostru putem seta cheia străină din Cheltuieli la o valoare a cheii primare din Furnizori, care să conțină un furnizor predefinit - de exemplu cu numele de “Furnizor șters”.
- **FĂRĂ MODIFICARE** În cazul ștergerii unei entități din tipul de entitate părinte, nu se actualizează deloc cheile străine din tipurile de entități fiu.

*Cazul 6: Modificarea cheii primare în tipul de entitate părinte (Furnizori):*  
Dacă se modifică cheia primară din tipul de entitate părinte, integritatea referințelor se strică. Pentru menținerea integrității, se pot folosi toate cazurile descrise mai sus, dar cel mai indicat este folosirea cazului CASCADĂ.

#### *Regulile întreprinderii.*

În final evidențiem acele reguli care sunt date de realitatea ce se va modela în baza de date.

#### *Documentarea tuturor regulilor de integritate.*

Trecem toate regulile de integritate în dicționarul de date.



### *Pasul 2.7. Verificarea modelului logic local cu ajutorul utilizatorului.*

Obiectivul: Convingerea că modelul creat reprezintă în totalitate realitatea care trebuie modelată în baza de date.

La acest pas modelul local logic este clomplet și este bine documentat. Înainte de a trece la pasul 3, trebuie verificată modelul, să fie conform cu realitatea. În cazul în care se găsesc diferențe, ne vom reîntoarce la pașii anteriori și modificăm cele necesare.

### **Pasul 3. Crearea și validarea modelului global logic de date.**

Obiectivul: Combinarea modelelor locale logice într-un model logic glaobal care să reprezinte întreprinderea care este modelată.

A treia activitate majoră în proiectarea bazei de date logice este crearea modelului logic global, prin compunerea tuturor modelelor locale. După combinarea modelelor locale, trebuie validată modelul global prin tehnica de enormalizare, după care prin tehnica tranzațiilor. Acest proces utilizează aceleași tehnici pe care le-am descris la pașii 2.3. și 2.4.

Acest proces este foarte important în proiectarea bazei de date, pentru că el demonstrează că reprezentarea întreprinderii este independentă de orice utilizator, funcție sau aplicație. Activitățile din acest pas sunt:

- Pasul 3.1. Compunerea medelelor logice locale într-un model logic global.
- Pasul 3.2. Validarea modelului logic global.
- Pasul 3.3. Verificarea posibilității de a completa baza de date în viitor.
- Pasul 3.4. Desenarea diagramei ER finale.
- Pasul 3.5. Verificarea modelului logic global cu ajutorul utilizatorului.

#### *Pasul 3.1. Compunerea modelelor logice locale într-un model logic global.*

Obiectivul: Compunerea tuturor modelelor logice locale într-un model logic global al întreprinderii.

În cazul unui sistem mic, cu puține vederi ai utilizatorilor, este relativ ușor de a compune modelele logice locale. În cazul unui sistem mai mare însă, trebuie să urmăim un proces sistematic de realizare a modelului global. Pașii acestui proces sunt următoarele:

- (1) Verificarea numelor entităților și a cheilor lor primare.
- (2) Verificarea numelor relațiilor.
- (3) Compunerea entităților de pe view-urile locale.
- (4) Includerea (fără compunere) a entităților care apar pe doar una dintre view-uri.
- (5) Compunerea relațiilor de pe view-urile locale.
- (6) Includerea (fără compunere) a relațiilor care apar pe doar una dintre view-uri.

- (7) Căutarea entităților și a relațiilor care lipsesc (dacă există).
- (8) Căutarea cheilor străine.
- (9) Căutarea regulilor de integritate.
- (10) Desenarea modelului logic global.
- (11) Actualizarea documentației.

Cel mai ușor de compus mai multe modele locale este compunerea succesivă a două câte două dintre modele.

*(1) Verificarea numelor entităților și a cheilor lor primare.*

Această verificare se face folosind și dicționarul creat în decursul pașilor de dinainte. Probleme apar doar atunci când:

- Două entități au același nume, dar sunt de fapt diferiți.
- Două entități sunt aceleași, dar nu au aceleași nume.

Probabil va fi necesară analizarea atributelor entităților, printru a rezolva această problemă. În particular, putem utiliza cheia primară și numele entității, pentru a identifica entitățile echivalente.

*(2) Verificarea numelor relațiilor.*

Această activitate este asemănătoare celui descris la entități.

*(3) Compunerea entităților de pe view-urile locale.*

Examinăm numele și atributele entităților ca vor fi compuse. Activitățile care se includ în acest pas sunt:

- Compunerea entităților care au același nume și același chei primare.
- Compunerea entităților care au același nume, dar cu chei primare diferite.
- Compunerea entităților care au nume diferite, cu chei primare egale sau diferite.

*Compunerea entităților care au același nume și același chei primare.* În general entitățile cu același chei primare reprezintă “lumea reală”, și deci pot fi compuse. Atributele care apar în entitățile de pe ambele view-uri, vor fi trecute doar o singură dată. Dacă într-un view apare un atribut compus, iar într-un alt view același atribut dar descompus în atribute simple, atunci vom întreba, dacă se poate, utilizatorii pentru a decide asupra formei de utilizare a atributului.

*Compunerea entităților care au același nume, dar au chei primare diferite.* În astfel de situații, căutăm două entități care au același nume și nu au același chei primare, dar au același chei candidat. Cele două entități pot fi compuse, urmând ca după compunere să alegem o cheie primară, restul rămânând chei alternante.

*Compunerea entităților care nu au nume comune și nu au același chei primare.* Aceste entități se pot depista prin analiza atributelor celor două entități.

*(4) Includerea (fără compunere) a entităților care apar doar într-un view.*

Pasul anterior identifică toate entitățile comune. Celelalte entități, se vor include în modelul logic global exact așa cum apar în view-ul respectiv.

*(5) Compunerea relațiilor de pe modelele locale.*

În acest pas analizăm similitudinile dintre relații de pe diferite modele locale. În timpul compunerii relațiilor trebuie rezolvate și conflictele dintre relații, ca și

regulile de cardinalitate și participare. Putem compune relații care au aceleași nume și același scop, sau același scop, dar nume diferite.

*(6) Includerea (fără compunere) a relațiilor care apar doar pe un view.*

Relațiile care au rămas neincluse în modelul global după pasul anterior, se includ în modelul global fără modificare.

*(7) Căutarea entităților și relațiilor care lipsesc.*

Este unul din cei mai grei pași din crearea modelului global. Trebuie căutate acele entități și relații, care s-au omis la pașii anteriori și n-au ajuns în modelul global.

*(8) Căutarea cheilor străine.*

În decursul pașilor anterioare, s-au modificat entități, chei primare și chei străine. În acest pas verificăm dacă cheile străine sunt corecte în fiecare entitate fiu și efectuăm toate modificările necesare.

*(9) Căutarea regulilor de integritate*

Verificăm dacă regulile de integritate a modelului global nu sunt în conflict cu regulile definite la modelele locale. Fiecare problemă de acest gen se rezolvă cu ajutorul utilizatorului sistemului.

*(10) Desenarea diagramei ER.*

Acum desenăm diagrama ER pentru modelul global de date.

*(11) Actualizarea documentației.*

Actualizăm documentația, ca să reflecte toate modificările aduse în acest pas modelului.

### *Pasul 3.2. Validarea modelului logic global.*

Obiectivul: Validarea modelului global logic de date, folosind normalizarea, după care folosind tranzațiile cerute.

Acest pas este schivalent cu pașii 2.3. și 2.4., unde am validat modelul local de date.

### *Pasul 3.3. Verificarea posibilităților de extindere a bazei de date în viitor.*

Obiectivul: Determinarea ca dacă modelul se acomodează ușor la modificări oricât de mari ce pot intervenii în viitor.

Este important ca modelul creat să fie expansibil în viitor. Dacă modelul nu are această calitate, viața lui poate fi scurtă și pentru o mai mare modificare va trebui refăcută de la început.

### *Pasul 3.4. Desenarea diagramei Entitz-Relationship finale.*

Obiectivul: Desenarea unei diagrame ER, care să reprezinte modelul global de date al întreprinderii.

*Pasul 3.5. Verificarea modelului global cu ajutorul utilizatorului.*

Obiectivul: Verificarea că modelul global reprezintă în totalitate realitatea.

În acest moment modelul global este complet și documentat. Împreună cu utilizatorul sistemului se verifică acest model și se aduc eventualele corecturi prin întoarcerea la psurile în cauză.

## 4. Metodologia proiectării bazei de date logice - Exemplu

În acest capitol vom învăța despre:

- Cum să folosim metodologia de proiectare a bazelor de date relaționale, desclisă în capitolul 3.
- Cum să folosim această metodologie pentru proiectarea bazei de date al sistemului Asociație de locatari.

În acest capitol vom explica detaliat, cum putem folosii metodologia prezentată în capitolul 3. Vom exemplifica folosirea metodologiei cu proiectarea bazei de date pentru sistemul informatic al asociației de locatari.

În decursul acestui capitol, vom folosii expresiile “entitate” și “relație” în locul expresiilor “tip de entitate” respectiv “tip de relație”. În cazul în care pot apărea ambiguități, se va folosii “tip de entitate” respectiv “tip de relație”.

### 4.1. Cerințele utilizatorilor.

Analiza și colectarea datelor îl vom începe la o asociație de locatari. Aici trebuie să cerem informațiile necesare pentru realizarea sistemului. Trebuie să colectăm informațiile despre lucrul de zi cu zi a șefului asociației de locatari. Vom mai avea nevoie de toate rapoartele pe care ei le întocmesc. Toate informațiile despre sistemul de evidență a asociației de locatari se pot împărții în mai multe view-uri. Aceste view-uri sunt:

- Evidența locatarilor și a apartamentelor din asociație,
- Evidența cheltuielilor locatarilor,
- Evidența personalului asociației,
- Evidența obiectelor de inventar și a mijloacelor fixe,
- Plata datoriilor asociației către furnizori”.

#### 4.1.1. Cerințele la baza de date.

##### **Evidența locatarilor și a apartamentelor din asociație:**

- (1) În baza de date vor fi memorate toți locatarii asociației de locatari. Informațiile necesare despre locatari sunt: adresa (număr bloc, scara, etajul și apartamentul) și numele. Ei vor fi unic determinați de un număr matricol unic pe toată asociația de locatari.
- (2) Dintre locatari trebuie să evidențiem pentru fiecare familie câte un reprezentant. Pentru fiecare familie trebuie să avem la dispoziție informațiile: numărul

membriilor de familie, numărul persoanelor prezente în locuință în luna curentă și numărul de chei de la ușa principală.

- (3) Tot dintre locatari se alege câte un șef de scară pentru fiecare scară din asociație. Șef de scară trebuie să fie exact unu pe fiecare scară și trebuie să locuiască în scara pentru care s-a ales. Pentru șeful de scară mai avem în plus informația: data intrării în funcție.
- (4) La evidența clădirilor din asociația de locatari, avem de memorat scările care aparțin asociației. Scările sunt identificate unic prin numărul blocului și litera scării. Mai trebuie să știm despre fiecare scară dacă are lift.
- (5) Pe fiecare scară avem mai multe apartamente. Despre apartamente trebuie să memorăm informațiile următoare: nr. apartamentului, suprafața, numărul de cutii poștale și numărul de prize tv.

### **Evidența cheltuielilor locatarilor**

- (1) Fiecare cheltuială se înregistrează în momentul primirii facturii de la furnizor. Cheltuielile sunt identificate unic printr-un număr de ordine care este continuu pe un an. Pentru fiecare cheltuială se înregistrează informațiile: codul cheltuielii, codul furnizorului, numărul și data facturii, valoarea facturii.
- (2) Cheltuielile se pot referii la o singură scară sau la mai multe scări, sau se pot referii la o singură familie sau mai multe familii.
- (3) Pentru fiecare familie se înregistrează lunar valoarea ce trebuie plătită asociației. Dacă este cazul, se înregistrează și restanțele.
- (4) Când o familie își achită datoria la asociație, se emite o chitanță, care trebuie să conțină informații despre valoarea plătită, data plății și numele persoanei care a făcut plata. Chitanța se identifică unic prin numărul său.
- (5) Pentru a efectua mai ușor înregistrarea cheltuielilor, memorăm și datele despre furnizori. Aceste date sunt: denumirea, codul fiscal, contul bancar și banca, adresa (strada, numărul, bl., sc., ap., localitatea și județul). Furnizorii se pot identifica unic printr-un cod intern - cod furnizor.
- (6) Pe lângă cheltuielile către furnizori există și cheltuieli, care trebuiesc plătite de locatari, dar care rămân la asociație. Astfel de cheltuială este fondul de rulment, fondul de reparații, sau dacă este cazul, și alte fonduri. Aceste cheltuieli se vor înregistra în același mod ca și celelalte, cu diferența că aici se va introduce un “furnizor” special pregătit.
- (7) Fiecare familie trebuie să aibă plătită o sumă la asociația de locatari pentru fondul de rulment. În cazuri speciale se adună bani și pentru fondul de reparații, sau alte fonduri.

### **Evidența personalului asociației.**

- (1) Personalul asociației se va memora în baza de date, identificarea făcându-se după un număr matricol unic. Mai folosim informațiile: numele, data nașterii, meseria, data angajării și scările din asociație unde lucrează.

- (2) Un angajat poate să lucreze în mai multe scări și pe o scară pot să lucreze mai mulți angajați.

**Evidența mijloacelor fixe și a obiectelor de inventar.**

- (1) Mijloacele fixe și obiectele de inventar se vor identifica unic, cu ajutorul unui cod numit număr de inventar. Pentru aceste obiecte noi reținem următoarele informații: numele, valoarea și locul unde este amplasat.

**Plata datoriilor asociației către furnizori.**

- (1) O factură sosită de la un furnizor se poate achita prin casă, sau prin ordin de plată. Pentru aceste achitări reținem informațiile: valoarea achitată, data achitării și numărul urdinului de plată sau a chitanței.

**4.1.2. Tranzacțiile cerute.**

**Evidența locatarilor și a apartamentelor din asociație:**

- (1) Listă cu locatarii de pe o scară,
- (2) Listă cu familiile de pe o scară,
- (3) Listă cu șefii de scară,
- (4) Listă cu proprietățile apartamentelor ocupate pe o scară.

**Evidența cheltuielilor locatarilor**

- (1) Listă cu cheltuielile fiecărei familii pe scări,
- (2) Listă cu toate cheltuielile asociației de locatari.
- (3) Chitanța care se emite la fiecare plată.

**Evidența personalului asociației.**

- (1) Listă cu personalul asociației.

**Evidența mijloacelor fixe și a obiectelor de inventar.**

- (1) Listă cu mijloacele fixe și valoarea lor, din asociația de locatari,
- (2) Listă cu obiectele de inventar ai asociației de locatari.

**Plata datoriilor asociației către furnizori.**

- (1) Listă cu facturile scadente primite de la furnizori,
- (2) Situația plăților facturilor de la furnizori, înglobând facturile dintr-o perioadă dată, împreună cu modul lor de plată.

## **4.2. Utilizarea metodologiei de proiectare a bazelor de date relaționale.**

### **Pasul 1. Crearea modelelor conceptuale locale, bazate pe view-urile utilizatorilor:**

Înainte să ne apucăm de crearea modelului conceptual local, să ne reamintim din ce se compune acest model:

- tipuri de entități,
- tipuri de relații,
- attribute,
- domeniile atributelor,
- chei candidat,
- chei primare.

În acest capitol vom exemplifica metodologia de proiectare pe două vederi ai sistemului informatic:

- Evidența locatarilor și a apartamentelor din asociație,
- Evidența cheltuielilor locatarilor.

#### *Pasul 1.1. Identificarea tipurilor de entități.*

Începem să identificăm tipurile de entități din toate vederile utilizatorilor. Vom descrie separat identificarea acestor entități pentru cele două vederi.

În cazul evidenței locatarilor și al apartamentelor, tipurile de entități sunt:

Locatari	Scări
Șef de scară	Apartamente
Familii	

În cazul evidenței plăților tipurile de entități sunt:

Cheltuieli	Furnizori
Plăți	Chitanțe
Familii	Scări

#### *Documentarea tipurilor de entități*

În documentarea tipurilor de entități includem o descriere amănunțită a fiecărei entități, împreună cu aliasurile lor. Aceste informații sunt prezentate în anexa 4.1.

#### *Pasul 1.2. Identificarea tipurilor de relații.*

În continuare identificăm tipurile de relații. Tipurile de relații se reprezintă prin verbe ale relațiilor. Tipurile de relații din cele două view-uri sunt prezentate în tabelele de mai jos:



Tipuri de relații din view-ul “Locatari”:

Tip de entitate	Tip de relație	Tip de entitate
Scări	sunt locuite de	Locatari
	sunt locuite de	Familii
	sunt conduse de	Şef de scară
	se compun din	Apartamente

Tipuri de relații din view-ul “Cheltuieli”:

Tip de entitate	Tip de relație	Tip de entitate
Furnizorii	provoacă	Cheltuieli
Celtuieli	sunt plătite de	Familii
	sunt plătite de	Scări
Familii	trebuie să plătească	Plăți
	primesc	Chitanțe

Dacă la acest pas apar ambiguități, ele trebuie neapărat clarificate cu utilizatorii.

*Determinarea cardinalității și a participării pentru tipurile de relații.*

În continuare vom determina cardinalul și participarea relațiilor prezentate în tabelele de mai sus. Cardinalul poate să fie unul dintre următoarele: unu-la-unu (1:1), unu-la-multe (1:M), sau multe-la-multe (M:N). Participarea poate să fie parțială sau totală.

Informațiile pe baza cărora trebuie să determinăm aceste caracteristici ai tipurilor de relații, sunt prezentate în capitolul 4.1.1. În cazul în care apar ambiguități, ele trebuie clarificate cu ajutorul utilizatorului.

Să luăm de exemplu relația dintre Scări și Locatari. Pentru a fi foarte siguri de cardinalul relației, vom pune următoarea întrebare:

Întrebare: Pot locui pe o scară mai mulți locatari?

Răspuns: Da.

Deci relația Scările *sunt locuite de* Locatari are cardinalul 1:M. În continuare va trebui să aflăm și cardinalul relației inverse, adică a relației Locatarii *locuiesc în* apartamentele de pe Scări.

Întrebare: Un locatar poate locui în mai multe apartamente de pe scări diferite?

Răspuns : Nu.

La această întrebare avem nevoie de o explicație: Dacă o persoană are două locuințe în aceeași asociație de locatari, atunci el va apare în entitatea locatari de

două ori, cu număr matricol diferit. Deci un număr matricol poate să locuiască doar într-un singur apartament.

Deci cardinalul acestei relații este de 1:1, de unde rezultă că relația inițială are cardinalul 1:M.

Pentru a afla participarea entităților la relație, punem următoarele întrebări:

Întrebare: Fiecare scară este locuită de cel puțin un locatar?

Răspuns : Nu.

Întrebare: Fiecare locatar locuiește într-unul din aceste scări?

Răspuns: Da.

Deci tipul de entitate scări participă parțial, iar tipul de entitate Locatari participă total la relație.

Fie relația Cheltuieli *sunt plătite de* Scări, din view-ul “Cheltuieli”. Să determinăm cardinalul și participarea ei:

Întrebare: Există cheltuială care se referă la mai multe scări?

Răspuns : Da.

Întrebare: Scările pot avea mai multe cheltuieli?

Răspuns : Da.

Întrebare: Fiecare cheltuială trebuie să fie asociată unei scări?

Răspuns : Nu, pentru că poate fi asociată doar unor familii.

Întrebare: Fiecare scară trebuie să aibă cheltuieli?

Răspuns : Nu, pentru că pot fi scări fără locatari.

După aceste întrebări și răspunsuri, am ajuns la concluzia că relația are cardinalul N:M (ambele relații - și cea directă și cea inversă - au cardinalul 1:M), iar participarea parțială de ambele părți.

Cardinalul și participarea relațiilor de mai sus sunt prezentate în anexa 4.3.

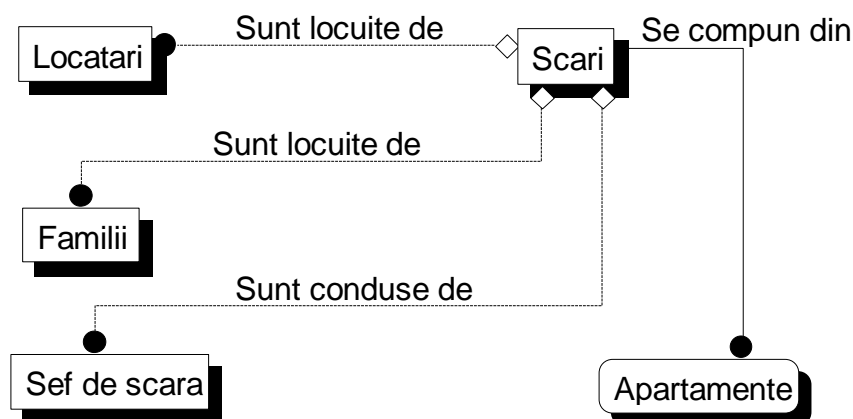
*Utilizarea modelării ER.*

Pentru o înțelegere mai ușoară a relațiilor dintre entități, se poate folosi diagrama ER. Diagramele ER ale celor două view-uri ale căror relații au fost prezentate mai sus, se pot vedea în figura 4.1. (a) și (b).

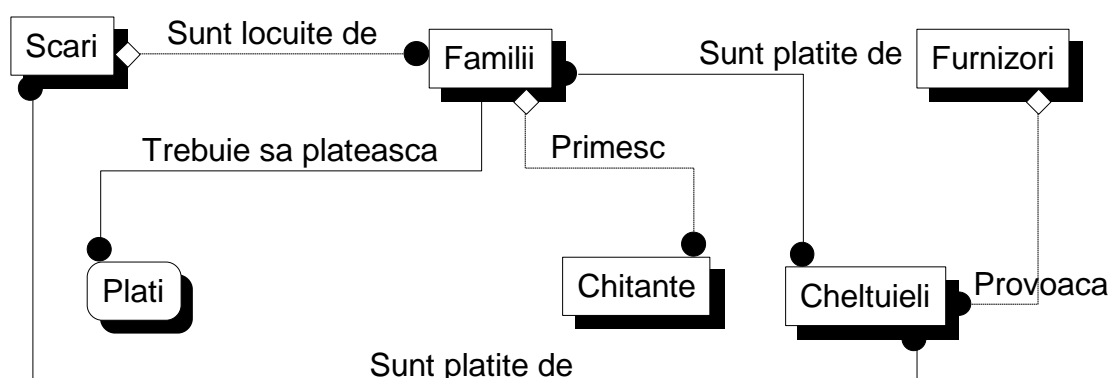
Menționăm, că verbele care reprezintă relațiile de cardinal 1:M se pun în direcția unu-la-multe. În cazul în care relația a fost altfel evidențiată, ea va fi redenumită.

*Documentarea tipurilor de relații.*

Documentarea view-urilor care apar în figura 4.1. este realizată în anexa 4.3.



**Figura 4.1.(a)**  
Diagrama ER. a view-ului “Locatari”.



**Figura 4.1.(b).** Diagrama ER a view-ului “Cheltuieli”

*Pasul 1.3. Identificarea și asocierea atributelor la tipurile de relații și tipurile de entități.*

În acest pas identificăm attributele ce se asociază tipurilor de entități sau tipurilor de relații. Attributele sunt informații care caracterizează entitățile, resoeective relațiile. Aceste attribute le putem găsi în descrierea acordată de utilizatori. În cazul nostru, attributele identificate și asociate entităților se pot regăsi în tabela 4.2.a. și 4.2.b.

**Tabela 4.2.a.** Attributele tipurilor de entități din view-ul “Locatari”:

Tip de entitate	Atribute
Locatari	Nr_Mat Adresa (Nr_bloc, Scara, Etaj, Apartament)
Familii	Nume Nr_Mat Adresa (Nr_bloc, Scara, Etaj, Apartament) Nume Nr_pers Nr_pers_prez

Sef_Scara	Nr_chei Nr_Mat Adresa (Nr_bloc, Scara, Etaj, Apartament) Nume Data_intrare_func
Scări	Adresa (Nr_bloc, Scara) Lift
Apartamente	Apartament Suprafața Cutii_poștale Nr_prize_tv

**Tabela 4.2.b.** Atributele tipurilor de entități din view-ul “Cheltuieli”:

Tip de entitate	Atribute
Familii	Nr_mat Fond_rulment Fond_reparații Alte_fonduri
Plăți	Data Valoare Restanță
Chitanțe	Nr_Chit Valoare Data
Cheltuieli	Nr_crt Cod_cheltuiala Nr_factura Data_factura Valoare
Furnizori	Cod_furnizor Denumire Cod_fiscal Cont Banca Adresa (Strada, Nr., Bl., Sc., Ap., Localitate, Jud)

*Documentarea atributelor:*

Pentru documentație se descrie fiecare atribut, se menționează tipul de date folosit precum și lungimea câmpului, reguli, valoarea implicită, toate aliasurile lui, dacă atributul este sau nu compus, dacă este derivat sau nu, dacă admite mai multe valori și dacă admite valoarea nulă.

O parte a acestei documentații este exemplificată în anexa 4.2.

#### *Pasul 1.4. Determinarea domeniilor atributelor:*

În acest pas determinăm domeniile atributelor. Domeniul unui atribut este mulțimea acelor valori, pe care le poate lua în lucrul de zi cu zi.

##### *Documentarea atributelor.*

Câteva din domeniile exemplului nostru sunt arătate în anexa 4.4.

#### *Pasul 1.5. Determinarea atributelor care aparțin cheilor candidat și primare:*

Acum vom examina tabelele 4.2.a. și 4.2.b., și vom căuta cheile candidat pentru entitățile în cauză. Dintre aceste chei candidat vom selecta cheia primară, după criterii deja descrise în capitolul 3.

Cheile primare și alternante pentru entitățile din view-ul “Locatari” și view-ul “Cheltuieli”, sunt afișate în anexa 4.2. Menționăm, că în acest pas nu asociem chei primare entităților slabe, acesastă operație fiind rezolvată în pasul 2.2.

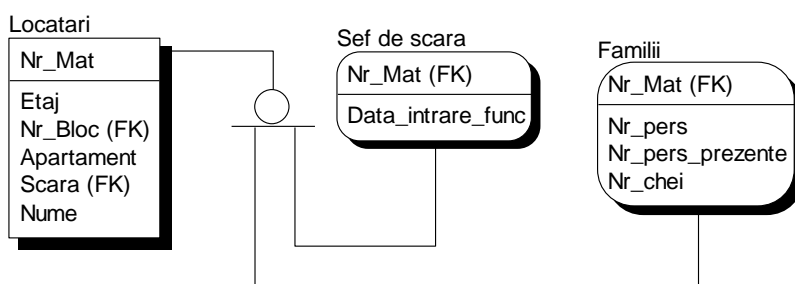
##### *Documentarea cheilor primare și alternante.*

Documentăm attributele care compun cheile primare și cheile alternante. Un exemplu de documentare găsiți în anexa 4.2.

#### *Pasul 1.6. Specializarea/generalizarea tipurilor de entități (pas opțional).*

În acest pas putem dezvolta modelul ER, utilizând procesul de specializare/generalizare asupra entităților identificate în pasul 1.1. Dacă vrem să folosim procesul de specializare, vom căuta diferențele dintre entități, iar în cazul procesului de generalizare, asemănările dintre ele.

De exemplu, în view-ul “Locatari”, entitățile Locatari, Familii și Șef\_scară au attributele Nr\_mat, Adresa și Nume în comun. Entitatea Familii mai are în plus attributele Nr\_pers, Nr\_pers\_prez și Nr\_chei iar entitatea Șef\_scară are în plus doar atributul Data\_intreare\_func. Entitatea Locatari are asociate doar attributele comune cu cele două entități. Deci putem generaliza entitățile Familii și Șef\_scară, punând pe post de superclasă entitatea Locatari, iar celelalte două fiind subclase. Relația superclasă-subclasă dintre entitatea Locatari și Familii respectiv Șef\_scară este o relație parțială și nondisjunctă, pentru că în Locatari sunt toți locatarii asociației, deci și acelea care nu sunt nici capi de familie și nici șefi de scară iar pe de altă parte, șeful de scară poate să fie și cap de familie. Figura 4.3. reprezintă diagrama celor trei entități după procesul de generalizare.



**Figura 4.3.**  
Exemplu de relație  
superclasă - subclasă

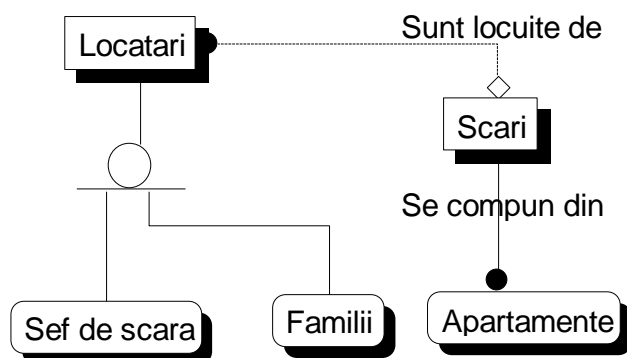
### *Pasul 1.7. Desenarea modelului ER.*

Pentru o mai bună înțelegere a modelului local creat, folosim diagrama ER. Această diagramă și documentația creată se referă la modelul local conceptual.

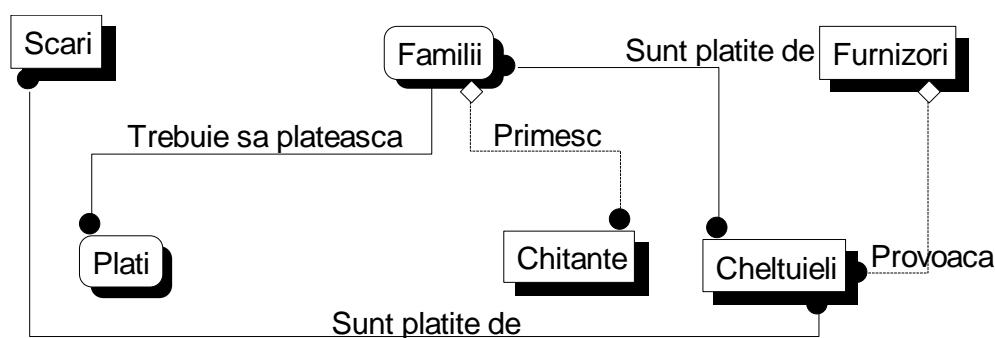
Diagramele ER ale celor două view-uri ai exemplului nostru se pot vedea în figura 4.4.a., respectiv 4.4.b.

### *Pasul 1.8. Verificarea modelului conceptual local cu ajutorul utilizatorului.*

Înainte de a termina pasul 1, trebuie verificat modelul la care s-a ajuns. Dacă se descoperă erori, atunci ne întoarcem la pasurile anterioare și le remediem. Vom repeta acești pași până când modelul creat va fi în conformitate cu realitatea.



**Figura 4.4.a.**  
Modelul conceptual local  
al view-ului “Locatari”



**Figura 4.4.b.** Modelul conceptual local al view-ului “Cheltuieli”.

### **Pasul 2. Crearea și validarea modelului local de date.**

În acest pas revizuim modelul creat și eliminăm acele structuri care se pot implementa greu în sistemul de gestiune a bazelor de date. După aceea validăm modelul, utilizând metoda normalizării și a tranzacțiilor.

### *Pasul 2.1. Proiectarea modelului local conceptual în modelul local logic.*

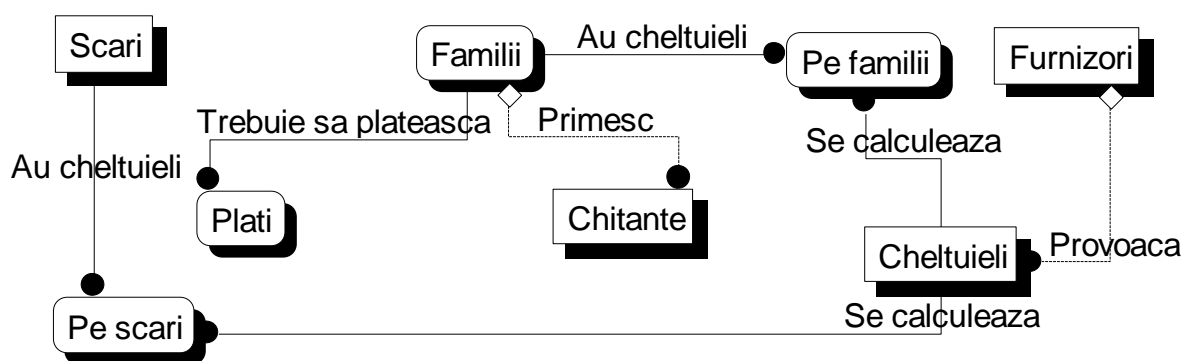
În acest pas eliminăm acele structuri din baza de date, care sunt dificil de implementat în sistemul de gestiune al bazelor de date. Pentru a rezolva această problemă, se vor urma următorii pași:

- (1) Eliminarea relațiilor de tipul N:M.
- (2) Eliminarea relațiilor complexe.
- (3) Eliminarea relațiilor recursive.
- (4) Eliminarea relațiilor cu attribute.
- (5) Reexaminarea relațiilor de tipul 1:1.
- (6) Eliminarea relațiilor redundante.

#### *(1) Eliminarea relațiilor de tipul N:M.*

Cum putem observa și în figura 4.4.b., relațiile *Cheltuieli Sunt plătite de Familii* și *Cheltuieli Sunt plătite de Scări* au cardinalul de N:M. Vom descompune aceste relații în câte două relații de tipul unu-la-multe (1:M), numite în ambele cazuri *Se calculează* și *Au cheltuieli*. Această modificare devine posibilă prin introducerea a doi entități noi, numite *Pe\_familii*, respectiv *Pe\_scări*. Aceste entități noi vor fi entități slabe, pentru că vor depinde de entitățile *Cheltuieli*, *Familii* și *Scări*. Adică cheia primară a lor va fi derivat din cheia primară a entităților tari.

Modificările se pot observa în figura 4.5.



**Figura 4.5.** View-ul “Cheltuieli”, după eliminarea relațiilor N:M.

#### *(2) Eliminarea relațiilor complexe.*

În acest pas, eliminăm toate relațiile complexe (care sunt între mai mult decât două entități) din modelul conceptual. Această eliminare se poate rezolva prin crearea a noi entități.

Exemplul nostru nu conține nici o relație complexă.

#### *(3) Eliminarea relațiilor recursive.*

Pasul acesta este pentru eliminarea tuturor relațiilor recursive, adică acele relații care pun în relație o entitate cu el însăși. În exemplul nostru nu avem astfel de cazuri.

#### *(4) Eliminarea relațiilor cu attribute.*

Se elimină toate acele relații, care au asociate atribute. Eliminarea se realizează prin adăugarea a noi entități, care vor memora aceste atribute. Nu este cazul exemplului nostru.

*(5) Reexaminarea relațiilor de tipul 1:1.*

În foarte multe din cazuri, entitățile care sunt relaționate prin relații cu cardinalul 1:1, se pot îngloba într-o singură entitate. De aceea este indicat reexaminarea relațiilor de tipul unu-la-unu.

Exemplul nostru nu conține relații de acest tip.

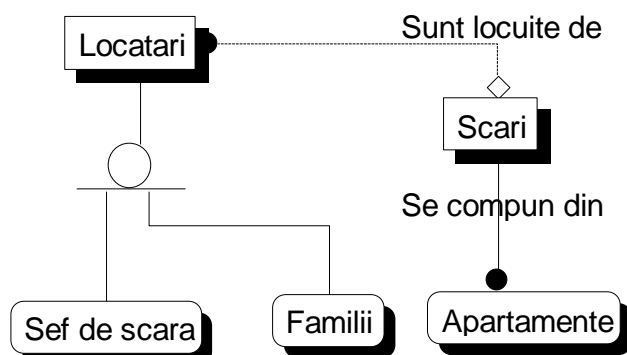
*(6) Eliminarea relațiilor redundante.*

Pot exista cazuri în care să avem relații redundante. Adică să se poată ajunge de la o entitate la alte prin mai multe drumuri diferite. În acest caz relațiile redundante trebuie eliminate, pentru că noi trebuie să ajungem la o bază de date minimală și foarte stabilă.

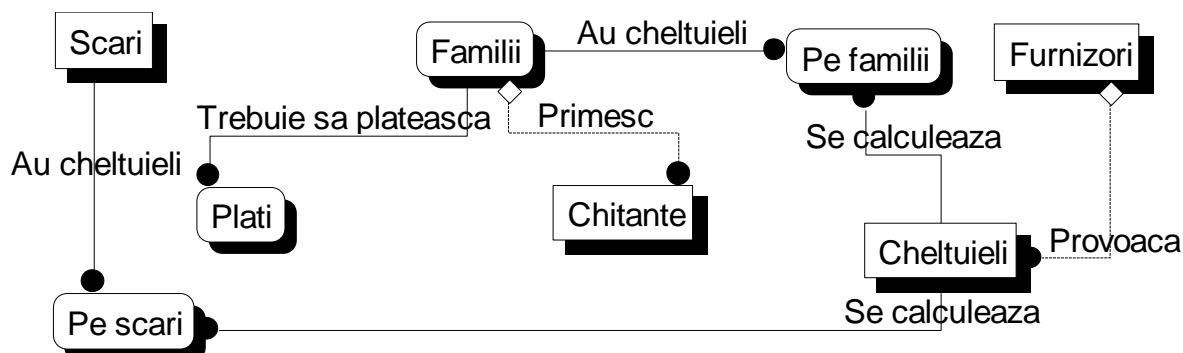
Nu este și cazul exemplului nostru.

*Desenarea diagramei ER.*

Modelul conceptual care este reprezentată în figura 4.4. a. și b., s-a modificat în acest pas. Am eliminat din acest model, toate structurile greu de implementat într-un sistem de gestiune a bazelor de date. Diagrama modificată se va numi în continuare modelul logic local al bazei de date. Acest model este prezentat în figura 4.6.a. și 4.6.b.



**Figura 4.6.a.**  
Modelul logic local al view-ului "Locatari".



**Figura 4.6.b.** Modelul logic local al view-ului "Cheltuieli".

*Pasul 2.2. Crearea de relații peste modelul logic local.*



În acest pas vom crea relațiile între entitățile din modelul logic local de date, cu ajutorul mecanismului chei primare/chei străine.

Pentru exemplificarea acestui proces, să luăm relația Furnizori *Provoacă* Cheltuieli din view-ul “Cheltuieli”. Vom folosi limbajul de definire a bazelor de date (DBDL), pentru descrierea fiecărei relații.

Pentru fiecare entitate tare a modelului, creăm o relație care include toate atributele simple ale entității. Structura entității Furnizori este:

**Furnizori** (Cod\_furnizor, Denumire, Cod\_fiscal, Cont, Banca, Str., Bl., Sc.,  
Ap., Loc., Jud.)

**Cheie primară:** Cod\_furnizor

Pentru fiecare entitate slabă, descriem relația, ănccludând atributele simple ale entității, specificând cheia străină și entitatea la care se referă această cheie străină. Structura entității Cheltuieli este:

**Cheltuieli** (Nr\_crt, Cod\_furnizor, Cod\_cheltuială, Nr\_factură, Data\_factură,  
Valoare)

**Cheie primară:** Nr\_crt

**Cheie străină :** Cod\_furnizor

Acest proces continuă, până când se vor prelucra toate relațiile din baza de date.

*Documentarea relațiilor și a atributelor din cheile străine.*

Relațiile derivate pentru view-urile “Locatari” și “Cheltuieli” se pot vedea în anexa 4.5., la sfârșitul acestui capitol. La crearea relațiilor trebuie actualizat și dicționarul de date, cu atributele nou introduse și cu noile chei primare și alternante.

### *Pasul 2.3. Validarea modelului prin normalizare.*

În acest pas validăm baza de date prin normalizare. Procesul de normalizare este descris amănunțit în capitolul 2.

- Forma Normală Unu (1NF), eliminarea repetițiilor din attribute.
- Forma Normală Doi (2NF), eliminarea dependențelor parțiale de cheia primară.
- Forma Normală Trei (3NF), eliminarea dependențelor tranzitive de cheia primară.
- Forma Normală Boyce-Codd (BCNF), eliminarea anomaliilor care au mai rămas.

Trebuie să analizăm fiecare relație care este edscrisă în anexa 4.5. Verificăm dacă relațiile sunt în forma normală Boyce-Codd, iar dacă găsim unul care nu

satisface această formă normală, ne întoarcem la pașii precedenți și rezolvăm problema.

Pentru a exemplifica acest proces, să analizăm relația *Furnizori Provoacă Cheltuieli*, descrisă în anexa 4.5. Menționăm că notațiile de aici nu includ cheia străină.

**Furnizori** (Cod\_furnizor, Denumire, Cod\_fiscal, Cont, Banca, Str, Nr, Bl, Sc, Ap, Loc, Jud)

**Cheie primară:** Cod\_furnizor

**Cheie alternantă:** Cod\_fiscal

Cod\_furnizor → Denumire, Cod\_fiscal, Cont, Banca, Str, Nr, Bl, Sc, Ap, Loc, Jud

Cod\_fiscal → Cod\_furnizor, Denumire, Cont, Banca, Str, Nr, Bl, Sc, Ap, Loc, Jud

**Cheltuieli** (Nr\_crt, Cod\_furnizor, Cod\_cheltuială, Nr\_factură, Data\_factură, Valoare)

**Cheie primară:** Nr\_crt

Nr\_crt → Cod\_furnizor, Cod\_cheltuială, Nr\_factură, Data\_factură, Valoare

Pentru acest exemplu avem următoarele demonstrații:

- Nu există grupuri repetitive în niciunul dintre entități. Deci relația este în forma normală unu.
- Fiecare cheie se compune dintr-un singur atribut. De aceea nu putem avea dependență parțială de cheie. Deci relația este în formă normală doi.
- Nu există dependențe tranzitive de cheie, deci relația este în formă normală trei.
- Fiecare determinant evidențiat este cheie, deci relația este în formă normală Boyce-Codd.

Acest proces se continuă pentru fiecare relație care apare în anexa 4.5.

#### *Pasul 2.4. Validarea modelului prin tranzacțiile cerute.*

Pentru a valida prin tranzacții un model logic de date, folosim diagrama ER. din figura 4.6.a. respectiv 4.6.b. și documentația întocmită. Încercăm să rezolvăm manual toate tranzacțiile cerute de utilizator, folosind doar datele și relațiile din modelul de date creat. Dacă nu putem rezolva o tranzacție, înseamnă că am omis ceva la modelarea bazei de date și deci trebuie să ne întoarcem la pașii anteriori și să remediem eroarea. Pe de altă parte, dacă o parte a modelului de date nu este folosit la nici o tranzacție, atunci - în caz că nu se prevede ca în viitor să folosim această parte a modelului - va trebui să-l considerăm redundant, și să-l eliminăm din modelul de date.

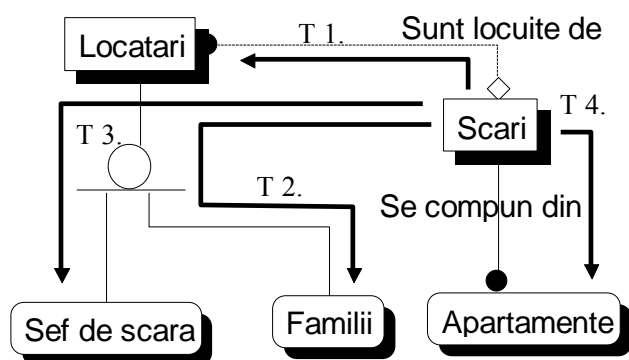
Considerăm două posibilități de verificare a modelului de date. Conform primei metode, ne asigurăm că există în modelul de date toate entitățile și relațiile

care sunt necesare pentru tranzacția aleasă. Pentru a exemplifica această metodă, să verificăm următoarea tranzacție:

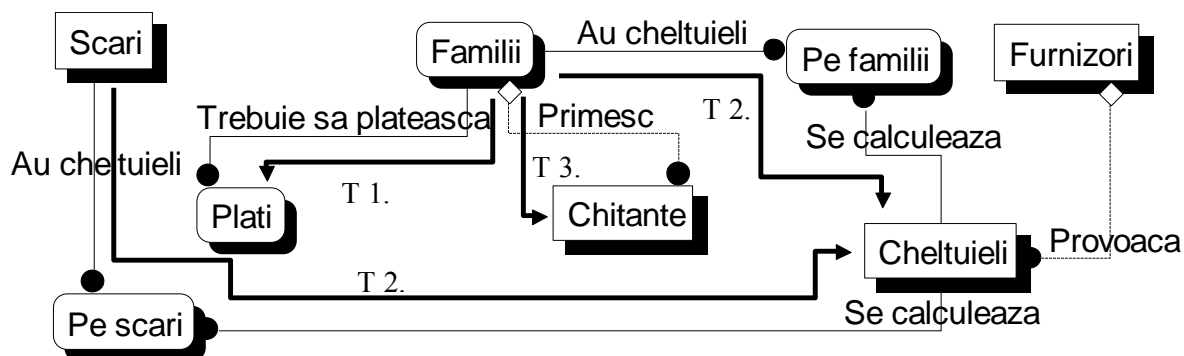
### Crearea și modificarea de înregistrări cu detaliile unei cheltuieli.

- Pentru crearea înregistrării despre o cheltuială, după introducerea datelor despre cheltuială, se vor selecta și scările sau familiile la care este asociată. O cheltuială nu poate fi asociată și unor familii și unor scări.
- Pentru a modifica o cheltuială, trebuie să știm numărul cheltuielii din baze de date. Căutăm acea cheltuială și dacă există modificăm detaliile despre cheltuială, recreind și legăturile la familii sau scări, depinde cine trebuie să plătească. Dacă nu există, atunci apare un mesaj de eroare și procedura de modificare se întrerpe.
- În cazul ștergerii unei cheltuieli, prima dată trebuie căutate înregistrările de legătură a acelor cheltuieli cu scări sau cu familii. Dacă există se șterg, după care se șterge și cheltuiala. Dacă nu există înseamnă că nu există nici cheltuiala și se va afișa un mesaj de eroare.

A doua modalitate de a verifica modelul de date este de a desena căile generate de relații, pentru fiecare tranzacție în parte. Aceste căi se notează cu o linie având o săgeată în direcția tranzacției. Tranzacțiile descrise în capitolul 4.1.2. sunt reprezentate prin diagramele din figurile 4.7.a. respectiv 4.7.b.



**Figura 4.7.a.**  
Diagrama tranzacțiilor  
cerute la view-ul



**Figura 4.7.b.** Tranzacțiile cerute pentru view-ul “Cheltuieli”.

Diagrama ER din figura 4.7.a. și 4.7.b. ne ajuta să identificăm ușor relațiile critice pentru tranzacții, precum și ariile din modelul creat, care nu iau parte la nici o tranzacție. După identificarea și eliminarea acestor arii, redesenăm diagrama ER.

În cazul nostru această diagramă nu se modifică.

#### *Pasul 2.6. Definirea regulilor de integritate.*

În acest pas identificăm acele reguli care să ne garanteze că datele introduse în baza de date rămân consistente. Putem identifica cinci tipuri de reguli:

- necesitatea datelor,
- domeniile atributelor,
- integritatea entităților,
- integritatea relațiilor,
- reguli date de întreprindere.

##### *Necesitatea datelor.*

Identificăm acele attribute ale bazei de date, care nu admit valoarea nulă. De exemplu attributele Nr\_mat - din entitatea Locatari - și Adresa (Nr\_bloc, Scara) - din entitatea Scări.

Aceste reguli asupra atributelor modelului sunt descrise în pasul 1.3. și sunt documentate în anexa 4.2.

##### *Domeniile atributelor.*

Domeniile atributelor identifică valorile posibile pe care le poate lua un atribut. De exemplu atributul Etaj din entitatea locatari poate lua valori între -1 și 100. Domeniile atributelor sunt descrise în pasul 1.4. și sunt documentate în anexa 4.4.

##### *Integritatea entităților.*

Cheia primară a entităților nu poate lua valori nule. De exemplu Nr\_mat, cheia primară a entității Locatari nu poate lua valoarea nulă. Cheile primare ale fiecărei entități au fost identificate în pasul 1.6. iar documentarea cheilor se găsește în anexa 4.5.

##### *Integritatea relațiilor.*

Relațiile dintre entități sunt reprezentate prin copierea chei primare a entității “părinte”, în cheia străină a entității “fiu”. Integritatea relațiilor se referă la faptul că dacă cheia străină dintr-o entitate slabă conține o valoare, atunci această valoare să exista și în entitatea tare la care este asociat prin relație.

De exemplu dacă se înregistrează plăți unei familii în entitatea Plăți, atunci acea familie trebuie să existe și în entitatea Familii. Attributele care compun cheile primare și cheile străine sunt descrise în anexa 4.5.

Este important ca să identificăm regulile relațiilor, pentru ca relațiile dintre entități să rămână consistente. Menționăm că relațiile se descriu cu limbajul specific definirii bazelor de date (DBDL).

Prima dată să considerăm cazul că cheile străine au valoare nulă. În general când participarea entității slabe la relație este totală, nu se permite valoare nulă în cheia străină. De exemplu să descriem relația Scări *Se compun din* Apartamente. Entitatea Apartamente (slabă) participă total la relație, deci nu admitem valori nule atributelor din cheia străină și anume Nr\_bloc și Scara.

Pe de altă parte, dacă entitatea slabă participă parțial la relație, putem admite valoare nulă și cheii străine. Nu este cazul exemplului nostru, pentru că nu avem entitate slabă cu participare parțială.

Descriem relația Scări *Se compun din* Apartamente cu limbajul DBDL:

**Scări** (Nr\_bloc, Scara, Lift)

**Cheie primară:** Nr\_bloc, Scara

**Apartamente** (Nr\_bloc, Scara, Apartament, Suprafața, Cutii\_poștale, Nr\_prize\_tv)

**Cheie primară:** Nr\_bloc, Scara, Apartament

**Cheie străină :** Nr\_bloc, Scara **Nenulă referindu-se la** Scări (Nr\_bloc, Scara)

În continuare considerăm cazul în care se modifică cheia primară a entității tari. Menționăm că inserarea cheii primare, sau ștergerea cheii străine nu dăunează consistenței bazei de date.

Pentru celelalte chei primare, definim modul de modificare și ștergere a cheii primare. Putem alege între patru moduri: FĂRĂ ACȚIUNE, CASCADĂ, SETARE NULĂ și SETARE IMPLICITĂ. Descriem aceste modalități de modificare/ștergere pe baza relației Scări *Se compun din* Apartamente, extinzând în continuare limbajul DBDL.

**Scări** (Nr\_bloc, Scara, Lift)

**Cheie primară:** Nr\_bloc, Scara

**Apartamente** (Nr\_bloc, Scara, Apartament, Suprafața, Cutii\_poștale, Nr\_prize\_tv)

**Cheie primară:** Nr\_bloc, Scara, Apartament

**Cheie străină :** Nr\_bloc, Scara **Nenulă, Cascadă,**  
**referindu-se la** Scări (Nr\_bloc, Scara)

Această operație se execută pentru fiecare relație descrisă în anexa 4.5.

*Regulile întreprinderii.*

Aceste reguli sunt definite de întreprindere și reprezintă reguli ce trebuie respectate și în viața de zi cu zi. De exemplu lista cu plățile locatarilor se poate tipări doar pe durată de o lună. Toate aceste reguli sunt prezentate în anexa 4.6.

*Documentarea regulilor de integritate.*

Toate cele descrise mai sus se documentează în modelul de date.

*Pasul 2.7. Revizuirea modelului de date cu ajutorul utilizatorului.*

În acest pas revizuim modelul creat cu ajutorul utilizatorului. Este foarte important ca modelul să fie o reprezentare fidelă a realității. Utilizatorul sistemului trebuie să verifice atât diagrama cât și documentația întocmită. Dacă se găsește o eroare, se vor reface pașii necesari pentru corectarea erorii.

### **Pasul 3. Crearea și validarea modelului global de date.**

În acest pas creăm modelul global prin compunerea entităților de pe diversele modele locale. Acest model global va trebui și ea validat prin normalizare.

*Pasul 3.1. Compunerea modelelor locale în modelul global.*

În pasul acesta compunem două modele locale într-un model global de date. Mai multe modele globale se compun tot două câte două, până când ajungem la modelul global. Pentru început identificăm similaritățile dintre modelele locale, după care trebuie identificate și rezolvate ariile de conflicte între modele iar în final se trec toate modelele într-un singur model. Pașii care trebuie urmați pentru compunerea modelelor sunt prezentați în continuare:

*(1) Revizuirea numelor entităților și cheile lor primare.*

Comparăm numele și cheile primare dintre două modele locale, comparație ce este prezentată în tabela 4.8.

<b>Tip de entitate (view-ul Locatari)</b>	<b>Cheie primară</b>	<b>Tip de entitate (view-ul Cheltuieli)</b>	<b>Cheie primară</b>
Scări	Nr_bloc, Scară	Scări	Nr_bloc, Scară
Familii	Nr_mat	Familii	Nr_mat
Locatari	Nr_mat		
Șef de scară	Nr_mat		
Apartamente	Nr_bloc, Scara, Ap.		
		Plăți	Data, Nr_mat
		Chitanțe	Nr_chit
		Pe_familii	Nr_crt
		Pe_scări	Nr_crt
		Cheltuieli	Nr_crt
		Furnizori	Cod_furnizor

**Tabela 4.8.** O comparație între numele entităților și ale cheilor lor primare.

Aceste similarități dintre entități arată care sunt acele modele care se suprapun. Entitățile care se suprapun se pot compune, creând o singură entitate cu toate atributele entității din cele două modele.

(2) *Revizuirea numelor relațiilor.*

În continuare, comparăm numele relațiilor din două modele de date. O comparație între modelele Locatari și Cheltuieli este prezentată în tabela 4.9. Relațiile sunt incluse în această tabelă doar o singură dată în direcția de la entitatea tare la cea slabă.

Tip entitate view-ul Locatari	Tip de relație	Tip entitate view-ul Locatari	Tip entitate view-ul Cheltuieli	Tip de relație	Tip entitate view-ul Cheltuieli
Scări	<i>Sunt locuite de</i>	Locatari	Scări	<i>Au cheltuieli</i>	Pe_scări
	<i>Se compun din</i>	Apartamente	Familii	<i>Au cheltuieli</i>	Pe_familii
				<i>Trebuie să plătească</i>	Plăți
				<i>Primesc</i>	Chitanțe
			Cheltuieli	<i>Se calculează</i>	Pe_familii
				<i>Se calculează</i>	Pe_scări
			Furnizori	<i>Provoacă</i>	Cheltuieli

**Tabela 4.9.** Comparația tipurilor de relații.

Informațiile cuprinse în această tabelă arată încă o dată aria modelelor care se suprapun. Este important să înțelegem că același nume se relație în două modele nu înseamnă că acele relații au și același rol. Trebuie să avem mare grijă la astfel de nume de relații (se numesc homonome). Mai există și posibilitatea ca relații care reprezintă aceleași concepte să fie denumite cu nume diferite (sinonime).

Aceste probleme la identificarea relațiilor se pot rezolva, analizând atributele (și în particular domeniile cheilor) asociate entităților care fac parte din relație.

În final trebuie să ne asigurăm că relațiile pe care le considerăm echivalente să reprezinte aceleași relații și în “lumea reală”.

(3) *Compunerea entităților din cele două modele.*

În acest pas analizăm numele și conținutul entităților din ambele modele. În particular, utilizăm cheia primară să vedem care dintre entități sunt echivalente, chiar dacă apar sub nume diferite în cele două modele.

Acest pas include următoarele activități:

- Compunerea entităților cu aceleași nume și aceleași chei primare.
- Compunerea entităților cu aceleași nume dar cu chei primare diferite.

- Compunerea entităților cu nume diferite dar cu chei primare aceleași sau diferite.

*Compunerea entităților cu aceleași nume și aceleași chei primare.*

Entitățile care sunt denumite cu aceleași nume și aceleași chei primare în cele două modele reprezintă în general aceleași concepte a “lunii reale”. Deci aceste entități sunt cele mai sumpu de identificat. Astfel de entități sunt de exemplu entitățile Scări și Familii.

Entitățile combinate vor include toate atributele celor două entități din cele două modele, eliminându-se dublurile. Exemplul de compunere a entității Familii este prezentată în figura 4.10.

*Compunerea entităților cu aceleași nume dar cu chei primare diferite.*

În cazurile când entitățile au același nume dar cu chei primare diferite, trebuie să identificăm și cheile alternante a celor două entități. Dacă între cheile candidat a unei entități apare cheia primară a celeilalte, atunci compunem entitățile și alegem o cheie primară dintre cele două cei primare.

În exemplul nostru nu avem astfel de caz.

*Compunerea entităților cu nume diferite dar cu chei primare aceleași sau diferite.*

Putem întâlni cazuri în care nici numele entităților și nici cheile primare șă nu fie la fel, dar știm din atributele celor două entități că ele reprezintă aceleași concepte. Aceste entități se compun ca la cazul anterior, având în plus obligația de a alege un nume care poate să fie una dintre cele două nume, sau o nume nouă.

**Figura 4.10.** Exemplu de compunere a două entități.

**(view-ul Locatari)**

**Familii** (Nr\_mat, Nr\_pers, Nr\_pers\_prez, Nr\_chei)

**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_mat **referindu-se la** Locatari (Nr\_mat)

**(view-ul Cheltuieli)**

**Familii** (Nr\_mat, Fond\_rulment, Fond\_reparații, Alte\_fonduri)

**Cheie primară:** Nr\_mat



(modelul global)

**Familii** (Nr\_mat, Nr\_pers, Nr\_pers\_prez, Nr\_chei, Fond\_rulment, Fond\_reparații, Alte\_fonduri)

**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_mat **referindu-se la** Locatari (Nr\_mat)

*(4) Includerea (fără compunere) a entităților unice în cele două modele.*

După includerea tuturor entităților comune în modelul global, mai rămân alte entități care nu sunt comune și care încă nu au fost incluse. Aceste entități se vor



include neschimbat în modelul global. Astfel de entități sunt: Cheltuieli din modelul Cheltuieli, respectiv Apartamente din modelul Locatari.

*(5) Compunerea tipurilor de relații din modelele locale.*

În acest pas analizăm numele și caracteristicile relațiilor pentru a le putea compune. Este important de rezolvat conflictele datorate din participarea și cardinalul relațiilor. Numele relațiilor din cele două modele sunt listate în tabela 4.9.

*Compunerea relațiilor cu aceleași nume și aceleași caracteristici.*

Aceste relații sunt cel mai ușor de identificat. Compunerea se rezolvă prin simpla înscriere a relației în modelul global. Pot exista situații când cardinalul sau participarea nu coincid la cele două modele. Cazul acesta trebuie clarificată cu utilizatorul sistemului.

Atragem atenția că pot exista relații cu aceleași denumiri dar care să aibă roluri diferite (omonime).

*Compunerea relațiilor cu nume diferite dar cu aceleași caracteristici.*

Identificăm acele relații care apar în ambele modele dar cu denumiri diferite. Această identificare devine evidentă după ce observăm că relația leagă aceleași entități. Și în acest caz trebuie rezolvată problema cardinalului și a participării.

*(6) Includerea (fără compunere) a relațiilor unice pe cele două modele.*

Identificăm toate relațiile pe care nu am inclus-o încă și le includem neschimbat în modelul global. Toate relațiile descrise în tabela 4.9. sunt de acest tip.

*(7) Verificarea dacă s-a omis vre-o entitate sau relație.*

Această acțiune de verificare este foarte importantă dar și foarte grea de rezolvat. Entitățile și relațiile compuse pot avea chiar alt nume decât cele de pe modelele locale ceea ce face greu de urmărit includerea entității sau a relației în modelul global.

*(8) Verificarea cheilor străine.*

În acest pas verificăm dacă toate entitățile slabe conțin cheia străină asociată lor. Aceste chei trebuie să se formeze la compunerea entităților, însă tot la compunere se pot și redenumii.

*(9) Verificăm regulile de integritate.*

Verificăm toate regulile de integritate pe modelul global de date iar dacă apar probleme la verificare, ne întoarcem la pașii anteriori și le remediem.

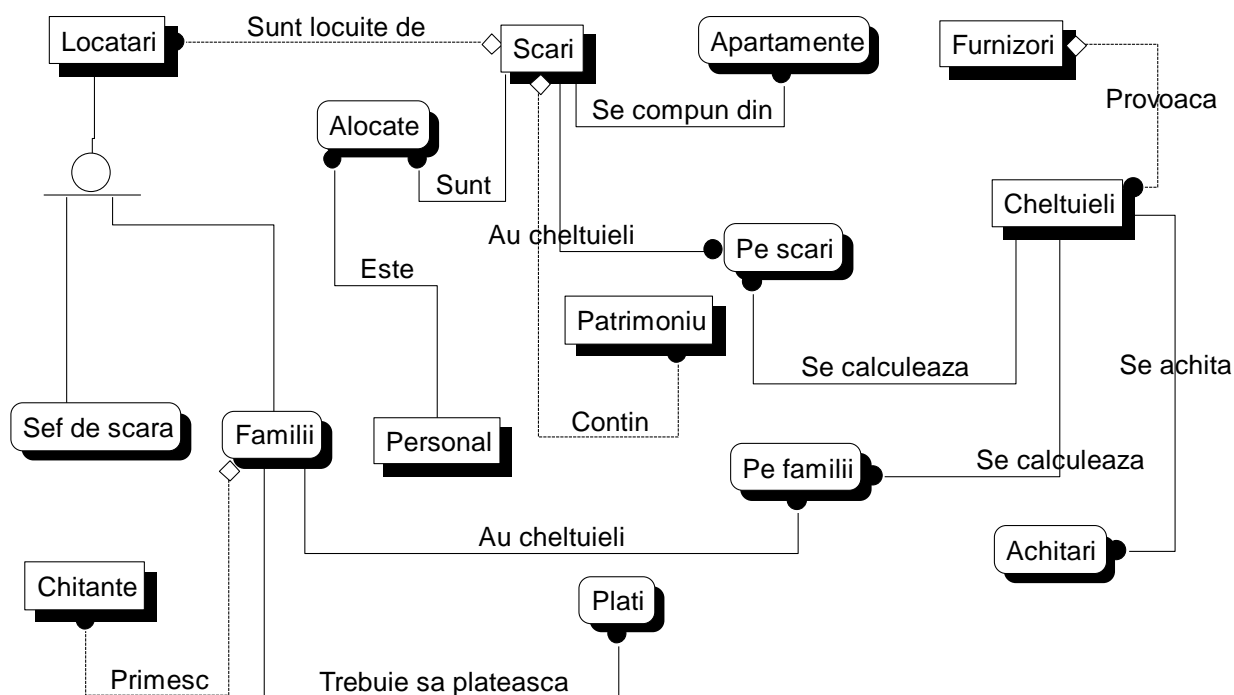
*(10) Desenarea modelului global de date.*

Acum desenăm modelul global de date. Modelul global al sistemului de evidență a asociației de locatari este prezentată în figura 4.11.

Numele entităților și a relațiilor se poate schimba la această acțiune.

*(11) Completarea documentației.*

Este foarte importantă actualizarea documentației, pentru a reflecta modificările efectuate asupra bazei de date. Anexa 4.7. descrie relațiile modelului global prezentat în figura 4.11., folosind limbajul DBDL.



**Figura 4.11.** Diagrama globală și finală ER.

### *Pasul 3.2. Validarea modelului global de date.*

În același mod cum am validat modelele locale, trebuie validat și modelul global, folosind normalizarea și validarea prin tranzații. Acesta este foarte important pentru a verifica (și demonstra) dacă nu cumva s-au introdus erori la crearea modelului global.

### *Pasul 3.3. Verificarea posibilității de extindere în viitor.*

Este important ca modelul creat să se poată extinde în viitor. De exemplu în sistemul de evidență a asociației de locatari se va putea introduce un modul care să rezolve salarizarea personalului asociației. Modelul global al aplicației este extensibilă, extinderea putându-se realiza cu minime modificări.

### *Pasul 3.4. Desenarea diagramei ER finale.*

La acest pas observăm că nu am modificat nimic de la ultima diagramă desenată, deci diagrama ER finală este diagrama din figura 4.11.

### *Pasul 3.5. Verificarea modelului global cu ajutorul utilizatorului.*

Este important să reexaminăm modelul creat împreună cu utilizatorul, pentru a vedea dacă îndeplinește toate cerințele. Dacă apar cerințe care nu sunt îndeplinite ne întoarcem la pașii anteriori și remediem situația.



## Anexa 4.1. Documentarea tipurilor de entități în view-urile Locatari și Cheltuieli

Tipurile de entități din view-ul “Locatari”:

<b>Nume tip de entitate</b>	<b>Descriere</b>	<b>Aliasuri</b>	<b>Entități</b>
Locatari	Oamenii care locuiesc în asociație		Persoanele care au domiciliul în asociație
Familii	Famiiliile din asociație	Cap_familie	Câte o persoană din fiecare familie, considerată reprezentantul acesteia
Sef_Scara	Persoana care se ocupă cu problemele unei scări.		Câte o persoană de pe fiecare scară din asociație
Scări	Scările din asociație		Fiecare scară care aparține asociației, împreună cu caracteristicile ei.
Apartamente	Apartamentele din asociație		Fiecare apartament din asociație, împreună cu caracteristicile ei.

Tipurile de entități din view-ul “Cheltuieli”:

<b>Nume tip de entitate</b>	<b>Descriere</b>	<b>Aliasuri</b>	<b>Entități</b>
Scări	Scările din asociație		Analog ca la “Locatari”
Familii	Famiiliile din asociație	Cap_familie	Analog ca la “Locatari”
Furnizori	Societățile de la care vin facturile		Fiecare furnizor cu care s-a lucrat vreodată în asociație
Cheltuieli	Cheltuielile asociației către furnizori		Fiecare factură, care reprezintă o cheltuială a locatarilor din asociație.
Plăți	Plățile calculate		Plățile calculate pentru fiecare familie
Chitanțe	Chitanțele emise		Toate chitanțele emise către locatari.

## Anexa 4.2. Documentarea atributelor

Atributele tipurilor de entități din view-ul “Locatari”:

Tip de entitate	Atribute	Descriere	Tip de date și lungime	Reguli	Valoare implicită	Alias	Valoare nulă	Derivat?
Locatari	Nr_Mat Etaj Apartament Nume	Det. unic locatarii Etajul la care loc. Apartamentul Numele locatarului	Întreg Întreg [-1,100] Întreg 25 de caractere	Cheie primară			Nu Nu Nu Nu	Nu Nu Nu Nu
Familii	Ca la Locatari, plus Nr_pers Nr_pers_prez  Nr_chei	Nr. pers. în familie Nr. pers. prezente în luna curentă Nr. de chei de la ușa principală	Ca la Locatari plus Întreg Întreg  Întreg	Ca la Locatari			Nu Nu  Da	Nu Nu  Nu
Sef_Scara	Ca la Locatari plus DataIntrFunc		Ca la Locatari plus Dată	Ca la Locatari			Nu	Nu
Scări	Adresa Lift	(Nr_bloc, Scara) Există sau nu lift	Logic	Cheie primară			Nu	Nu
Apartamente	Apartament Suprafața Cutii_poștale	Apartamentul Sup. totală a ap. Nr. cutii poștale	Întreg Real Întreg				Nu Nu Nu	Nu Nu Nu

	Nr_prize_tv	Nr. prize tv.	Întreg				Nu	Nu
--	-------------	---------------	--------	--	--	--	----	----

Atributele tipurilor de entități din view-ul “Cheltuieli”:

Tip de entitate	Atribute	Descriere	Tip de date și lungime	Reguli	Valoare implicită	Alias	Valoare nulă	Derivat?
Familii	Nr_mat	Identifică familia	Întreg	Cheie primară			Nu	Nu
	Fond_rulment	Fond de rulment	Real				Nu	Nu
	Fond_reparații	Fond de reparații	Real				Nu	Nu
	Alte_fonduri	Alte fonduri	Real				Nu	Nu
Plăți	Data	Luna pt. care e plata	Data				Nu	Nu
	Valoare	Suma de plată	Real				Nu	Nu
	Restanță	Suma restantă	Real				Nu	Nu
Chitanțe	Nr_chit	Id. unic chitanța	Întreg	Cheie primară			Nu	Nu
	Valoare	Valoarea plătită	Real				Nu	Nu
	Data	Data plății	Data				Nu	Nu
Cheltuieli	Nr_crt	Identifică cheltuiala	Întreg	Cheie primară			Nu	Nu
	Cod_cheltuiala	Tipul cheltuielii	Întreg				Nu	Nu
	Nr_factura	Numărul facturii	Întreg				Nu	Nu
	Data_factura	Data facturii	Data				Nu	Nu
	Valoare	Valoarea cheltuielii	Real				Nu	Nu
Furnizori	Cod_furnizor	Identifică furnizorul	Întreg	Cheie primară			Nu	Nu
	Denumire	Numele societății	30 de caractere				Nu	Nu
	Cod_fiscal	Codul fiscal	10 caractere	Cheie alt.			Nu	Nu
	Cont	Contul bancar	25 de caractere				Da	Nu
	Banca	Banca	20 de caractere				Da	Nu
	Adresa	Str,Nr,BI,Sc,Ap,Loc						

### **Anexa 4.3. Documentarea tipurilor de relații din view-urile “Locatari” și “Cheltuieli”**

Tipuri de relații din view-ul “Locatari”:

Tip de entitate	Tip de relație	Tip de entitate	Cardinal	Participare*
Scări	sunt locuite de	Locatari	1:M	P:T
	sunt locuite de	Familii	1:M	P:T
	sunt conduse de	Șef de scară	1:M	P:T
	se compun din	Apartamente	1:M	T:T

Tipuri de relații din view-ul “Cheltuieli”:

Tip de entitate	Tip de relație	Tip de entitate	Cardinal	Participare*
Furnizorii	provoacă	Cheltuieli	1:M	P:T
Celtuieli	sunt plătite de	Familii	M:N	P:P
	sunt plătite de	Scări	M:N	P:P
Familii	trebuie să plătească	Plăți	1:M	T:T
	primesc	Chitanțe	1:M	P:T

\* P=parțial, T=total.

### **Anexa 4.4. Documentarea domeniilor atributelor**

Nume domeniu	Caracteristici	Exemple
Etaj	Întreg între -1 și 100	0=Parter
String30	30 de caractere, lungime variabilă	ROMGAZ
String10	10 caractere, lungime variabilă	
String20	20 de caractere, lungime variabilă	



### **Anexa 4.5. Descrierea relațiilor din view-urile “Locatari” și “Cheltuieli”**

Descrierea relațiilor din view-ul “Locatari”.

**Locatari** (Nr\_mat, Nr\_bloc, Scara, Etaj, Apartament, Nume, Nr\_pers, Nr\_pers\_prez, Nr\_chei, Data\_intrare\_func)

**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_bloc, Scara **referindu-se la** Scări (Nr\_bloc, Scara)

**Scări** (Nr\_bloc, Scara, Lift)

**Cheie primară:** Nr\_bloc, Scara

**Apartamente** (Nr\_bloc, Scara, Apartament, Suprafața, Cutii\_poștale, Nr\_prize\_tv)

**Cheie primară:** Nr\_bloc, Scara, Apartament

**Cheie străină :** Nr\_bloc, Scara **referindu-se la** Scări (Nr\_bloc, Scara)

Descrierea relațiilor din view-ul “Cheltuieli”:

**Scări** (Nr\_bloc, Scara)

**Cheie primară:** Nr\_bloc, Scara

**Familii** (Nr\_mat, Fond\_rulment, Fond\_reparații, Alte\_fonduri)

**Cheie primară:** Nr\_mat

**Plăți** (Data, Nr\_mat, Valoare, Restanță)

**Cheie primară:** Data, Nr\_mat

**Cheie străină :** Nr\_mat **referindu-se la** Familii (Nr\_mat)

**Chitanțe** (Nr\_chit, Nr\_mat, Valoare, Data)

**Cheie primară:** Nr\_chit

**Cheie străină :** Nr\_mat **referindu-se la** Familii (Nr\_mat)

**Furnizori** (Cod\_furnizor, Denumire, Cod\_fiscal, Cont, Banca, Str, Nr, Bl, Sc, Ap, Loc, Jud)

**Cheie primară:** Cod\_furnizor

**Cheie alternantă:** Cod\_fiscal

**Cheltuieli** (Nr\_crt, Cod\_furnizor, Cod\_cheltuială, Nr\_factură, Data\_factură, Valoare)

**Cheie primară:** Nr\_crt

**Cheie străină :** Cod\_furnizor **referindu-se la** Furnizori (Cod\_furnizor)

**Pe\_familii** (Nr\_crt, Nr\_mat)

**Cheie primară:** Nr\_crt

**Cheie străină :** Nr\_crt **referindu-se la** Cheltuieli (Nr\_crt)

**Cheie străină :** Nr\_mat **referindu-se la** Familii(Nr\_mat)

**Pe\_scări** (Nr\_crt, Nr\_bloc, Scara)

**Cheie primară:** Nr\_crt

**Cheie străină :** Nr\_crt **referindu-se la** Cheltuieli (Nr\_crt)

**Cheie străină :** Nr\_bloc, Scara **referindu-se la** Scări (Nr\_bloc, Scara)

***Anexa 4.6. Regulile date de întreprindere în cazul modelului  
“Locatari” și “Cheltuieli”***

- (1) Lista de plăți se elaborează doar pe o lună întreagă.
- (2) Fiecare familie trebuie să declare la începutul lunii dacă va lipsii un membru al familiei toată luna.
- (3) Nici o familie nu va putea să se mute până când nu si-a plătit toate datoriile.

### **Anexa 4.7. Modelul global de date al sistemului Asociația de Locatari**

**Locatari** (Nr\_mat, Nr\_bloc, Scara, Etaj, Apartament, Nume)

**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_bloc, Scara **Nenulă, referindu-se la** Scări (Nr\_bloc, Scara)  
la ștergere și la modificare **Cascadă.**

**Familii** (Nr\_mat, Nr\_pers, Nr\_pers\_prezente, Nr\_chei, Fond\_rulment,  
Fond\_reparații, Alte\_fonduri)

**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_mat **Nenulă, referindu-se la** Locatari (Nr\_mat)  
la ștergere și modificare **Cascadă.**

**Șef\_de\_scară** (Nr\_mat, Data\_intrare\_func)

**Cheie primară:** Nr\_mat

**Cheie străină :** Nr\_mat **Nenulă, referindu-se la** Locatari (Nr\_mat)  
la ștergere și modificare **Cascadă.**

**Scări** (Nr\_bloc, Scara, Lift)

**Cheie primară:** Nr\_bloc, Scara

**Apartamente** (Nr\_bloc, Scara, Apartament, Suprafața, Cutii\_poștale, Nr\_prize\_tv)

**Cheie primară:** Nr\_bloc, Scara, Apartament

**Cheie străină :** Nr\_bloc, Scara **Nenulă, referindu-se la** Scări (Nr\_bloc, Scara)  
la ștergere și modificare **Cascadă.**

**Plăți** (Data, Nr\_mat, Valoare, Restanță)

**Cheie primară:** Data, Nr\_mat

**Cheie străină :** Nr\_mat **Nenulă, referindu-se la** Familii (Nr\_mat)  
la ștergere **Fără acțiune**, la modificare **Cascadă.**

**Chitanțe** (Nr\_chit, Nr\_mat, Valoare, Data)

**Cheie primară:** Nr\_chit

**Cheie străină :** Nr\_mat **Nenulă, referindu-se la** Familii (Nr\_mat)  
la ștergere **Fără acțiune**, la modificare **Cascadă.**

**Furnizori** (Cod\_furnizor, Denumire, Cod\_fiscal, Cont, Banca, Strada, Nr, Bl, Sc,  
Ap, Localitate, Judet)

**Cheie primară:** Cod\_furnizor

**Cheie alternantă:** Cod\_fiscal

**Cheltuieli** (Nr\_crt, Cod\_furnizor, Cod\_cheltuială, Nr\_factură, Data\_factură, Valoare\_factură)

**Cheie primară:** Nr\_crt

**Cheie străină :** Cod\_furnizor **Nenulă, referindu-se la** Furnizori (Cod\_furnizor) la ștergere **Fără acțiune**, la modificare **Cascadă**.

**Pe\_familii** (Nr\_crt, Nr\_mat)

**Cheie primară:** Nr\_crt

**Cheie străină :** Nr\_crt **Nenulă, referindu-se la** Cheltuieli (Nr\_crt) la ștergere și modificare **Cascadă**.

**Cheie străină :** Nr\_mat **Nenulă, referindu-se la** Familii(Nr\_mat) la ștergere și modificare **Cascadă**.

**Pe\_scări** (Nr\_crt, Nr\_bloc, Scara)

**Cheie primară:** Nr\_crt

**Cheie străină :** Nr\_crt **Nenulă, referindu-se la** Cheltuieli (Nr\_crt) la ștergere și modificare **Cascadă**.

**Cheie străină :** Nr\_bloc, Scara **Nenulă, referindu-se la** Scări (Nr\_bloc, Scara) la ștergere și modificare **Cascadă**.

**Personal** (Nr\_matricol, Nume, Data\_nașterii, Meseria, Data\_angajării)

**Cheie primară:** Nr\_matricol

**Alocate** (Nr\_matricol, Nr\_bloc, Scara)

**Cheie primară:** Nr\_matricol, Nr\_bloc, Scara

**Cheie străină :** Nr\_matricol **Nenulă, referindu-se la** Personal (Nr\_matricol) la ștergere și modificare **Cascadă**.

**Cheie străină :** Nr\_bloc, Scara **Nenulă, referindu-se la** Scări (Nr\_bloc, Scara) la ștergere și modificare **Cascadă**.

**Patrimoniul** (Nr\_inventar, Nr\_bloc, Scara, Denumire, Inv\_fix, Valoare)

**Cheie primară:** Nr\_inventar

**Cheie străină :** Nr\_bloc, Scara **Nenulă, referindu-se la** Scări (Nr\_bloc, Scara) la ștergere și modificare **Cascadă**.

**Achitări** (Nr\_crt, Nr\_doc, Tip\_op, Valoare\_achit, Data)

**Cheie primară:** Nr\_Doc, Tip\_Op

**Cheie străină :** Nr\_crt **Nenulă, referindu-se la** Cheltuieli (Nr\_crt) la ștergere și modificare **Cascadă**.