BAZE DE DATE

Prin **bază de date** înţelegem o colecţie de date elementare şi interdependente, structurate şi organizate astfel încât să fie accesibile unei comunităţi de utilizatori. Bazele de date sunt concepute pentru a prelucra volume mari de date.

Organizarea datelor în unul sau mai multe fișiere stocate în memoria externă este în concordanță cu această definiție.

Din pacate, utilizarea directă a fișierelor are mai multe dezavantaje precum:

- acces dificil la date - în practică, pentru accesarea fiecarui fișier trebuie scris un program, în plus orice modificare a structurii fișierului presupune modificarea tuturor programelor care îl folosesc.
- lipsa de securitate - întrucât orice programator poate accesa direct fișierele, este imposibil să se garanteze securitatea și integritatea datelor.
- într-un mediu în care mai mulți utilizatori accesează aceleași fișiere apar probleme de concurență greu de rezolvat.

Prin urmare, devine necesară utilizarea unui software specializat, responsabil cu gestionarea fișierelor bazei de date, ușurarea rezolvării aspectelor de securitate și de furnizarea interfețelor necesare pentru accesarea datelor.

Acest software complex poartă denumirea de Sistem de Gestiune a Bazelor de Date (SGBD) și reprezintă sistemul de programe care permite construirea bazelor de date, introducerea informaţiilor în bazele de date şi dezvoltarea de aplicaţii privind bazele de date. Conceptul de Sistem de Gestiune a Bazelor de Date (SGBD) se regăseşte cu aceeaşi denumire şi semnificaţie în terminologia franceză (SGBD - Systeme de Gestion de Bases de Donnees) şi engleză (DBMS - DataBase Management System).

Conceptul de *bază de date* a apărut în 1964 în cadrul primului raport CODASYL[1] prezentat la lucrările unei conferinţe pe probleme de limbaje de gestiune a datelor *"Development and Management of Computer – centered date-base"*. La această conferinţă a fost lansată ideea organizării datelor prin intermediul unui fişier de descriere globală, numit *dicţionar de date* care are menirea de a asigura independenţa programelor faţă de date şi a datelor faţă de programe[2].

Baza de date conţine nu numai datele operaţionale ale organizaţiei, ci şi o descriere a acestora. De aceea ea este definită şi ca o colecţie autodescrisă de

[1] COnference on DAta SYstems Languages – Conferinţa despre Limbajele Sistemelor de Date

[2]Lungu, I., ș.a., Baze de date, Organizare, proiectare şi implementare, Editura All, Bucureşti, 1995, p.13

înregistrări integrate. Această descriere a datelor este cunoscută sub denumirea de *catalog de sistem* sau *dicţionar de date* sau *meta-date* (date despre date).

Natura autodescriptivă a bazelor de date este cea care determină independenţa program-date.

Accesul utilizatorilor la informaţiile despre structura unei baze de date se realizează prin intermediul dicţionarului de date.

Sistemul de gestiune a bazelor de date constituie o interfaţă între utilizatori şi baza de date şi asigură următoarele funcţiuni:

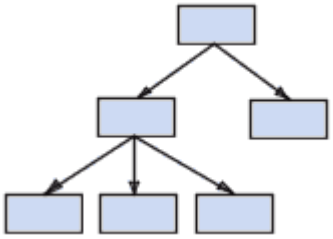
1. Descrierea structurii bazei de date: a fiecărui tabel, a relaţiilor dintre tabele, a restricţiilor în reprezentarea informaţiilor etc. cu ajutorul unui limbaj de descriere a datelor LDD.
   LDD este utilizat atât pentru proiectarea bazelor de date cât şi pentru redefinirea lor.
2. Operaţii asupra datelor aflate în baza de date:
   - introducerea inițială a datelor
   - adăugarea de noi date
   - modificarea unor date existente pentru a le pune în acord cu realitatea
   - ştergerea unor date devenite inutile

   Această functiune este realizată cu ajutorul unui limbaj de manipulare a datelor LMD.
3. Interogarea bazei de date, adică extragerea unor informaţii stocate în aceasta, realizarea de statisci asupra datelor etc. cu ajutorul unui limbaj de cereri LC
4. Administrarea și protecția bazei de date - ansamblul de măsuri necesare pentru asigurarea integrităţii şi securităţii datelor.

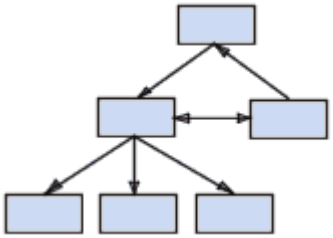
Diferite modele de baze de date

Există mai multe modele de baze de date, diferențiate în funcție de reprezentarea datelor pe care le conține:

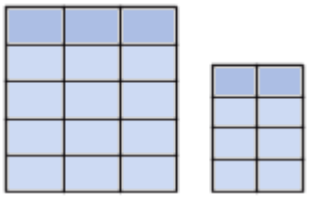
Modelul ierarhic: Datele sunt clasificate ierarhic într-o structură de tip arbore. Acest model folosește pointeri între înregistrări. Acesta a fost primul model de SGBD



Modelul de rețea: ca si modelul ierarhic acest model foloseste pointeri la înregistrări. Structura sa nu este neapărat arborescentă.



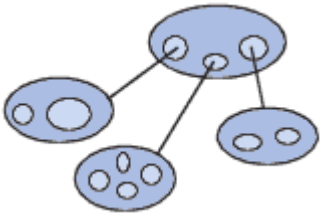
Modelul relațional: datele sunt stocate în tabele bidimensionale (formate din rânduri și coloane). Manipularea acestor date are loc în conformitate cu teoria matematică a relațiilor.



Bazele de date relaționale reprezintă cel mai frecvent tip de baze de date utilizat.

Modelul deductiv: datele sunt reprezentate în formă tabelară, dar manipularea lor se face prin calculul predicatelor.

Modelul orientat pe obiecte: datele sunt stocate ca obiecte, adică structuri numite clase de date cu membri.



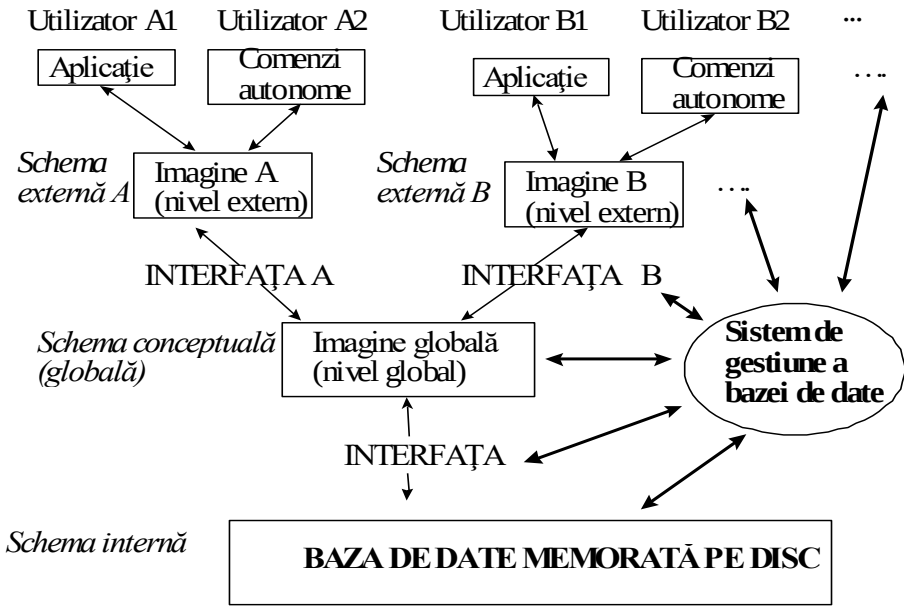
Baze de date XML: unitatea fundamentală de stocare este documentul XML și nu înregistrarea. XML (eXtensible Markup Language) este un limbaj generic bazat pe marcatori (taguri). Limbajul este considerat a fi extensibil deoarece permite utilizatorilor să îşi definească propriile elemente. XML a fost creat în principal pentru a facilita schimbul de date între diferite aplicaţii prin intermediul Internetului,.

Niveluri şi modele de reprezentare a datelor în bazele de date

În demersul realizării unei baze de date există trei niveluri de percepţie a acesteia:

- extern, corespunzător utilizatorilor, care îşi exprimă cerinţele informaţionale prin aşa - numitele scheme externe (subscheme);
- conceptual, aferent proiectanţilor bazei de date, care se concretizează în schema conceptuală;
- intern, corespunzător programatorului, care realizează reprezentarea datelor pe suportul fizic.

Văzută prin prisma celor trei niveluri, baza de date poate fi reprezentată ca în figura următoare.[3]



Nivele de abstractizare a datelor în bazele de date

Determinarea structurii unei baze de date se poate aborda ascendent, realizându-se descrierea schemelor externe, urmată de elaborarea schemei conceptuale, sau descendent definind mai întâi, schema conceptuală şi deducând ulterior schemele externe posibil de obţinut.

[3] Fotache, M., Baze de date relaţionale. Organizare, interogare şi normalizare, Editua Junimea, Iaşi, 1997, p.32

Cel mai adesea, schema conceptuală se obţine pe baza schemelor externe, prin eliminarea redundanţelor şi "rafinarea" acestora. Trecerea de la schema conceptuală la cea internă se face prin intermediul limbajelor de descriere a datelor, care sunt incluse în SGBD-uri. Un model de date presupune un limbaj de descriere a realităţii, în timp ce o schemă este o descriere a unei realităţi după un model dat.

Nivelul extern (schema externă)

La nivelul extern, fiecare grup de lucru care manipulează datele posedă o anumită descriere a acestora numită schemă externă. Această descriere corespunde felului în care grupul vede baza de date în programele lui de aplicaţii. Prin urmare nivelul extern reflectă mulţimea datelor care prezintă interes pentru un utilizator sau pentru un grup de utilizatori.

Există aşadar, mai multe scheme externe, fiecare schemă externă presupune utilizarea unei părţi din baza de date, folosind informaţiile într-un mod bine determinat. Această parte de informaţie se numeşte *vedere*. Fiecărei vederi îi corespunde un grup de utilizatori. Pentru un grup particular se definesc tipurile de cereri de informaţii şi modul de determinare a datelor din BD care formează răspunsuri la aceste cereri. Un pas important în proiectarea unei BD este determinarea vederilor şi a claselor de utilizatori asociate lor. Vederile permit ca aceleaşi date să fie privite din perspective diferite de diferiţi utilizatori.

Nivelul conceptual (modelul conceptual)

Nivelul conceptual este un nivelul central care structurează datele astfel încât acestea să poată fi preluate şi prelucrate cu ajutorul unui SGBD.

Nivelul conceptual reprezintă viziunea proiectanţilor BD asupra datelor.

Modelul conceptual (schema conceptuală) al datelor este esențial pentru buna dezvoltare a bazei de date. În măsura în care baza de date este fundamentul întregului sistem, o proiectare greșită a schemei conceptuale va induce în baza de date anomalii dificil de remediat ulterior.

În practica proiectarii bazelor de date (în principal a bazelor de date relaționale), la nivel conceptual, unul dintre modelele frecvent utilizat este modelul Entitate-Asociere (E/A). Modelul Entitate-Asociere (E/A) denumit și model Entitate-Relatie(E/R) a fost propus de Peter Pin-Shan Chen în 1976.

Modelul E/A este caracterzat prin a fi simplu dar suficient de puternic pentru a reprezenta structuri relaționale. Mai presus de toate, acest model se bazează pe o reprezentare grafică care facilitează foarte mult înțelegerea.

Modelul E/A împarte elementele unui sistem real în două categorii şi anume în *entităţi*(agregări de date elementare) şi în *asocieri* între entităţi.

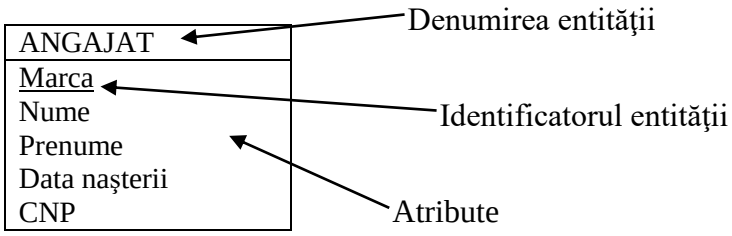
O *entitate* (sau tip entitate) este o grupare de obiecte cu caracteristici sau proprietăți similare. Entitățile sunt reprezentate grafic prin dreptunghiuri.

O entitate este identificată printr-un nume(un substantiv) și descrie proprietățile obiectelor pe care le reprezintă, numite atribute ale entității.

Exemple de entități: student, profesor, curs, angajat, produs, utilaj, factura.

Un *atribut* se defineşte ca fiind o proprietate a unei entităţi sau a unei asocieri. Fiecare atribut care a fost selectat la definirea unei entități este o caracteristică semnificativă pentru domeniul studiat.

Exemplu:



Un atribut poate fi simplu, când poate lua o singură valoare, sau repetitiv cînd poate lua mai multe valori(exemple: Limbi straine cunoscute, numar telefon).

Există atribute care necesită valori „nule” fapt care trebuie să fie luat în considerare la proiectarea bazei de date.

Entitatea este percepută ca un tip de obiecte. Fiecare obiect individual constituie o *realizare*(sau *instanţă*) a entităţii.

În cadrul unei entităţi nu pot exista două realizări(instanţe) identice. Fiecare entitate trebuie sa conţină un atribut sau un grup de atribute care identifică în mod unic instanţele entităţii. Acesta este denumit *identificatorul*(cheia) entităţii şi serveşte drept *cheie primară* în viitoarea bază de date.

Există două tipuri de identificatori *naturali* şi *artificiali*.

Un identificator natural este alcătuit dintr-un atribut sau un grup de atribute cu semnificaţie reală pentru entitatea în cauză. De exemplu, combinaţia Nume, Prenume, Data naşterii este un identificator natural pentru entitatea Angajat.

Un identificator artificial este alcătuit dintr-un atribut sau un grup de atribute fără semnificaţie reală pentru entitatea în cauză, fiind folosit doar pentru a face distincţie între instanţele entităţii. Exemple CNP, Marca, Număr inventar.

În reprezentarile grafice identificatorii entităţilor se noteaza subliniat.

Regula minimalității identificatorilor: în cazul identificatorilor compuşi, trebuie să nu existe un subgrup al său care să poată îndeplini rolul de identificator, adică numărul de elemente componente ale identificatorului să fie minim. Aceasta se poate asigura prin verificarea *dependenţelor funcţionale* dintre componentele identificatorului respectiv.

Asocierile modelează interdependenţele dintre clasele de obiecte reprezentate de entităţi. Sunt luate în considerare doar interdependenţele necesare aplicaţiei de proiectat, în lumea reală putând exista între entităţile analizate şi alte asocieri care nu sunt semnificative pentru aplicaţie. O asociere poate avea atribute proprii.

Asocierile sunt reprezentate uzual prin verbe și grafic prin linii între entități.

Asocierile pot fi binare (între 2 entităţi) sau n-are (între n entităţi, n>2).

O caracteristica foate importanta a asocierilor este *cardinalitatea*. Acesta exprima modul de participare al instanţelor fiecarei entitati la asociere, mai concret cardinalitatea ne arată la câte asocieri poate participa o instanță a unei entități.

Fiind date două entităţi, E1 şi E2, se definesc următoarele asocierile binare:

- *Asocierea "unu-la-unu"* (*one-to-one*) este asocierea în care unei instanţe a entităţii E1 îi coreaspunde cel mult o instanţă a entităţii E2, şi reciproc; se notează cu **1:1**.
- *Asocierea "unu-la-multi"* (*one-to-many*) este asocierea în care unei instanţe a entităţii E1 îi coreaspund zero, una sau mai multe instanţe ale entităţii E2, dar unei entităţi din E2 îi corespunde cel mult o instanţă a entităţii E1; se notează cu **1:N**.
- *Asocierea "multi-la-multi"* (*many-to-many*) este asocierea în care unei instanţe a entităţii E1 îi coreaspund zero, una sau mai multe instanţe ale entităţii E2, şi, de asemenea, instanţe a entităţii E2 îi coreaspund zero, una sau mai multe instanţe ale entităţii E1; se notează cu **N:N**.

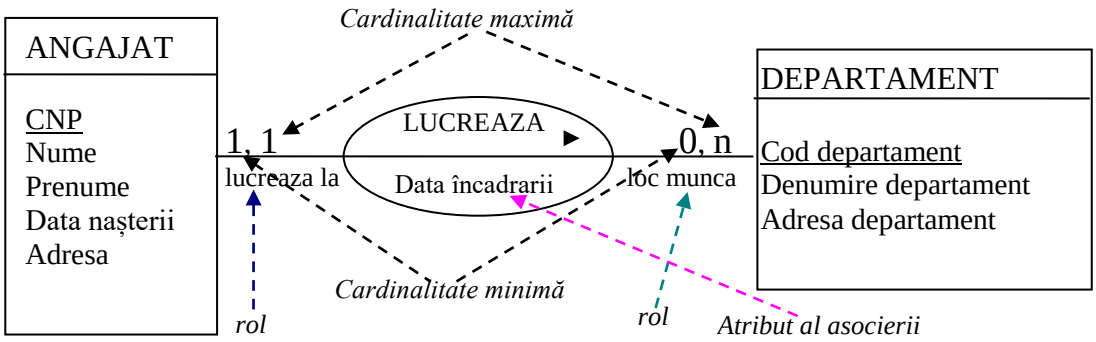
Aceste valori reprezintă cardinalitatea maximă a asocierii. O asociere este caracterizată şi de o cardinalitate minimă care indică obligativitatea participării instanţelor la asociere.

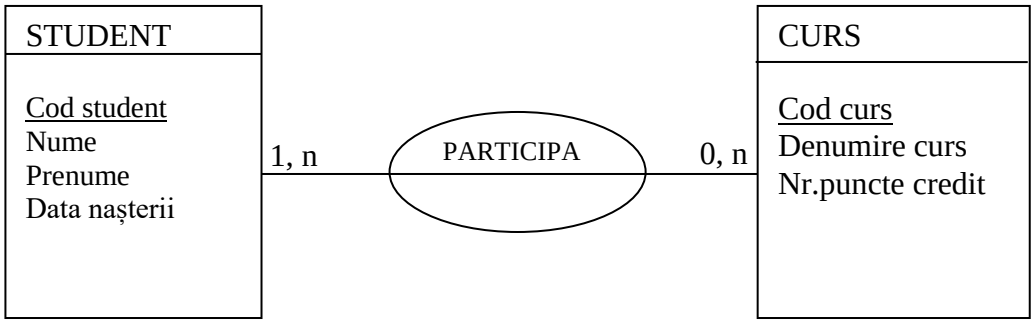
Cardinalitatea minimă zero: pot exista instanţe ale entităţii care nu participă la nici o realizare a asocierii.

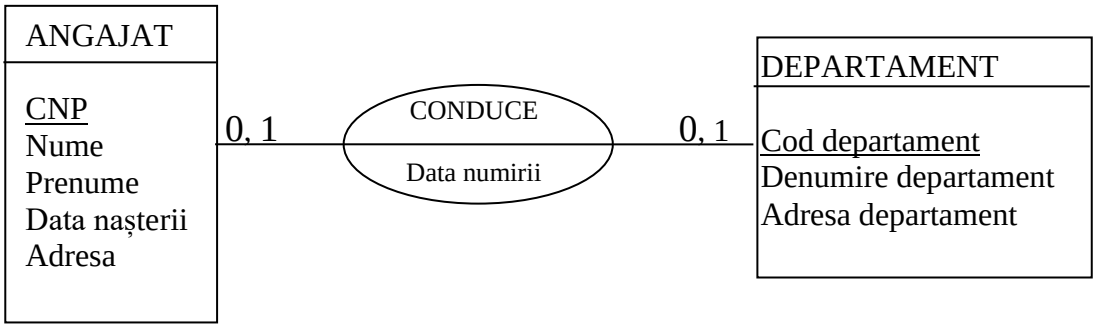
Cardinalitatea minimă unu: toate instanţele entităţii trebuie să participe la asociere.

Rolul unei entităţi este un nume care desemnează modul de participare al entităţii la o asociere. Identificarea asocierilor se realizează prin rolurile entităţilor participante deci, concret, cu ajutorul identificatorilor entităţilor participante.

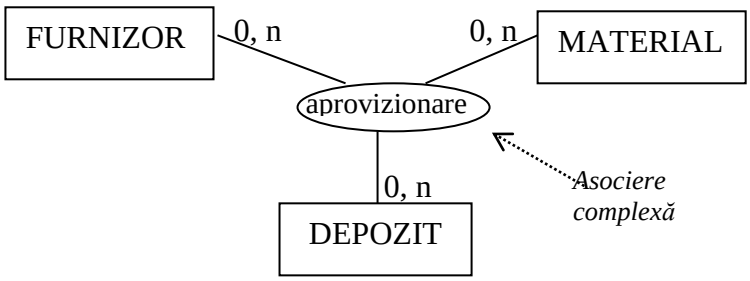
Exemple:







In determinarea asocierilor trebuie să se ţină seama şi de posibilitatea existenţei asocierilor complexe (între mai mult de două entităţi). De exemplu, fie entităţile FURNIZOR, MATERIAL, DEPOZIT şi asocierile binare corespunzătoare. Aceste asocieri permit să se determine furnizorii materialelor, materialele intrate în depozite respectiv furnizorii ce aprovizionează depozitele. Nu se poate determina ce furnizor aprovizionează cu un anumit material un anume depozit. Introducerea asocierii "aprovizionare" între entitățile FURNIZOR, MATERIAL și DEPOZIT permite rezolvarea cererii "ce furnizor aprovizionează cu un anumit material un anume depozit?".



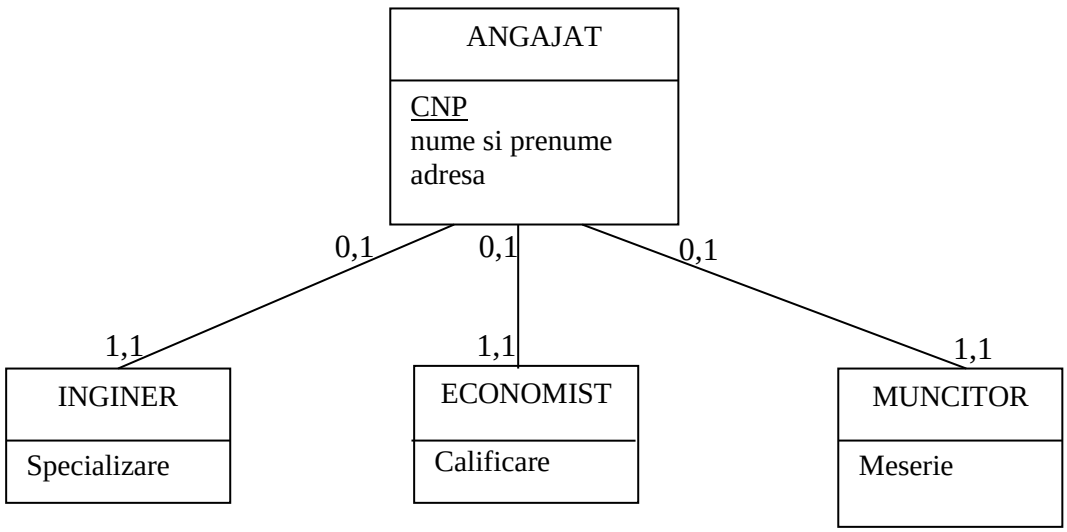
Cazuri speciale de entităţi şi asocieri

Subentitate-Superentitate. În modelul E-A se pot defini subtipuri de entităţi denumite şi subentităţi, care reprezintă specializări ale unor tipuri de entităţi denumite superentităţi, şi se pot defini, de asemenea, ierarhii pe mai multe nivele de tipuri şi subtipuri de entităţi.

Se pot folosi două modalităţi de definire a ierarhiilor de tipuri: *specializarea* şi *generalizarea*.

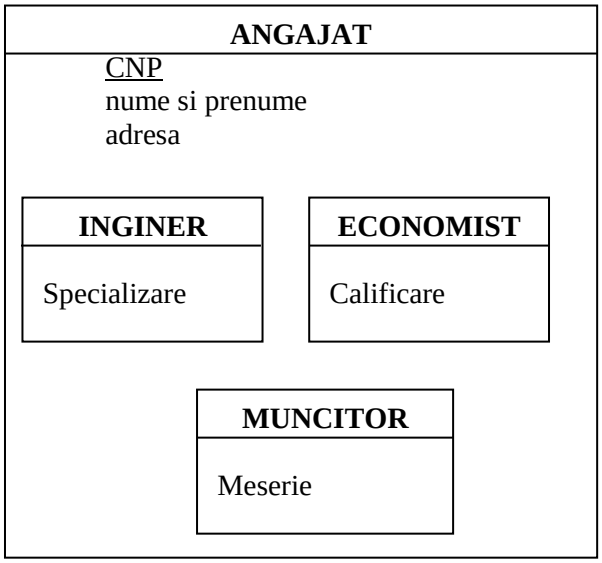
Specializarea este un proces de abstractizare a datelor prin care, pornind de la o entitate dată, se definesc una sau mai multe subentităţi, diferenţiate între ele în funcţie de rolul pe care îl au în modelul de date.

De exemplu, de la tipul de entitate ANGAJAT se definesc subentităţile INGINER, ECONOMIST, MUNCITOR. Acestea "moştenesc" toate atributele tipului iniţial dar au în plus atribute suplimentare, specifice rolului lor.



Intre subentitate şi superentitate există întotdeauna o asociere de tip1,1:0,, semnificând faptul că o instanţă a unei subentităţi este asociată cu o singură instanţă a entităţii de bază şi reciproc.

O subentitate se reprezintă într-un dreptunghi inclus în dreptunghiul care reprezintă superentitatea corespunzătoare.



Generalizarea este procesul de abstractizare invers specializării, prin care se crează un supertip de entitate pornind de la mai multe tipuri de entităţi. Pentru aceasta se identifică atributele comune ale mai multor tipuri de entităţi şi aceste atribute vor caracteriza superentitatea, iar atributele care diferă de acestea rămân atribute specifice ale fiecărui tip.

De exemplu, dacă au fost definite tipurile de entităţi: AUTOMOBIL (Marca, VitezaMaxima, Pret, NumarPasageri) şi CAMION (Marca, VitezaMaxima, Pret, Tonaj), se poate defini un supertip al acestor tipuri: VEHICUL (Marca, VitezaMaxima, Pret), care cuprinde toate atributele comune, iar tipurile AUTOMOBIL şi CAMION devin subtipuri ale tipului VEHICUL, fiecare conţinând atributele specifice (NumărPasageri pentru tipul AUTOMOBIL şi Tonaj pentru tipul CAMION).

Rezultatul obţinut prin generalizare este, ca şi în cazul specializării, o ierarhie de tipuri şi subtipuri de entităţi, iar asocierea dintre mulţimile corespunzătoare este tot o asociere 1:1. Ceea ce diferă în generalizare faţă de specializare, este doar modul în care se definesc nivelele ierarhiei.

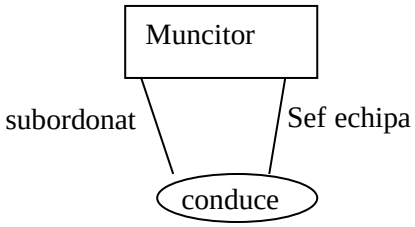
Specializarea poate fi totală(instanţele superentităţii aparţin unei subentităţi) sau parţiala(pot exista instanţe ale superentităţii care să nu aparţină nici unei subentităţi). Generalizarea, fiind prin definiţie gruparea de entităţi deja existente în baza de date nu poate fi decât totală.

Intre subtipuri poate exista o excluziune, ceea ce se traduce prin faptul ca o anumita instanţă nu poate apartine decat unei singure subentităţi. Exista insa si cazuri în care aceeași instanţă aparține mai mulor subtipuri. De exemplu un angajat al unei universitați poate fi și student la frecvență redusă. Pentru astfel de situații trebuie precizate după caz relații de excluziune, incluziune, etc.

Introducerea de subtipuri prin specializare sau prin generalizare prezintă două avantaje principale: favorizează proprietăţile comune la nivelul tipului (superentităţii) şi face mult mai clară reprezentarea unor tipuri de asocieri la care participă numai o parte dintre entităţi.

Asocieri reflexive

Pot exista asocieri şi intre o entitate şi ea insăşi, acestea se numesc asocieri *reflexive* sau *recursive*.



Etapele obținerii modelului Entitate-Asociere

1) Identificarea entităților sistemului
2) Identificarea atributelor și a identificatorilor entităților
3) Identificarea asocierilor între entități
4) Identificarea atributelor proprii asocierilor
5) Stabilirea cardinalităților
6) Trasarea diagramei

Nivelul intern

Nivelul intern este nivelul elementar la care pot fi considerate datele şi se referă la modul în care sunt stocate datele pe suporturi magnetice. La acest nivel structura datelor este foarte detaliată. Nivelul intern cuprinde structurile de date şi organizările fişierelor utilizate pentru stocarea datelor pe dispozitivele de stocare. El tratează probleme cum ar fi: alocarea spaţiului de stocare pentru date şi indexuri, descrierile înregistrărilor pentru stocare, cu dimensiunile de stocare pentru articolele de date, plasarea înregistrărilor, tehnicile de comprimare şi de codificare a datelor. Nivelul intern interacţionează cu metodele de acces al sistemului de operare (tehnici de administrare a fişierelor, pentru stocarea şi regăsirea înregistrărilor de date) pentru a plasa datele pe suporturile de stocare, a regăsi datele, a realiza indexurile.

Includerea în baza de date a descrierii structurii acesteia o deosebeşte calitativ de fişierele de date, deoarece prin aceasta se asigură independenţa datelor din bază faţă de programele de aplicaţii şi invers. Posibilitatea modificării structurii la un nivel, fără a afecta structura celorlalte niveluri este întâlnită sub numele de independenţa datelor, prezentă sub două forme:

- independenţa fizică de date, adică posibilitatea modificării structurii bazei de date la nivel intern (cum ar fi utilizarea unor organizări ale fişierelor sau structuri de stocare diferite, a unor dispozitive diferite de stocare, modificarea de indexuri sau de algoritmi hash), fără a fi necesară schimbarea structurii conceptuale şi rescrierea programelor de prelucrare a datelor. Asemenea modificări sunt necesare pentru ameliorarea performanţelor de lucru (viteză de acces, mărimea fişierelor etc.). Autonomia fizică este cea care asigură şi portabilitatea bazei de date de pe un sistem de calcul pe altul fără modificarea schemei conceptuale şi a programelor;

- independenţa logică de date se referă la faptul că modificarea schemei conceptuale a bazei de date (cum ar fi adăugarea sau eliminarea unor entităţi, atribute sau relaţii) nu necesită şi modificarea schemei externe sau rescrierea programelor de aplicaţii.

Este important să se facă distincţie între descrierea bazei de date şi baza de date însăşi. Descrierea bazei de date constituie schema bazei de date. Ea este specificată în timpul procesului de proiectare a bazei de date şi este schimbată rareori. Setul de date din baza de date se numeşte instanţa bazei de date. Mai multe instanţe ale bazei de date pot corespunde aceleiaşi scheme a bazei de date.

Modelul relaţional al datelor

Modelul relațional al datelor a fost conceput și dezvoltat de Edgar Frank Codd (1923–2003, informatician american de origine engleză) la începutul anilor 1970 și cuprinde trei componente principale: structura datelor, integritatea datelor și prelucrarea datelor.

Structura datelor

In modelul relațional al datelor sunt definite noțiunile de *atribut*, *domeniu*, *relație*, *schema relației*, *schema relațională*.

Un *atribut* este un identificator(un nume) ce descrie o informatie memorată în baza de date.

Domeniul unui atribut este o mulțime de valori posibile. Aceste valori sunt de tip similar.

O *relație* este o submulțime a produsului cartezian a *n* domenii ale atributelor($n>0$).

Elementele unei relații se numesc *tupluri*.

Intr-o relație nu sunt permise tupluri identice.

Relațiile sunt reprezentate sub forma tabelelor bidimensionale în care fiecare rând reprezintă un tuplu și fiecare coloană(atribut al relației) reprezintă valorile actuale ale tuplurilor dintr-un domeniu al produsului cartezian. Tabelele formează structura logică a modelului relațional.

La nivel fizic, sistemul utilizează diverse tehnici de stocare (fișiere secvențiale, indexate, tabele de dispersie, compresia datelor, pointeri etc). Tabelele reprezintă o abstractizare a datelor înregistrate fizic în memorie.

Gradul (*aritatea*) unei relații este dat de numărul de atribute.

Pentru o relație(tabel), pot exista trei tipuri de *chei*(identificatori):

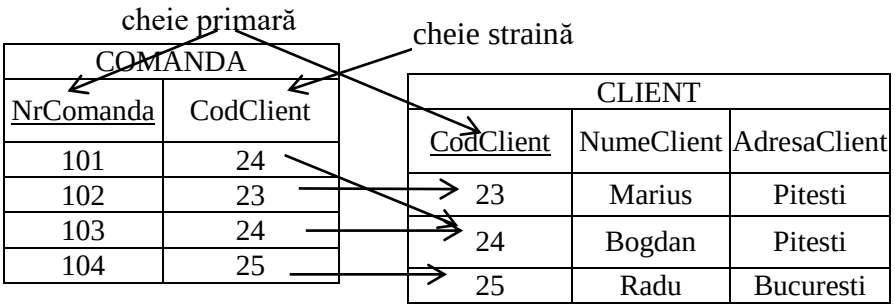
- *cheie candidat* – un ansamblu minimal de atribute(coloane) eventual un singur atribut care permite identificarea fără echivoc a fiecărui tuplu(rând) al relației(tabelului).
- *cheie primară*. Pentru fiecare relație se alege o cheie candidat care va fi desemnată ca fiind cheia primară a relației. Celelalte chei candidat, daca există, poartă denumirea de *chei alternante*. Atributele care compun cheia primară nu pot avea valoarea Null (valoare convențională pentru situația în care valoarea unui atribut este necunoscută sau neaplicabilă).

- *cheie străină* – un ansamblu de atribute(eventual un singur atribut) al unei relații R care în același timp este cheie primară sau alternantă în altă relație S, nu neapărat distinctă de R și care respectă regula integrității referențiale(*RIR*) adică valorile cheii străine, dacă nu sunt *null*, se regăsesc printre valorile cheii primare (candidat)ale relației S.

Schema relației precizează numele relației, lista de atribute cu domeniile lor și cheia primară a relației.

Schema relațională este compusă din mulțimea schemelor relațiilor cu menționarea cheilor străine.

De exemplu, pentru relațiile COMANDA și CLIENT urmatoare:



COMANDA

NrComanda	CodClient
101	24
102	23
103	24
104	25

CLIENT

CodClient	NumeClient	AdresaClient
23	Marius	Pitesti
24	Bogdan	Pitesti
25	Radu	Bucuresti

avem următoarea schemă relațională:

COMANDA(NrComanda, #CodClient)

CLIENT(CodClient, NumeClient, AdresaClient)

Integritatea datelor

Este dată de corectitudinea informațiilor conținute în baza de date și presupune detectarea, corectarea și prevenirea diferitelor erori. Condițiile de integritate nu permit introducerea în baza de date a unor date aberante.

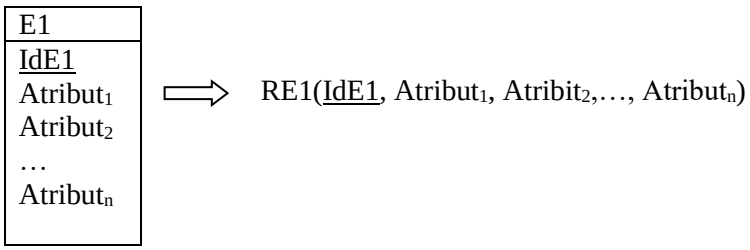
Există constrângeri(reguli) de integritate la nivel de domeniu care privesc anumite valori pentru atribute, constrângeri la nivel de tuplu și constrângeri multituplu- combinații de mai multe tupluri, constrângeri legate de cheile candidat (unicitatea valorilor cheii și pentru cheia primara, excluderea valorilor nule), constrângeri legate de cheia străină (necesitatea existenței unei valori aparținând cheii primare din relația asociată care să coincidă cu valoarea cheii străine nenulă (RIR)).

Prelucrarea datelor

Are la bază algebra relațională bazată pe o colecție de operatori ce au ca operanzi relații. Rezultatul aplicării unui operator la una sau două relații(în funcție de aritatea acelui operator) este tot o relație.

Sunt cinci operații de bază care pot fi aplicate relațiilor: reuniunea, diferența, produsul cartezian, proiecția și selecția.

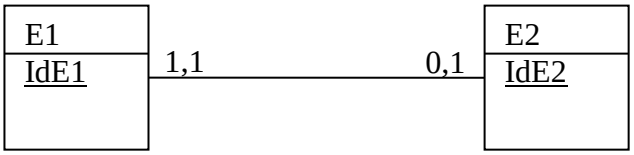
Algoritm de transformare a modelului Entitate-Asociere în model relațional.
1 O entitate devine o relație(tabel)
2 Un atribut al unei entități devine atribut al relației corespunzătoare(coloană în tabel). Identificatorul entității devine cheie primară a relației.



3. O asociere, în funcție de tipul său, va fi reprezentată fie printr-o relație specială (tabel asociativ) fie printr-o cheie străină într-una din cele două relații care face referire la cheia primară a celeilalte relații.

3.1 Asocierile maximale 1:1 se rezolvă prin intermediul cheii străine.
Cheia străină va fi plasată în funcție de cardinalitatea minimă.

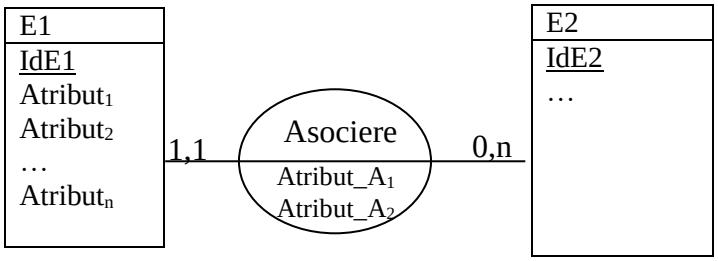
-Pentru cardinalitatea minimă 1:0 cheia străină va fi plasată în relația cu cardinalitatea minimă 1.



RE1(IdE1, #IdE2, …)
RE2(IdE2, …)
-Pentru cardinalitatea minimă 0:0 cheia străină va fi plasată în relația cu mai puține tupluri.

-Pentru cardinalitatea minimă 1:1 cheia străină va fi plasată în oricare relație.

3.2 Asocierea maximală 1:n se rezolvă prin intermediul cheii străine. Cheia străină va fi plasată în relația de partea 1 a asocierii.



E1(IdE1, $Atribut_1$, $Atribut_2$, …, $Atribut_n$, #IdE2, $Atribut\_A_1$, $Atribut\_A_2$)

E2(IdE2, …..)

3.3 Asocierea maximală n:n se transformă într-o relație nouă, un tabel asociativ care conține două chei străine corespunzătoare celor două tabele asociate. Cheia primară a tabelului asociativ este compusă din cele două chei străine plus eventual alte coloane adiționale.

Exemplificare:

Se dorește realizarea unei aplicații cu baze de date pentru evidența vânzărilor produselor unei firme. Produsele se vând către clienți prin intermediul facturilor. Se consideră următoarea colecție (simplificată) a datelor:
NrFactura, DataFactura, CodClient, NumeClient, AdresaClient, CodProdus, DenumireProdus, Cantitate, Pret unitar.
Reguli de gestiune:

- O factură este emisă pentru un client
- Unui client îi pot fi emise mai multe facturi
- O factură face referire la mai multe produse
- Un produs este obiectul mai multor facturi

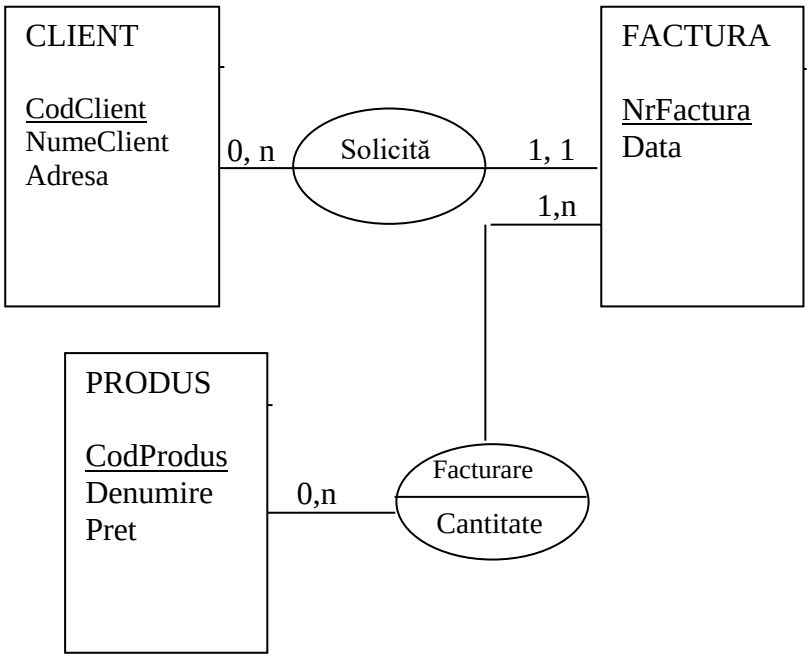
Să se realizeze modelul conceptual și modelul relațional al datelor.

Rezolvare:
Entitățile sistemului sunt:
Factura, Client, Produs.
Entitatea Factura este caracterizată prin atributele NrFactura și DataFactura. NrFactura reprezintă identificatorul entității.
Entitatea Client este caracterizată prin atributele CodClient, NumeClient, AdresaClient. CodClient reprezintă identificatorul entității.
Entitatea Produs este caracterizată prin atributele CodProdus, DenumireProdus, Pret. CodProdus reprezintă identificatorul entității.

Intre entitățile Client și Factură există asocierea <<*solicită*>> de cardinalitate n:*1*.

Intre entitățile Factură și Produs există asocierea <<*facturare*>> de cardinalitate *n:n* și cu atributul specific *cantitate*.

Modelul conceptual al sistemului analizat este următorul:



Prin aplicarea algoritmui de transformare a modelului Entitate-Asociere în model relațional obținem următorul model relațional:

CLIENTI (CodClient, NumeClient, Adresa),
FACTURI (NrFactura, Data, #CodClient),
PRODUSE (CodProdus, Denumire, Pret),
DETALIIFACTURI (#NrFactura, #CodProdus, Cantitate)