5. MODELUL RELAȚIONAL

O problema fundamentala a unui SGBD este modul in care datele sunt organizate in vederea stocarii si exploatarii lor de catre aplicatii. Din punct de vedere istoric, in anii '60 au existat doua modele de organizare a datelor care au fost apoi abandonate din cauza problemelor pe care le generau:

- Modelul ierarhic, folosit de IBM in sistemul IMS (care inca este unul dintre produsele furnizate de aceasta firma), in care organizarea este sub forma arborescenta: nodurile contin date si legaturi ('pointeri') catre nodurile fiu (vezi http://www-306.ibm.com/software/data/ims/
- Modelul retea. In cadrul acestuia inregistrarile sunt structurate sub forma unui graf orientat, fiecare nod putand avea mai multe inregistrari 'tata' si mai multi fii. Au existat mai multe sisteme de gestiune si limbaje de programare dezvoltate pe baza acestui model (de exemplu limbajul COBOL).

Dezavantajul principal al acestor doua modele este ca accesul la o inregistrare necesita navigarea prin arbore sau graf pentru a o localiza. Din acest motiv apar o serie de probleme mai ales legate timpul necesar scrierii de noi programme si a detectarii anomaliilor care pot sa apara in proiectarea bazei de date.

Modelul relational al datelor, folosit in acest moment de majoritatea covarsitoare a sistemelor de gestiune aflate pe piata a fost intrudus in anul 1970 prin articolul lui Edgar Frank Codd "***A relational model for large shared databanks***". Acest model, in care datele sunt stocate sub forma tabelara are o serie de avantaje care au dus la inlaturarea celorlalte modele:

- Datele sunt stocate doar ca valori; nu exista pointeri sau navigare prin date;
- Face posibila dezvoltarea de limbaje de cereri de nivel inalt in care utilizatorul specifica ***ce*** date doreste si nu ***cum*** se ajunge la rezultat, modul in care este calculat acesta fiind in sarcina sistemului de gestiune (exemplu de astfel de limbaj: SQL)
- Furnizeaza o baza solida pentru problemelor de corectitudine a datelor (redundanta, anomalii, etc).
- Permite tratarea problemelor de independenta a datelor (discutate in capitolul 1).
- Este extensibil, putand fi utilizat si pentru modelarea si manipularea de date complexe.

3.1. Elementele de baza ale modelului

3.1.1. Domeniu

Definitie: *Domeniu* (eng. Domain) = o multime de valori avand asociat un nume.

Un domeniu se poate defini fie prin enumerarea elementelor sale fie prin specificarea unor caracteristici definitorii ale acestora.
Exemple:

- Culori = {rosu, galben, albastru, violet, verde}
- Nota = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} sau Nota = $\{n \in N^* \mid n \geq 1 \text{ si } n \leq 10\}$
- Sir40 = {Multimea sirurilor de maxim 40 de caractere}
- Numar = {Multimea numerelor intregi pozitive din intervalul [0, 100000]}

Din teoria multimilor se cunoaste notiunea de ***produs cartezian*** al unor multimi: fiind date ***n*** domenii $D_1, D_2, \ldots, D_n$ produsul lor cartezian este:

$$D_1 \times D_2 \times \ldots \times D_n = \{ (v_1, v_2, \ldots, v_n) \mid vi \in D_i , i = 1, \ldots, n\}$$

Trebuie mentionat ca in sirul de domenii care participa la un produs cartezian unele se poate gasi in mod repetat:

$$PC = Numar \times Sir40 \times Numar \times Numar \times Sir40 \times Sir40$$

3.1.2. Relatie

Definitie: *Relatie* (eng. Relation) = o submultime a unui produs cartezian avand asociat un nume.

Termenul de relatie provide de asemenea din matematica. Un exemplu de relatie apartinand produsului cartezian PC din paragraful urmator este:
Produse = { (101, 'Imprimanta laser', 30, 20, 'Xerox', 'Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti',) , (105, 'Calculator PC', 20, 23, 'IBM', 'Bd. D.Cantemir nr.1, Bucuresti'), (124, 'Copiator', 10, 20, 'Xerox', 'Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti',) }

Elementele unei relatii sunt denumite in literatura de specialitate ***tupluri*** (engl. tuple). Relatia de mai sus contine doar 3 dintre elementele produsului cartezian din care provine (3 tupluri).

O reprezentare intuitiva pentru relatia de mai sus este si urmatoarea, in care fiecare element al relatiei devine o linie a unei tabele si fiecare coloana corespunde unui domeniu din produsul cartezian de baza:

Produse

101	Imprimanta laser	30	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
105	Calculator PC	20	23	IBM	Bd. D.Cantemir nr.1, Bucuresti
124	Copiator	10	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti

In fapt deci o relatie se reprezinta o tabela care contine date, fiecare coloana avand asociat un anumit tip de date, dat de domeniul din care provine.

3.1.3. Atribut

Deoarece o relatie are o reprezentare tabelara putem vorbi de 'coloană a unei relatii'. In mod obisnuit, intr-o tabela coloanele au un nume.

Definitie: *Atribut* (eng. Attribute) = coloană a unei relatii avand asociat un nume.

Pentru relatia Produse putem fixa de exemplu urmatoarele nume de atribute:

- `IdP` – Codul produsului (nu exista doua produse avand acelasi cod)
- `NumeP` – numele produsului
- `Qty` – Cantitate
- `IdF` – Codul furnizorului (nu exista doi furnizori avand acelasi cod)
- `NumeF` – Numele furnizorului
- `AdresaF` – Adresa furnizorului

Produse

IdP	NumeP	Qty	IdF	NumeF	AdresaF
101	Imprimanta laser	30	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
105	Calculator PC	20	23	IBM	Bd. D.Cantemir nr.1, Bucuresti
124	Copiator	10	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti

3.1.4. Schema unei relatii

Continutul unei relatii (vazuta ca o tabela) poate varia in timp: se pot adauga sau sterge linii sau se pot modifica unele dintre valorile din liniile existente. Ceea ce ramane constanta este structura relatiei: numele relatiei, numarul si tipul atributelor sale. In terminologia relationala structura unei relatiei este denumita si schema relatiei.

Definitie: *Schema unei relatii* (eng. Relation scheme) = numele relatiei urmat de lista atributelor sale si (eventual) de domeniul din care acestea provin.

Exista mai multe modalitati prin care se poate specifica schema unei relatii. In exemplele urmatoare prezentam cateva dintre acestea cu referire la relatia Produse:

```
Produse(IdP, NumeP, Qty, IdF, NumeF, AdresaF)
Produse(IdP: Numar, NumeP: Sir40, Qty: Numar, IdF: Numar,
        NumeF: Sir40, AdresaF: Sir40)
Produse = IdP, NumeP, Qty, IdF, NumeF, AdresaF
```

In cazul prezentarii unora dintre elementele de teorie a bazelor de date relationale se folosesc si notatii de forma:

```
R = ABCDE
```

Cu semnificatia: schema relatiei R contine 5 atribute notate cu A, B, C, D si respectiv E.

3.1.5. Cheia unei relatii

O relatie fiind o multime (de tupluri) nu poate contine elemente duplicat – spre deosebire de exemplu de un tabel Excel unde putem avea dubluri.
Rezulta ca tuplurile pot fi deosebite intre ele prin valorile aflate pe una sau mai multe coloane din relatie.

Definitie: ***Cheia unei relatii*** = multime minimala de atribute ale caror valori identifica in mod unic un tuplu al relatiei respective

Cheia unei relatii este o caracteristica a schemei acesteia si nu este determinata prin inspectarea valorilor aflate la un moment dat in relatie.
In tabela **Produse** cele trei linii existente pot fi identificate unic de valorile de pe 3 atribute (IdP, NumeP si Qty) dar numai IdP este cheie:

- IdP identifica (prin definitie) in mod unic un produs; rezulta ca multimea de atribute { IdP } este cheie (fiind si minimala prin natura sa}.
- Multimea de atribute {IdP, IdF} identifica de asemenea unic fiecare tuplu al relatiei dar nu este cheie nefiind minimala: prin inlaturarea lui IdF multimea ramane in continuare cheie.
- Multimea de atribute {NumeP} nu este cheie: este posibil ca in tabela **Produse** sa avem mai multe linii cu NumeP = 'Imprimanta laser', 'Copiator' sau 'Calculator PC'. Asa cum am mentionat cheia se determina din semnificatia atributelor relatiei si nu din valorile la un moment dat.
- Din acelasi motiv nici una dintre celelalte multimi de atribute ale relatiei **Produse** nu este cheie: fie nu este minimala (in cazul in care il include pe IdP) fie nu identifica unic tuplurile relatiei (pot exista valori duble)

In termeni de tabele rezulta ca nu pot exista doua linii avand aceeasi combinatie de valori pe coloanele care formeaza cheia tabelei respective (proprietate denumita in literatura de specialitate si unicitatea cheii)

O relatie poate avea mai multe chei. Sa ne imaginam o relatie **Studenti** continand date despre studentii romani ai unei facultati:

Studenti (IdStud, NrMatricol, Nume, CNP, SerieCI, NumarCI)

In acest caz avem mai multe chei:

- { IdStud } – pentru ca IdStud este un numar asignat de sistem fiecarei inregistrari, fara repetitii
- { NrMatricol } – pentru ca nu pot exista doi studenti ai unei facultati cu acelasi numar matricol
- { CNP } – pentru ca nu pot exista doi cetateni romani (deci nici doi studenti romani) cu acelasi cod numeric personal
- { SerieCI, NumarCI } – pentru ca nu pot exista doi cetateni romani (deci nici doi studenti romani) cu aceeasi combinatie serie/numar carte de identitate.

Observatie: Orice relatie are cel putin o cheie: deoarece intr-o relatie nu pot exista doua elemente identice, rezulta ca multimea tuturor atributelor relatiei este cheie sau contine cel putin o cheie.

In literatura de specialitate si in sistemele de gestiune a bazelor de date exista trei alte concepte legate de cheie si care vor fi prezentate in paragrafele urmatoare ale acestui capitol:

- Cheie primara (eng. Primary key),
- Cheie straina (eng. Foreign key),
- Supercheie (eng. Superkey).

3.1.6. Valori nule

Uneori, unele elemente ale unei relatii (celule ale tabelei) nu au nici o valoare concreta. Se spune ca in acel loc exista o ***valoare nula***.

Definitie: ***Valoare nula*** (eng. Null value) = o valoare diferita de oricare alta si care modeleaza o informatie necunoscuta sau o informatie inaplicabila.

Exemplul urmator prezinta o relatie Studenti in care exista astfel de valori nule si care au fost simbolizate (pentru a iesi in evidenta) prin <NULL>:

Studenti

IdS	NumeStud	Codfacultate	IdTutor	Medie
1001	**Ionescu Ion**	**03**	**<NULL>**	**9,10**
1002	**Popescu Vasile**	**03**	**1001**	**<NULL>**
1003	**Georgescu Ion**	**<NULL>**	**1001**	**8,40**

- ***Modelarea unei informatii necunoscute***: Codul facultatii studentului Georgescu si media lui Popescu sunt nule pentru ca in momentul incarcarii cu date informatia respectiva, desi existenta in lumea reala, nu era cunoscuta celui care a incarcat datele. La un moment ulterior aceste valori nule vor fi inlocuite cu valori nenule care specifica informatia respectiva.
- ***Modelarea unei informatii inaplicabile***: Sa presupunem ca unii dintre studenti sunt consiliati in activitatea lor de un student de an mai mare, numit si tutor. Codul tutorului unui student este inscris pe coloana IdTutor (de exemplu studentii Popescu si Georgescu il au ca tutor pe studentul Ionescu avand codul 1001). In cazul studentului Ionescu insa valoarea lui IdTutor este nula pentru ca acest student (de an mai mare) nu are la randul sau un tutor, valoarea nula fiind cea corecta in contextul respectiv.

3.1.7. Corectitudinea datelor

Schema unei relatii contine descrierea structurii acesteia dar nu da informatii privind corectitudinea datelor continute in aceasta. Exemplul urmator prezinta o incarcare cu date incorecte pentru tabela Produse, erorile fiind urmatoarele:

- Exista doua produse diferite avand acelasi IdP (101)
- Ultimul produs din tabela nu are asignata o valoare pe coloana IdP
- Aceeasi firma are doua coduri numerice diferite (20 si 22)
- Exista doua firme diferite cu acelasi cod (20)

- Aceeasi firma are doua adrese diferite

Produse

IdP	NumeP	Qty	IdF	NumeF	AdresaF
101	Imprimanta laser	30	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
101	Calculator PC	20	20	IBM	Bd. D.Cantemir nr.1, Bucuresti
	Copiator	10	22	Xerox	Str. Batistei, Bucureşti

Specificarea conditiilor de corectitudine pe care trebuie sa le verifice datele se face astfel:

- In cadrul teoriei bazelor de date relationale, o relatie contine date corecte daca acestea verifica setul de ***dependente functionale*** (sau de alt tip) atasat relatiei respective (dependentele functionale sunt prezentate in capitolul 4)
- In cazul sistemelor de gestiune a bazelor de date existente pe piata, acestea pun la dispozitie mecanisme de verificare numite ***constrangeri de integritate***. Constrangerile de integritate se definesc fie la crearea tabelei fie ulterior si sunt de obicei de cinci tipuri, descrise in continuare. Efectul definirii unor astfel de constrangeri de integritate este acela ca SGBD-ul va rejecta orice operatie care violeaza vreuna dintre constrangerile definite pe tabela respectiva.

a. Constrangeri NOT NULL (valori nenule)

Este o constrangere la nivelul unei coloane dintr-o tabela si specifica faptul ca pe coloana respectiva nu pot sa apara valori nule. In cazul tabelei Produse o astfel de constrangere se poate asocia de exemplu pentru toate coloanele sau doar o parte din acestea. Orice incercare de a adauga o linie care contine valori nule pe acea coloana sau de a modifica o valoare nenula intr-una nula va fi respinsa de sistem.

b. Constrangeri PRIMARY KEY (cheie primara)

Asa cum s-a vazut anterior, o relatie poate avea mai multe chei. In momentul creerii tabelei corespunzatoare relatiei intr-un sistem de gestiune a bazelor de date, una dintre ele poate fi aleasa ca si cheie primara (principala) a tabelei respective. O tabela nu poate avea decat o singura cheie primara, formata din una sau mai multe atribute (coloane) ale acesteia. SGBD-ul creaza automat structuri de cautare rapida (index) pentru cheia primara a tabelei.
Alegerea cheii care devine cheie primara va fi facuta in concordanta cu tipul de aplicatie in care este folosita acea tabela. Pentru exemplul tabelei de la paragraful 3.1.5:

Studenti (IdStud, NrMatricol, Nume, CNP, SerieCI, NumarCI)

avand cheile { IdStud }, { NrMatricol }, { CNP } si { SerieCI, NumarCI } alegerea cheii primare se poate face astfel:

- In cazul in care tabela este folosita intr-o aplicatie de gestiune a datelor privind scolaritatea, se poate alege cheia primara NrMatricol, avand in vedere ca o serie de date privind rezultatele unui student sunt legate de matricola sa (informatie de legatura cu alte tabele)

- In cazul in care tabela este folosita intr-o aplicatie a politiei universitare, alegerea se va face probabil intre CNP si (SerieCI, NumarCI), legatura cu bazele de date de la nivelurile superioare facandu-se dupa aceste informatii.
- In ambele cazuri se poate alege cheia primara IdStud, continand numere unice generate automat de sistem.

O caracteristica a cheii primare a unei tabele este (in majoritatea SGBD-urilor existente) cerinta ca pe coloanele componente nu pot sa apara valori nule.

c. Constrangeri UNIQUE (cheie)

Prin acest tip de constrangere se modeleaza celelalte chei ale tabelei. Pe coloanele unei chei definita cu UNIQUE pot insa sa apara valori nule, unicitatea fiind cerificata doar pentru valorile nenule de pe coloanele cheii respective.
In exemplul anterior, daca s-a ales cheia primara IdStud, pentru celelalte trei chei se pot defini trei constrangeri de acest tip.

d. Constrangeri FOREIGN KEY (cheie straina)

Sunt cazuri in care o multime de coloane ale unei tabele contin valori care exista pe cheia primara a unei alte tabele. Sa consideram o baza de date in care exista urmatoarele doua tabele (cheile lor primare sunt cele subliniate):

Studenti(IdS, NumeStud, CodFacultate, IdTutor, Medie)
Facultati(CodFacult, NumeFacultate, Adresa)

Coloana Codfacultate din tabela Studenti nu este cheie in aceasta tabela (pot exista mai multi studenti cu aceeasi valoare pe aceasta coloana, fiind studenti ai aceleiasi facultati) dar in mod normal contine valori care pot fi doar dintre cele existente pe cheia primara CodFacult din tabela Facultati.
O constrangere activa de acest tip (numita si ***constrangere referentiala***) va avea ca efect respingerea inserarilor/modificarilor in tabela Studenti care ar face ca pe coloana Codfacultate sa apara o valoare care nu este deja in tabela Facultati.
Rezulta implicit ca in momentul incarcarii cu date este necesar sa fie completata intai tabela Facultati si apoi tabela Studenti, altfel operatia de incarcare cu date va esua din cauza violarii acestei constrangeri.

In cazul multor SGBD-uri se poate specifica in constrangere si stergerea automata a inregistrarilor 'fiu' in cazul stergerii inregistrarii 'tata': la stergerea liniei corespunzatoare unei facultati (din tabela Facultati) se vor sterge automat si liniile din tabela Studenti continand studentii acelei facultati.

Constrangerile referentiale provin de obicei din transformarea asocierilor unare si binare unu-unu si multi-unu (descrisa in capitolul precedent).

e. Constrangeri CHECK (conditie)

Acest tip de constrangere specifica faptul ca valorile unei linii din tabela trebuie sa verifice o conditie (expresie logica).
Exemplu: In tabela:

Studenti(IdS, NumeStud, CodFacultate, IdTutor, Medie)

pe coloana Medie putem defini o constrangere de acest tip specificand ca valoarea (daca exista o valoare nenula) trebuie sa fie din intervalul [0, 10].

3.2. Transformarea diagramelor EA in modelul relational

In procesul de transformare vom pleca de la o diagrama EA si vom obtine trei tipuri de scheme de relatie:

a. Relatii provenite din entitati. Ele contin aceleasi informatii ca si entitatile din care au rezultat.

b. Relatii provenite din entitati si care contin chei straine.

Ele contin pe linga informatiile provenite din entitatile din care au rezultat si atribute care in alte entitati sint identificatori. Este cazul acelor entitati care au asocieri multi-unu si partial din cele care au asocieri unu-unu cu alte entitati.

c. Relatii provenite din asocieri. Este cazul celor care apar din transformarea asocierilor binare multi-multi si a asocierilor de grad mai mare ca doi. Ele contin ca atribute reuniunea identificatorilor entitatilor asociate plus atributele proprii ale asocierilor.

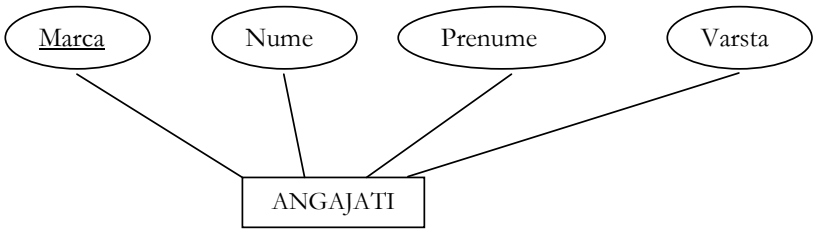
Procesul de transformare are un algoritm foarte precis si este din aceasta cauza pasul care se preteaza cel mai bine pentru crearea de instrumente software care sa-l asiste.

Transformarea entitatilor

Fiecare entitate a diagramei se transforma intr-o schema de relatie avind:

Numele relatiei	=	Numele entitatii
Atributele relatiei	=	Atributele entitatii
Cheia relatiei	=	Identificatorul entitatii

Exemplu: Fie entitatea Angajati de mai jos:



Schema rezultata este Angajati(Marca, Nume, Prenume, Varsta)

Transformarea asocierilor unare si binare unu-unu si multi-unu

Fiecare asociere din aceasta categorie va avea ca rezultat imbogatirea multimii de atribute descriptive ale uneia dintre cele doua scheme rezultate la pasul 2.1 din entitatile asociate, cu cheia celeilalte scheme. Aceste atribute care se adauga sint denumite in prezentarea de fata cheie straina deoarece ele sint cheie dar in alta schema de relatie.

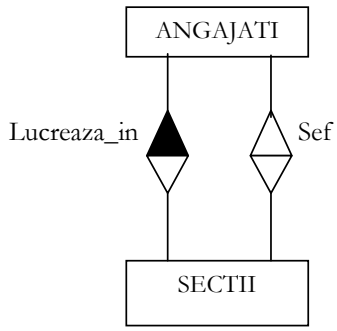
a. In cazul asocierilor multi-unu, se adauga identificatorul entitatii unu in schema rezultata din entitatea multi

b. In cazul asocierilor unu-unu, se adauga identificatorul unei entitati in schema rezultata din transformarea celeilalte. Alegerea schemei in care se face adaugarea se poate face dupa doua criterii:

- fie in acea schema care defineste relatia cu cele mai putine tupluri din cele doua,
- fie pastrindu-se, daca exista, filiatia naturala intre cele doua entitati: identificatorul tatalui se adauga la fiu.

In cazul acestui tip de asocieri, la schema de relatie care primeste cheia straina se ataseaza una sau doua dependente functionale primare (vezi tabelul 1.1.)

Exemplu:



Intre entitatile SALARIATI si SECTII exista doua asocieri: una unu-unu, SEF, care modeleaza faptul ca o sectie este condusa de un salariat (seful de sectie) si una multi-unu, LUCREAZA_IN descrisa in paragrafele anterioare.
Rezultatul transformarii este cel de mai jos. Atribulele aflate dupa punctele se suspensie sint cele adaugate (chei straine).

SALARIATI (Marca, Nume, Prenume, Virsta, ..., Cod_Sectie)
SECTII(Cod_Sectie, Profil, ..., Marca_Sef)

Din asociere LUCREAZA_IN

Din asociere SEF

Pe cimpul Cod_Sectie din relatia SALARIATI se va inregistra pentru fiecare salariat codul sectiei in care acesta lucreaza iar pe cimpul Marca din relatia SECTII se va inregistra pentru fiecare sectie marca sefului de sectie. Pentru asociere SEF s-a aplicat primul criteriu (relatia SECTII va avea mult mai putine inregistrari decit SALARIATI), dar si al doilea criteriu este indeplinit.

Transformarea asocierilor unare si binare multi-multi si a celor de grad mai mare ca doi

Fiecare asociere binara multi-multi si fiecare asociere cu grad mai mare ca doi se transforma intr-o schema de relatie astfel:

Nume relatie	=	Nume asociere
Atribute relatie	=	Reuniunea identificatorilor entitatilor asociate la care se adauga atributele proprii ale asocierii
Cheia relatiei	=	Conform tabelului 1.2.

Grad	**Conectivitate**	**Cheia relatiei provenite din asociere**
Unare	multi (E) - multi (E)	Cheie(E) + Cheie(E)
Binare	multi (E1) - multi (E2)	Cheie(E1) + Cheie(E2)
Ternare	unu (E1) - unu (E2) - unu (E3)	Cheie(E1)+Cheie(E2) sau Cheie(E1)+Cheie(E3) sau Cheie(E2)+Cheie(E3) sau
	unu (E1) - unu (E2) - multi (E3)	Cheie(E1)+Cheie(E3) sau Cheie(E2)+Cheie(E3) sau
	unu (E1) - multi (E2) - multi (E3)	Cheie(E2)+Cheie(E3)
	multi (E1) - multi (E2) - multi (E3)	Cheie(E2)+Cheie(E3)+Cheie(E1)

Cheile schemelor de relatie rezultate din asocieri

Legenda:

X + Y: Multimea de atribute X impreuna cu multimea de atribute Y