1. **Date, informaţii, cunoştinţe**

Auzim adesea vorbindu-se despre “Era informaţiilor” sau “societate informaţională” sau “tehnologia informaţiei” însă de multe ori cuvântul "informaţie" este folosit fără a înţelege clar sensul acestui cuvânt, diferenţa dintre date, informaţii, cunoştinţe.

În general, conţinutul gândirii umane operează cu următoarele concepte:

1.       **Date** – constau în material brut, fapte, simboluri, numere, cuvinte, poze fără un înţeles de sine stătător, neintegrate într-un context, fără relaţii cu alte date sau obiecte. Ele se pot obţine în urma unor experimente, sondaje etc.

2.       **Informaţii** – prin prelucrarea datelor şi găsirea relaţiilor dintre acestea se obţin informaţii care au un înţeles şi sunt integrate într-un context. Datele organizate şi prezentate într-un mod sistematic pentru a sublinia sensul acestor date devin informaţii. Pe scurt informaţiile sunt date prelucrate. Informaţiile se prezintă sub formă de rapoarte, statistici, diagrame etc.

3.       **Cunoştinţele** sunt colecţii de date, informaţii, adevăruri şi principii învăţate, acumulate de-a lungul timpului. Informaţiile despre un subiect reţinute şi înţelese şi care pot fi folosite în luarea de decizii, formează judecăţi şi opinii devin cunoştinţe. Cu alte cuvinte, cunoştinţele apar în momentul utilizării informaţiei .

**2. Colectarea şi analizarea datelor. Modelul conceptual**

Primul pas în realizarea unei aplicaţii de baze de date este analiza datelor şi realizarea unei **scheme conceptuale (model conceptual)** al acestor date.

În această etapă sunt analizate natura şi modul de utilizare a datelor. Sunt identificate datele care vor trebui memorate şi procesate, se împart aceste date în grupuri logice şi se identifică relaţiile care există între aceste grupuri.

Analiza datelor este un proces uneori dificil, care necesită mult timp, însă este o etapă absolut obligatorie. Fără o analiză atentă a datelor şi a modului de utilizare a acestora, vom realiza o bază de date care putem constata în final că nu întruneşte cerinţele beneficiarului. Costurile modificării acestei baze de date este mult mai mare decât costurile pe care le-ar fi implicat etapa de analiză şi realizare a modelului conceptual. Modificarea modelului conceptual este mult mai uşoară decât modificarea unor tabele deja existente, care eventual conţin şi o mulţime de date. Ideea de bază a analizei datelor şi construirii modelului conceptual este "să măsori de două ori şi să tai o singură dată".

Informaţiile necesare realizării modelului conceptual se obţin folosind metode convenţionale precum intervievarea oamenilor din cadrul organizaţiei şi studierea documentelor folosite.

Odată obţinute aceste informaţii ele trebuiesc reprezentate într-o formă convenţională care să poată fi uşor înţeleasă de toată lumea. O astfel de reprezentare este ***diagrama entităţi-relaţii***, numită şi ***harta relaţiilor***, sau ***ERD***-ul (**E**ntity **R**elationship **D**iagram). Aceste scheme sunt un instrument util care uşurează comunicarea dintre specialiştii care proiectează bazele de date şi programatori pe de o parte şi beneficiari, pe de altă parte. Aceştia din urmă pot înţelege cu uşurinţă o astfel de schemă, chiar dacă nu sunt cunoscători în domeniul IT.

În concluzie putem sublinia câteva caracteristici ale ERD-urilor:

-       sunt un instrument de proiectare

-       sunt o reprezentare grafică a unui sistem de date

-       oferă un model conceptual de înalt nivel al bazelor de date

-       sprijină înţelegerea de către utilizatori a datelor şi a relaţiilor dintre acestea

-       sunt independente de implementare.

În cele ce urmează vom prezenta principalele elemente care intră în componenţa unui ERD precum şi convenţiile de reprezentare a acestora.

1. **Entităţi. Instanţe. Atribute. Identificator unic.**

O ***entitate*** este un lucru, obiect, persoană sau eveniment care are semnificaţie pentru afacerea modelată, despre care trebuie să colectăm şi să memorăm date. O entitate poate fi un lucru real, tangibil precum o clădire, o persoană, poate fi o activitate precum o programare  sau o operaţie, sau poate fi o noţiune abstractă.

O entitate este reprezentată în ERD printr-un dreptunghi cu colţurile rotunjite. Numele entităţii este întotdeauna un *substantiv la singular* şi se scrie în partea de sus a dreptunghiului cu *majuscule*, ca în figura I.1.1.



O entitate este de fapt o clasă de obiecte şi pentru orice entitate există mai multe ***instanţe*** ale sale. O instanţă a unei entităţi este un obiect, persoană, eveniment, particular din clasa de obiecte care formează entitatea. De exemplu, elevul **X** din clasa a IX-a A de la Liceul de Informatică din localitatea **Y** este o instanţă a entităţii **ELEV**.

După cum se vede pentru a preciza o instanţă a unei entităţi, trebuie să specificăm unele caracteristici ale acestui obiect, să-l descriem (precizăm de exemplu numele, clasa, şcoala etc). Aşadar, după ce am identificat entităţile trebuie să descriem aceste entităţi în termeni reali, adică să le stabilim ***atributele***. Un atribut este orice detaliu care serveşte la identificarea, clasificarea, cuantificarea, sau exprimarea stării unei instanţe a unei entităţi. Atributele sunt informaţii specifice ce trebuie cunoscute şi memorate.



De exemplu atributele entităţii **ELEV** sunt nume, prenume, adresa, număr de telefon, adresa de email, data naşterii etc.

În cadrul unui ERD, atributele se vor scrie imediat sub numele entităţii, cu litere mici. Un atribut este un *substantiv la singular* (vezi figura I.1.2).

Un atribut poate fi ***obligatoriu*** sau ***opţional***. Dacă un atribut este obligatoriu, pentru fiecare instanţă a entităţii respective trebuie să avem o valoare pentru acel atribut, de exemplu este obligatoriu să cunoaştem numele elevilor. Pentru un atribut opţional putem avea instanţe pentru care nu cunoaştem valoarea atributului respectiv. De exemplu atributul **email** al entităţii **ELEV**este opţional, un elev putând să nu aibă adresă de email. Un atribut obligatoriu este precedat în ERD de un asterisc **\***, iar un atribut opţional va fi precedat de un cerculeţ **o**.

Atributele care definesc în mod unic instanţele unei entităţi se numesc ***identificator unic*** (**UID**). UID-ul unei entităţi poate fi compus dintr-un singur atribut, de exemplu codul numeric personal poate fi un identificator unic pentru entitatea**ELEV**. În alte situaţii, identificatorul unic este compus dintr-o combinaţie de două sau mai multe atribute. De exemplu combinaţia dintre titlu, numele autorului şi data apariţiei poate forma unicul identificator al entităţii **CARTE**. Oare combinaţia titlu şi nume autor nu era suficientă? Răspunsul este NU, deoarece pot exista de exemplu mai multe volume scrise de Mihai Eminescu având toate titlul Poezii, dar apărute la date diferite.

****

Atributele care fac parte din identificatorul unic al unei entităţi vor fi precedate de semnul diez **#**(figura I.1.2 şi I.1.3). ***Atributele din UID sunt întotdeauna obligatorii***, însă semnul **#** este suficient, nu mai trebuie pus şi un semn asterisc în faţa acestor atribute.

Valorile unor atribute se pot modifica foarte des, ca de exemplu atributul vârstă. Spunem în acest caz că avem de a face cu un ***atribut volatil***. Dacă valoarea unui atribut însă se modifică foarte rar sau deloc (de exemplu data naşterii) acesta este un atribut ***non-volatil***. Evident este de preferat să folosim atribute non-volatile atunci când acest lucru este posibil.

**4. Relaţii între entităţi**

În lumea reală, obiectele nu există izolat.Intre ele exista relatii Aşadar, după ce aţi identificat care sunt entităţile şi atributele acestor entităţi este timpul să punem în evidenţă relaţiile care există între aceste entităţi, modul în care acestea comunică între ele. O ***relaţie*** este o asociere, legătură, sau conexiune existentă între entităţi şi care are o semnificaţie pentru afacerea modelată. Orice relaţie este bidirecţională, legând două entităţi sau o entitate cu ea însăşi. De exemplu, elevii studiază mai multe materii, o materie e studiată de către elevi.

Orice relaţie este caracterizată de următoarele elemente:

-       1. numele relaţiei ;                 2.opţionalitatea relaţiei;                                3. gradul (cardinalitatea) relaţiei.

Să luăm de exemplu relaţia existentă între entităţile **JUCĂTOR** şi **ECHIPĂ**. Vom spune:

Un **JUCĂTOR** joacă într-o **ECHIPĂ**. Si La o **ECHIPĂ** trebuie să joace unul sau mai mulţi **JUCĂTORI**.

-       **Numele relaţiei** este: *joacă*.

-       Pentru a stabili **opţionalitatea** relaţiei trebuie să răspundem la următoarea întrebare: Un jucător **trebuie** să joace într-o echipă? Se poate ca un jucător să nu joace în nici o echipă? Dacă acceptăm că toţi jucătorii trebuie să joace într-o echipă relaţia este obligatorie sau mandatorie şi vom spune:  Un **JUCĂTOR** ***trebuie*** să joace într-o **ECHIPĂ**.

Dacă însă acceptăm că există jucători care nu joacă în nici o echipă (de exemplu li s-a terminat contractul şi în momentul de faţă nu mai joacă la nici o echipă), atunci relaţia este opţională.

 În acest caz vom spune:                                                      Un **JUCĂTOR** ***poate*** juca la o **ECHIPĂ**.

-       **Cardinalitatea** relaţiei este dată de numărul de instanţe ale entităţii din partea dreaptă a relaţiei care pot intra în relaţie cu o instanţă a entităţii din partea stângă a relaţiei. Adică va trebui să răspundem la întrebări de genul: La câte echipe poate juca un jucător? Răspunsurile posibile sunt **unul şi numai unul**, sau **unul sau mai mulţi**. Vom spune:

                  Un **JUCĂTOR** trebuie/poate să joace la ***o*** **ECHIPĂ** ***şi******numai una***.

sau        Un **JUCĂTOR** trebuie/poate să joace la ***una sau mai multe*** **ECHIPE**.

Cea mai realistă varinată a relaţiei este aşadar:           Un **JUCĂTOR** poate să joace la o **ECHIPĂ** şi numai una.

**4.Convenţii de reprezentare a relaţiilor**

În cadrul diagramei entităţi-relaţii, o relaţie va fi reprezentată printr-o linie ce uneşte cele două entităţi.

Deoarece o relaţie este bidirecţională, linia ce uneşte cele două entităţi este compusă din două segmente distincte, câte una pentru fiecare entitate.  Tipul segmentului ce pleacă de la o entitate ne va indica **opţionalitatea** relaţiei dintre această entitate şi entitatea aflată în cealaltă parte a relaţiei. Dacă acest segment este continuu este vorba de o relaţie obligatorie, o linie întreruptă indică o relaţie opţională.

De exemplu în figura I.1.4 segmentul ce pleacă de la entitatea JUCĂTOR fiind ***întreruptă*** înseamnă că un jucător ***poate*** juca la o echipă, adică relaţia este opţională. Segmentul ce pleacă dinspre entitatea ECHIPĂ este ***continuă***, deci la o echipă ***trebuie*** să joace jucători.

http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image008.jpg

**Figura I.1.4.**Reprezentarea relaţiilor

Modul în care o linie se termină spre o entitate este important. Dacă se termină printr-o linie simplă, înseamnă că o instanţă şi numai una a acestei entităţi este în relaţie cu o instanţă a celeilalte entităţi. În exemplul anterior, linia de la**JUCATOR** la **ECHIPĂ** se termină în partea dinspre **ECHIPĂ** cu o ***linie simplă***, deci un jucător joacă la ***o***echipa ***şi numai una***.

Dacă linia se termină cu trei linii (picior de cioară)  înseamnă că mai multe instanţe ale entităţii pot corespunde unei instanţe a celeilalte entităţi. În exemplul anterior linia de la **ECHIPĂ** la **JUCĂTOR** se termină cu ***piciorul de cioară***, înseamnă că unei instanţe a entităţii **ECHIPĂ** îi corespund mai multe instanţe ale entităţii **JUCĂTOR**, adică o echipă are ***unul sau mai mulţi*** jucători.

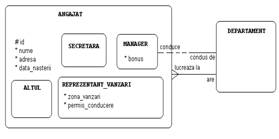
|  |  |  |
| --- | --- | --- |
| **Caracteristica relaţiei** | **Valoare** | **Mod de reprezentare** |
| **Numele relaţiei** | un verb | se scrie deasupra relaţiei |
| **Opţionalitatea** | relaţie obligatorie  (TREBUIE) | linie continuăhttp://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image010.jpg |
| relaţie opţională  (POATE) | linie întreruptăhttp://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image012.jpg |
| **Cardinalitatea** | una şi numai una | linie simplă              http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image014.jpg |
| una sau mai multe | picior de cioarăhttp://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image016.jpg |

**Tipuri şi subtipuri**

În lumea reală obiectele sunt deobicei clasificate. Astfel vorbim despre animale vertebrate şi nevertebrate, despre licee teoretice, colegii, grupuri şcolare etc. E normal ca în modelarea bazelor de date să putem modela şi astfel de clasificări.

Un **subtip** sau o **subentitate** este o clasificare a unei entităţi care are caracteristici comune cu entitatea generală, precum atribute şi relaţii. Subtipurile se reprezintă în cadrul hărţii relaţiilor ca entităţi în interiorul altei entităţi. Atributele şi relaţiile comune tuturor subtipurilor se vor reprezenta la nivelul **supertipului**, sau **superentităţii**. Atributele şi relaţiile supertipului vor fi moştenite de către subtipuri.

Un subtip poate avea la rândul său alte subtipuri incluse.



**Figura I.4.1.** Folosirea subtipurilor şi supertipurilor

Subtipurile trebuie să respecte două reguli importante:

-     trebuie să acopere toate cazurile posibile de instanţe ale supertipului, cu alte cuvinte, orice instanţă a supertipului trebuie să aparţină unui subtip. De multe ori ERD-urile includ un subtip "ALTUL" pentru a acoperi toate situaţiile, şi pentru a permite viitoare dezvoltări ale modelului.

subtipurile trebuie să se excludă reciproc. Această regulă se traduce pe exemplul de mai sus în faptul că un angajat nu poate fi, de exemplu, şi manager şi secretară în acelaşi timp.

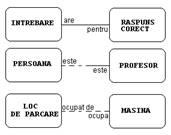
**Documentare Business Rules**

Pentru ca modelul conceptual sa fie complet se definesc **reguli structurale** (-indica tipuri de info ce vor fi stocate si  cum relationeaza ele) si reguli procedurale (legate de timp , etc, -acestea ne se repr pe ERD, ci trebuie implementate in programare ).

**Tipuri de relaţii**

Variantele de relaţii ce pot exista între două entităţi sunt prezentate mai jos:

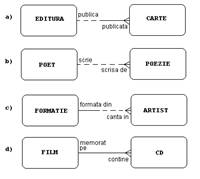
-       **relaţii one-to-one** – acest tip de relaţie este destul de rar întâlnit. Uneori astfel de relaţii pot fi modelate transformând una dintre entităţi în atribut al celeilalte entităţi.

-   

**Figura I.1.5.** Relaţii one-to-one

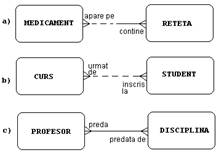
-       **relaţii one-to-many** – sunt cele mai întâlnite tipuri de relaţii, însă şi aici cazurile c şi d prezentate în figura I.1.6 sunt mai puţin uzuale.

Să facem câteva observaţii pe marginea exemplelor din figura I.1.6. **Cazul a** este foarte des întâlnit. La **cazul b**, am ales o relaţie opţională dinspre **POEZIE** spre **POET** deoarece poate fi vorba de o poezie populară şi în acest caz nu există un poet cunoscut. La **cazul c**, am considerat că o formaţie nu poate exista fără a avea cel puţin un membru, însă un artist poate avea o carieră solo, deci nu face parte din nici o formaţie.  **Varianta d** modelează o colecţie de filme memorate pe CD-uri. Pentru afacerea considerată, un CD conţine obligatoriu un film, dar unul singur, însă un film poate să nu încapă pe un singur CD de aceea el este poate fi memorat pe unul sau mai multe CD-uri.



**Figura I.1.6.** Relaţii one-to-many

-       **relaţii many-to-many** – aceste tipuri de relaţii apar în prima fază a proiectării bazei de date, însă ele trebuie să fie ulterior eliminate. Figura I.1.7 prezintă câteva exemple de relaţii many-to-many. La punctul b am considerat că un curs poate apărea pe oferta de cursuri a unei facultăţi, însă poate să nu fie aleasă de nici un student de aceea un curs **poate**fi urmat de unul sau mai mulţi studenţi. Invers, este posibil ca un student să fi terminat studiile şi să se pregătească pentru susţinerea examenului de licenţă şi de aceea el nu mai frecventează nici un curs. La punctul c, un profesor angajat al unei şcoli **trebuie** să predea cel puţin o disciplină. Iar o disciplină din planul de învăţământ trebuie să fie predată de cel puţin un profesor.

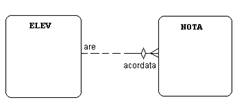


**Figura I.1.7.** Relaţii many-to-many

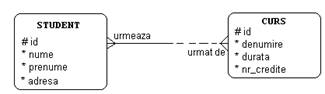
**Transferabilitate**

Spunem că o relaţie este nontransferabilă dacă o asociaţie între două instanţe ale celor două entităţi, odată stabilită, nu mai poate fi modificată. Nontransferabilitatea unei relaţii se reduce la faptul că valorile cheii străine corespunzătoare relaţiei respective nu pot fi modificate.Condiţia de nontransferabilitate a unei relaţii este asigurată prin program. De aceea trebuie să documentăm această restricţie.În ERD o relaţie nontransferabilă se notează cu un romb pe linia corespunzătoare relaţiei, înspre entitatea a cărei cheie străină nu este permis să o modificăm (adică în partea cu many a unei relaţii one-to-many).

În figura I.4.5 este dat un exemplu de relaţie nontransferabilă. Este vorba despre notele date elevilor. Este normal ca o notă dată unui elev să nu poată fi apoi transferată unui alt elev.



**Figura I.4.5.**Relaţii nontransferabile

******Rezolvarea relaţiilor many-to-many**

După cum am precizat mai devreme relaţiile many-to-many pot apărea într-o primă fază a proiectării bazei de date însă ele nu au voie să apară în schema finală. Să considerăm relaţia din figura I.1.14 dintre entităţile STUDENT şi CURS. Se ştie că orice curs se termină în general cu un examen. Unde vom memora nota studentului la fiecare examen?

**Figura I.1.14**

Dacă încercăm să introducem atributul **NOTA** la entitatea **STUDENT**, nu vom şti cărei materii corespunde acea notă, întrucât unei instanţe a entutăţii student îi corespund mai multe instanţe ale entităţii **CURS**. Invers dacă încercăm să memorăm nota în cadrul entităţii **CURS**, nu vom ştii cărui student îi aparţine acea notă.

Rezolvarea unei relaţii many-to-many constă introducerea unei noi entităţi numită ***entitate de intersecţie***, pe care o legăm de entităţile originale prin câte o relaţie one-to-many.

Paşii în rezolvarea unei relaţii many-to-many sunt următorii:

1)   se găseşte entitatea de intersecţie, pentru exemplul nostru vom introduce entitate **INSCRIERE**.

2)       crearea noilor relaţii

         opţionalitatea: relaţiile care pleacă **din entitatea de intersecţie sunt întotdeauna obligatorii în această parte**. În partea dinspre entităţile originale, relaţiile vor păstra opţionalitatea relaţiilor iniţiale.

         cardinalitatea: ambele relaţii sunt de tip one-to-many, iar partea cu many va fi întotdeauna înspre entitatea de intersecţie.

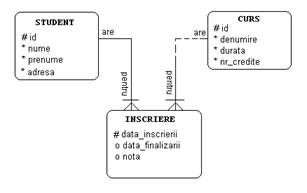
         numele noilor relaţii

3)       adăugarea de atribute în cadrul entităţii de intersecţie, dacă acestea există. În exemplul nostru ne poate interesa de exemplu data la care s-a înscris un student la un curs, data la care a finalizat cursul precum şi nota obţinută la sfârşitul cursului.

4)       stabilirea identificatorului unic pentru entitatea de intersecţie: dacă entitatea de intersecţie nu are un identificator unic propriu, atunci acesta se poate forma din identificatorii unici ai entităţilor iniţiale la care putem adăuga atribute ale entităţii de intersecţie.

În exemplul nostru, identificatorul unic al entităţii de intersecţie este format din id-ul studentului, id-ul cursului şi data înscrierii la curs.

5)       Faptul că identificatorul  unic al unei entităţi preia identificatorul unic din altă entitate cu care este legată este reprezentat grafic prin bararea relaţiei respective, înspre entitatea care preia UID-ul celeilalte entităţi.



**Analiza CRUD-**se refera la CREATE, RETRIVE, UPDATE, DELETE-(crea , reface, actualiza, sterge) operatii ce fac din ERD un model complet.Se verifica daca modelul exprima toate operatiile ce se pot face si nu are elem inutile, etc.

**UID artificial si compus**

UID-(Unique Identifier)-e atributul ce identifica in mod unic entitatea(ex: CNP, cod, id,). Daca e nevoie de o combinatie de mai multe atribute care sa identifice in mod unic entitatea , e vorba de un UID compus. Daca se recurge la o modalitate de identificare printr-un cod artificial oferit in mod automat de program, e vorba de UID artificial.

**Ce este normalizarea?**

Normalizarea este o tehnică de proiectare a bazelor de date prin care se elimină (sau se evită) anumite anomalii şi inconsistenţe a datelor. O baza de date bine proiectată nu permite astfel ca datele să fie redundante, adică aceeaşi informaţie să se găsească în locuri diferite, sau să memorezi în baza de date, informaţii care se pot deduce pe baza altor informaţii memorate în aceeaşi bază de date. Anomaliile care pot să apară la o bază de date nenormalizată sunt următoarele:

**anomalii la actualizarea** datelor la o bibliotecă se înregistrează într-o tabelă următoarele date despre cărţi: ISBN, titlu, autor,  preţ, subiect, editura, adresa editurii. La un moment dat o editură îşi schimbă adresa. Bibliotecara va trebui să modifice adresa editurii respective, în înregistrările corespunzătoare tuturor cărţilor din bibliotecă apărute la respectiva editură. Dacă această modificare nu se face cu succes, unele dintre înregistrări rămânând cu vechea adresă, apare din nou o inconsistenţă a datelor.

-                **anomalii de inserare** – în exemplul anterior, nu vom putea memora adresa unei edituri, lucru inacceptabil dacă  dorim să avem informaţii şi despre edituri a căror cărţi nu le avem în bibliotecă, eventual de la care dorim să facem comenzi.

-                **anomalii de ştergere** – să presupunem că într-o tabelă memorăm următoarele informaţii: codul studentului, codul cursului, codul profesorului. La un moment dat, nici un student nu mai doreşte să participe la un anume curs. Ştergând toate înregistrările corespunzătoare cursului, nu vom mai putea şti niciodată cine preda acel curs.

Edgar Codd a definit primele trei forme normale 1NF, 2NF şi 3NF. Ulterior s-au mai definit formele normale 4NF, 5NF, 6NF care însă sunt rar folosite în proiectarea bazelor de date.

**Prima formă normală**

**O entitate se găseşte în prima formă normală dacă şi numai dacă:- nu există atribute cu valori multiple;- nu există atribute sau grupuri de atribute care se repetă**. Cu alte cuvinte toate atributele trebuie să fie atomice, adică să conţină o singură informaţie.

Dacă un atribut are valori multiple, sau un grup de atribute se repetă, atunci trebuie să creaţi o entitate suplimentară pe care să o legaţi de entitatea originală printr-o relaţie de 1:m. În noua entitate vor fi introduse atributele sau grupurile de atribute care se repetă.

Să considerăm entitatea din figura I.2.1, referitoare la notele elevilor unei clase. Câteva observaţii referitoare la această entitate: câte discipline are un elev? Câte perechi (disciplina, nota)  va trebui să aibă entitatea Elevi? Să spunem că ştim exact câte discipline maxim poate studia un elev. Ce se întâmplă dacă în anul viitor şcolar acest număr de discipline va fi mai mare? În plus, la o materie un elev poate avea mai multe note. Câte note? Cum memorăm aceste note? Le punem în câmpul corespunzător disciplinei cu virgulă între ele?

Cum rezolvăm această problemă? Vom crea o nouă entitate în care vom introduce disciplina şi nota la disciplina respectivă (vezi figura I.2.2.).

                În acest fel fiecărui elev îi pot corespunde oricâte note, iar la o disciplină poate avea oricâte note, singura restricţie conform acestui model fiind că un elev nu va putea primi în aceeaşi zi la aceeaşi materie mai multe note.

|  |  |
| --- | --- |
| http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image032.jpg  **Figura I.2.1.** | http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image034.jpg  **Figura I.2.2** |

Un alt exemplu de încălcare a regulilor primei formei normale, puţin mai "ascuns", este prezentat în figura I.2.5. De ce? Pentru că adresa este de forma "str. Florilor, bl. 45, sc. A, ap. 28, etaj 3, Braşov, cod 123123", formă care de fapt conţine mai multe informaţii elementare. Aşadar, în mod normal acest atribut ar trebui "spart" în mai multe atribute ca în figura I.2.6.

|  |  |
| --- | --- |
| http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image036.jpg  **Figura I.2.5** | http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image038.jpg  **Figura I.2.6.** |

                Noile atributele introduse sunt opţionale întrucât dacă elevul locuieşte la casă, probabil atributele bloc, apartament, scara, etaj, nu au sens. Invers dacă elevul locuieşte la bloc, probabil nu poate fi completat numărul.

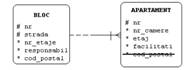
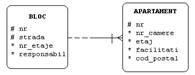
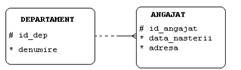
                Pentru acest tip de încălcare a regulilor formei normale 1NF poate fi totuşi ignorată, decizia depinzând de natura fenomenului, sau afacerii modelate. În exemplul anterior, întrucât datele din interiorul unei adrese este puţin probabil să se modifice, modificându-se el mult adresa completă a unui elev, se poate decide să nu operăm modificarea anterioară. Dacă însă aceste informaţii s-ar modifica frecvent, de exemplu denumirile străzilor s-ar modifica mereu, atunci probabil modificarea este de dorit.

**A doua formă normală**

O entitate se găseşte în a doua formă normală **dacă şi numai dacă se găseşte în prima formă normală şi în plus orice atribut care nu face parte din UID (unique identifier) va depinde de întregul UID nu doar de o parte a acestuia**.

De exemplu dacă memorăm angajaţii unui departament într-o entitate ca mai jos:

Se observă că **data\_nasterii** şi adresa sunt două atribute care depind doar de id-ul angajatului nu de întregul UID care este combinaţia dintre atributele **id\_dep** si **id\_angajat**. Această situaţie se rezolvă prin crearea unei noi entităţi **ANGAJAT**, pe care o legăm de entitatea **DEPARTAMENT** printr-o relaţie **1:m**.



O situaţie mai specială este în cazul relaţiilor barate, când trebuie ţinut seama că UID-ul unei entităţi este compus din atribute din entitatea respectivă plus un atribut sau mai multe atribute provenite din relaţia barată. Să considerăm următorul exemplu:

Se observă că UID-ul entităţii **APARTAMENT** este compus din combinaţia a trei atribute: numărul apartamentului, numărul blocului şi strada. Deci toate atributele din entitatea **APARTAMENT** care nu fac parte din UID, trebuie să depindă de întregul UID. Dar se ştie că atributul cod\_postal depinde doar de strada si de numărul blocului, nu şi de numărul apartamentului. Acest lucru ne spune ca acest atribut nu este memorat la locul potrivit. Deoarece depinde doar de combinaţia  (strada, nr\_bloc), înseamnă că de fapt depinde de UID-ul entităţii **bloc**. Aşadar vom muta atributul cod\_postal în entitatea **BLOC**.

***Observaţie.***Dacă o entitate se găseşte în prima formă normală şi UID-ul său este format dintr-un singur atribut atunci ea se găseşte automat în a doua formă normală.

**A treia formă normală**

O entitate se găseşte în a treia formă normală dacă şi numai dacă se găseşte **în a doua formă normală şi în plus nici un atribut care nu este parte a UID-ului nu depinde de un alt atribut non-UID**. Cu alte cuvinte nu se acceptă dependenţe tranzitive, adică un atribut să depindă de UID în mod indirect.

Luăm ca exemplu entitatea **CARTE** din figura I.2.10. Atributul **biografie\_autor** nu depinde de **ISBN** ci de atributul **autor**. Nerezolvarea acestei situaţii duce la memorarea de date redundante, deoarece biografia unui autor va fi memorată pentru fiecare carte scrisă de autorul respectiv. Rezolvarea acestei situaţii este să creăm o nouă entitate **AUTOR**, pe care o legăm de entitatea **CARTE** printr-o relaţie **1:m** (figura I.2.11.).

|  |  |
| --- | --- |
| http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image048.jpg  **Figura I.2.10.** | http://paulierco.ro/wp-content/uploads/2008/02/Design-rezumat_files/image050.jpg  **Figura I.2.11.** |

                Atributul nu por avea alte atribute, asa ca el devine entitate.

**3. Relaţii exclusive (arce)**

În unele situaţii, relaţiile se pot exclude reciproc, adică dintr-un grup de relaţii, la un moment dat doar una dintre ele poate avea loc. De exemplu, un cont anume la o bancă este deţinut fie de o persoană fizică fie de o firmă dar nu de ambele tipuri de clienţi simultan. Un grup de relaţii exclusive este reprezentat în harta relaţiilor printr-un arc peste relaţiile care fac parte din respectivul grup, ca în figura I.4.2. Toate relaţiile ce fac parte din grupul de relaţii exclusive trebuie să aibă aceeaşi opţionalitate. Un arc aparţine unei singure entităţi, adică va include doar relaţii care pleacă de la o aceeaşi entitate.

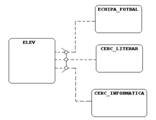
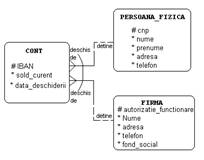
O entitate poate avea mai multe arce, dar o anumită relaţie nu poate face parte decât dintr-un singur arc.

Există două tipuri de relaţii exclusive:

- relaţii exclusive ***obligatorii*** în care toate relaţiile ce fac parte din arcul respectiv sunt obligatorii, ceea ce înseamnă că de fiecare dată, una dintre relaţii are obligatoriu loc. Este şi cazul din figura 1 Evident că un cont trebuie să fie deţinut de o persoană fizică sau de o firmă, o a treia variantă neexistând.

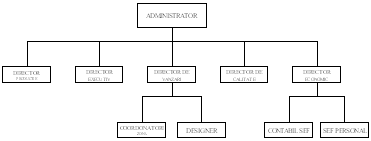
-  relaţii exclusive ***opţionale*** caz în care toate relaţiile ce fac parte din arc sunt opţionale. În acest caz de fiecare dată are loc cel mult una dintre relaţii, existând varianta ca pentru o instanţă a entităţii căreia aparţine arcul să nu aibă loc nici una din relaţiile din grupul respectiv. În figura 2, este exemplificată situaţia în care un elev poate opta să facă parte din echipa de fotbal, sau să participe la cercul literar sau la cercul de informatică. Însă regulile şcolii prevăd ca un elev să nu participe la două astfel de activităţi extraşcolare. Relaţiile fiind opţionale, înseamnă că un elev are libertatea de a decide să nu participe la nici o activitate extraşcolară.

**.**Relaţii exclusive obligatorii                                                           Relaţii exclusive opţionale



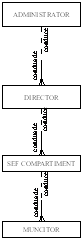
**Relaţii ierarhice. Relaţii recursive**

Haideţi să analizăm care este structura personalului într-o firmă oarecare. În figura I.1.8 este prezentată doar o parte din organigrama unei firme.



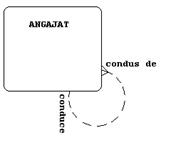
**Figura I.1.8.** Organigrama unei firme

                Un model de proiectare a unei astfel de structuri într-o bază de date ar fi cea din figura următoare:



**Figura I.1.9.** Implementarea unei structuri ierarhice

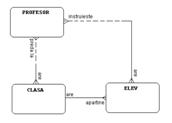
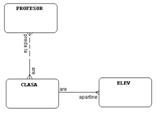
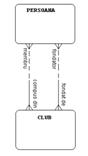
Problema este că fiecare tip de angajat din figura anterioară este de fapt un angajat şi probabil există foarte multe atribute comune tuturor acestor entităţi ca de exemplu nume, prenume, adresă, telefon, email, data naşterii etc. Vom putea de aceea modela această structură cu ajutorul unei singure entităţi numită **ANGAJAT**. Însă fiecare angajat poate fi condus de către un alt angajat. Aşadar vom avea o relaţie de la entitatea **ANGAJAT** la ea însăşi. O astfel de relaţie se numeşte ***relaţie recursivă***.



**Figura I.1.10.** Implementarea unei structuri ierarhice folosind relaţii recursive

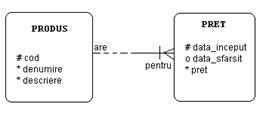
**Relaţii redundante si  multiple**

Atunci când o relaţie poate fi dedusă din alte relaţii spunem că acea relaţie este redundantă. Relatia se poate elimina.pot exista si  relaţii multiple între entităţi



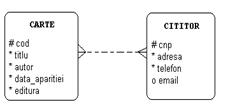
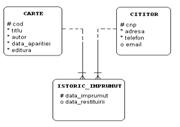
**Modelarea datelor istorice**

Viaţa înseamnă schimbare, orice lucru se schimbă de-a lungul timpului, şi nu doar obiectele se modifică în timp dar chiar şi relaţiile dintre aceste obiecte se schimbă. Preţul produselor poate suferi modificări destul de des. Factorii care duc la aceste modificări pot fi dintre cei mai diverşi, rata inflaţia, anotimpul etc. Aşadar atributul **preţ** din cadrul entităţii **produs** se modifică de-a lungul timpului. Dacă nu ne interesează decât preţul actual al fiecărui produs modelul este foarte simplu, ca cel din fig.Dacă însă pentru afacerea modelată este important să reţinem un istoric al preţurilor pentru fiecare produs, atunci atributul preţ se va transforma într-o nouă entitate

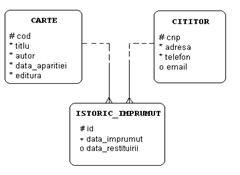
Atributul **data\_sfarsit**este opţional, deoarece data până la care este valabil preţul curent al unui produs nu este de obicei cunoscut.

Vom considera acum o situaţie puţin mai dificilă. Să presupunem că dorim să modelăm o bază de date pentru o bibliotecă. Evident este important de reţinut un istoric al tuturor împrumuturilor, deoarece pe baza acestora, se pot afla domeniile de interes ale cititorilor, şi astfel vom şti ce achiziţii de carte să facem în viitor, vom putea determina uzura cărţilor astfel încât să le putem înlocui etc.

Într-o primă fază vom obţine o relaţie de many-to-many între entităţile **CARTE**şi **CITITOR**. Fiecare carte poate fi împrumutată de mai mulţi cititori (evident nu în acelaşi timp), şi fiecare cititor poate împrumuta mai multe cărţi .



Să rezolvăm această relaţie many-to-many. Aplicând ceea ce am învăţat în capitolele anterioare vom obţine schema din fig. a 2 a

Să verificăm că acest caz este cel corect. Cheia primară este acum combinaţia coloanelor **cod\_carte** şi **data\_imprumut**. Poate un cititor împrumuta două cărţi în aceeaşi dată? Adică următoarele două înregistrări pot exista simultan în tabela **ISTORIC\_IMPRUMUTURI**? Răspunsul este DA, combinaţia celor două coloane, pentru cele două înregistrări fiind unică.

Deci bararea automată a celor două relaţii dinspre entitatea de intersecţie nu este întotdeauna o soluţie corectă. Pentru a evita aceste complicaţii putem recurge la introducerea unei chei artificiale în entitatea de intersecţie. În exemplul nostru se poate decide ca pentru fiecare împrumut în parte să se completeze câte o fişa separată care are un număr unic. Obţinem modelul din figura I.4.13, care este de asemenea unul corect.

**Fig.**Introducerea unei chei artificiale

**Conventii de ridabilitate**: Se aplica conventiile Oracle de scriere a ERD-ului:

Entitatea se scrie cu majuscule, singular in interiorul unui dreptunghi cu vf rotunjite.Atributele se scriu cu litere mici , avand in fata unul din semnele #,\*,o(UID, obligatoriu, optional).Orientarea liniilor este de la V la E si de sus in jos, evitand intersectia. Se pot folosi subdiagrame de explicare a diagramelor complexe, si explicarea entitatilor cu multe atribute.

**Modelarea generica**

Modelul generic aduce beneficii daca cerintele afacerii se schimba des. Atunci e nevoie de entitati si atribute noi.Se poate modela o singura entitate **Article type**care sa pastreze oricate tipuri de articole e nevoie, aceasta reduce nr de entitati.

**Procesul maparii**

Transformarea modelului conceptual, a ERD-ului, în modelul fizic, adică în baza de date propriu zisă, se numeşte mapare. Acest proces implică transformarea fiecărui element al ERD-ului.

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| titlu | Varchar2 | Pk | \* |
| autor | Varchar2 | Pk | \* |
| data\_apariţiei | Date |  | \* |
| Format | Varchar2 |  | \* |
| Nr\_pagini | Number |  | \* |

Entitati    tabele, (CARTE-carti.dbf)

atribute  campuri,

coloane, UIDcheie primara,

relatiecheie straina,

business rules constrangeri

Se mapeaza procesul de transformare in diagrama tabelei:

Tipuri de date in Oracle:

|  |  |  |
| --- | --- | --- |
| **Tipul de date** | **Descriere** | **Dimensiune Maximă** |
| **VARCHAR2** | Şir de caractere de lungime variabilă | **4000**bytes |
| **CHAR** | Şir de caractere de lungime fixă | **2000** bytes |
| **NUMBER(p,s)** | Număr având p cifre din care s la partea zecimală. (s negativ reprezintă numărul de cifre semnificative din faţa punctului zecimal) | p (precizia) între 1 şi 38.  s (scala) între -84 şi 127. |
| **DATE** | Dată calendaristică | De la  1 Ianuarie 4712 BC pana la  31 Decembrie, **9999** AD. |
| **TIMESTAMP** | Se memorează data calendaristică, ora, minutul, secunda şi fracţiunea de secundă | Fracţiunea de secundă este memorată cu o precizie de la 0 la 9. |
| **INTERVAL YEAR**  **TO MONTH** | perioadă de timp în ani şi luni. |  |
| **INTERVAL DAY**  **TO SECOND** | memorează un interval de timp în zile, ore, minute şi secunde |  |
| **CLOB** | Character Large Object | 4 Gigabytes |
| **BLOB** | Binary Large Object | 4 Gigabytes |
| **BFILE** | Se memorează adresa unui fişier binar de pe disc | 4 Gigabytes |

dacă relaţia pe partea many este opţională atunci şi coloanele cheii străine vor fi opţionale. Ce înseamnă acest lucru? Faptul că un jucător poate la un moment dat să nu joace la nici o echipă, atunci câmpul cod\_echipă va rămâne necompletat în dreptul lui (va avea valoarea NULL). Dacă însă relaţia este obligatorie pe partea many atunci coloanele ce fac parte din cheia străină vor fi opţionale.

În gereral, la maparea unei relaţii de tip one-to-many, vom introduce în tabela corespunzătoare entităţii de pe partea many a relaţiei cheia primară a entităţii de pe partea one a relaţiei. Câmpurile astfel întroduse se vor numi **cheie străină**(foreign key).

Aşadar:

-          cheia străină a unei tabele este cheia primară din tabela referintă

-          cheia străină este întotdeauna introdusă în tabela corespunzătoare entităţii din partea many a relaţiei.

**Maparea relaţiilor one-to-one**

Dându-se două entităţi **A** şi **B** legate între ele printr-o relaţie one-to-one, este evident că putem include cheia primară **A** în cadrulul tabelei **B**, dar putem proceda la fel de bine şi invers, incluzând cheia primară a tabelei **B** în cadrul tabelei **A**, deoarece fiecărei instanţe a entităţii A îi corespunde cel mult o instanţă a entităţii **B**, dar şi invers, oricărei instanţe a entităţii **B** îi corespunde cel mult o instanţă a entităţii **A**.

Pentru relaţia din figura I.3.3 de exemplu putem memora pentru fiecare persoană seria de paşaport, dar şi invers, pentru fiecare paşaport putem memora cnp-ul deţinătorului.

**Figura I.3.3.**

                Decizia depinde de specificul afacerii modelate. Dacă de exemplu ne interesează în primul rând persoanele şi abia apoi datele de pe paşapoarte, atunci vom adopta probabil prima variantă, a memorării seriei de paşaport în cadrul tabelei **PERSOANE**, dacă însă baza de date este destinată evidenţei paşapoartelor, atunci probabil vom adopta varianta a doua.

Uneori este convenabil să memorăm cheia străină în ambele părţi ale relaţiei, în exemplul nostru pentru fiecare paşaport să memorăm cnp-ul persoanei care îl deţine, dar şi pentru fiecare persoană să memorăm seria de paşaport.

**Maparea relaţiilor recursive**

Dacă vom privi o relaţie recursivă ca pe o relaţie de tipul one-to-many între o entitate şi ea însăşi, atunci acest caz se reduce la ceea ce deja am discutat. Să exemplificăm relaţia din figura I.3.4. Relaţia recursivă din această figură poate fi privită ca o relaţie între două entităţi identice, ca în figura I.3.5.

|  |  |
| --- | --- |
| **Figura I.3.4.** | **Figura I.3.5.** |

                Aşadar vom introduce în cadrul tabelei **ANGAJAŢI**, marca şefului său. Diagrama de tabela va arăta ca mai jos.

**Tabelul I.3.4.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Marca | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Prenume | Varchar2 |  | \* |
| Data\_angajarii | Date |  | \* |
| Adresa | Varchar2 |  | \* |
| Telefon | Varchar2 |  | o |
| Email | Varchar2 |  | o |
| **Marca\_sef** | **Number** | **Fk** | **o** |

**I.3.4. Maparea relaţiilor barate**

Relaţiile barate sunt mapate ca cheie străină în tabela aflată în partea many a relaţiei, la fel ca la maparea oricărei relaţii one-to-many. Bara de pe relaţie exprimă faptul că acele coloane ce fac parte din cheia străină vor devenii parte a cheii primare a tabelei din partea many a relaţiei barate.

Pentru exemplul din figura I.3.6, cheia primară a tabelei **ATRIBUTE** va fi format din coloanele **denumire\_atribut** şi **denumire\_entitate**, aceasta din urmă fiind de fapt cheie străină în tabela **ATRIBUTE**.

**Figura I.3.6.** Maparea relaţiilor barate

**Tabelul I.3.5.**Tabela **ENTITĂŢI**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| denumire | Varchar2 | Pk | \* |

**Tabelul I.3.5.**Tabela **ATRIBUTE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| denumire\_atribut | Varchar2 | Pk | \* |
| denumire\_entitate | Varchar2 | **Pk, Fk** | \* |
| optionalitate | Varchar2 |  | \* |

Să considerăm acum un exemplu în care există mai multe relaţii barate, în cascadă.

**Figura I.3.7.** Relaţii barate în cascadă

**Tabelul I.3.6.**Tabela **A                                Tabelul I.3.7.**Tabela **B**

|  |  |  |
| --- | --- | --- |
| **Numele coloanei** | **Tip cheie** | **Opţionalitate** |
| **idA** | **Pk** | \* |
| C1 |  | \* |

|  |  |  |
| --- | --- | --- |
| **Numele coloanei** | **Tip cheie** | **Opţionalitate** |
| **idB** | **Pk** | \* |
| C2 |  | \* |
| **idA** | **Pk, Fk** | \* |

**Tabelul I.3.8.**Tabela **C                                      Tabelul I.3.9.**Tabela **D**

|  |  |  |
| --- | --- | --- |
| **Numele coloanei** | **Tip cheie** | **Opţionalitate** |
| **idC** | **Pk** | \* |
| C3 |  | \* |
| **idA** | **Pk, Fk** | \* |
| **idB** | **Pk, Fk** | \* |

|  |  |  |
| --- | --- | --- |
| **Numele coloanei** | **Tip cheie** | **Opţionalitatea** |
| **idD** | **Pk** | \* |
| C4 |  | \* |
| **idA** | **Fk** | \* |

**. Operaţii specifice prelucrării bazelor de date**

Orice sistem de gestiune a bazelor de date (SGBD) trebuie să asigure următoarele **funcţii**:

         definirea structurii bazei de date

         încărcarea datelor în baza de date (adăugarea de noi înregistrări la baza de date)

         accesul la date pentru:

o    interogare (afişarea datelor, sortarea lor, calcule statistice etc.)

o    ştergere

o    modificare

         întreţinerea bazei de date:

o    refacerea bazei de date prin existenţa unor copii de siguranţă

o    repararea în caz de incident

o    colectarea şi refolosirea spaţiilor goale

         posibilitatea de reorganizare a bazei de date prin:

o    restructurarea datelor

o    modificarea accesului la date

         securitatea datelor.

O parte din aceste operaţii pot fi realizate cu ajutorul limbajului SQL, altele cu ajutorul unor programe specializate, care sunt puse la dispoziţia administratorului bazei de date de către sistemul de gestiune al bazelor de date.

**. Reguli de integritate**

Detalierea caracteristicilor pe care trebuie să le prezinte un SGBD pentru a fi considerat relaţional s-a făcut de E. F. Codd în 1985 sub forma a 13 reguli. Una dintre aceste reguli precizează că restricţiile de integritate trebuie să poată fi definite în limbajul utilizat de SGBD pentru definirea datelor.

Regulile de integritate garantează că datele introduse în baza de date sunt corecte şi valide. Aceasta înseamnă că dacă există orice o regulă sau restricţie asupra unei entităţi, atunci datele introduse în baza de date respectă aceste restricţii. În Oracle, regulile de integritate se definesc la crearea tabelelor folosind **constrângerile**. Dar asupra acestora vom reveni în partea a doua a manualului.

Tipurile de reguli de integritate sunt următoarele:

  **Integritatea entităţilor** – indică faptul că nici o coloană ce face parte din cheia primară nu poate avea valoarea NULL. În plus, pentru fiecare înregristrare, cheia primară trebuie să fie unică.

  **Integritatea** **de domeniu** – acest tip de reguli permite ca într-o anumită coloană se introducă doar valori dintr-un anumit domeniu. De exemplu putem impune ca salariul unui angajat să fie cuprins între 4500 şi 5000 RON.

  **Integritatea referenţială**– este o protecţie care asigură ca fiecare valoare a cheii străine să corespundă unei valori a cheii primare din tabela referită. De exemplu, referindu-ne la tabelele **JUCĂTORI** şi **ECHIPE**, corespunzătoare ERD-ului din figura I.3.2, **cod** este cheie primară în tabela **ECHIPE**, iar în tabela **JUCĂTORI**, **cod** devine cheie străină. Astfel valoarea câmpului **cod** din cadrul tabelei **JUCĂTORI** corespunzătoare unui anumit jucător trebuie să se regăsească printre valorile câmpului **cod** din tabela **ECHIPE**, altfel ar însemna că jucătorul respectiv joacă la o echipă inexistentă (vezi figura I.3.8).

**Figura I.3.8.** Exemplu de încălcare a integrităţii referenţiale

Situaţii de încălcare a integrităţii referenţiale pot apărea:

  la **adăugarea** unei noi înregistrări în baza de date, se poate încerca introducerea unor valori invalide pentru câmpurile cheii străine;

  la **actualizarea** bazei de date;

   la **ştergerea** unei înregistrări. De exemplu se şterge înregistrarea corespunzătoare unei anumite echipe (echipa se desfiinţează). Înregistrările jucătorilor care au jucat la acea echipă vor încălca integritatea referenţială, deoarece se vor referi la o echipă care nu mai există. Soluţiile posibile sunt ca la ştergerea unei echipe, toţi jucătorii care au activat la acea echipă să fie şi ei şterşi din baza de date (ştergere în cascadă) sau valoarea câmpului **cod\_echipă** pentru acei jucători să fie setată la **NULL**, ceea ce va înseamnă că acei jucători nu activează la nici o echipă.

**Programe de validare şi de acţiune**

În realizarea modelului conceptual al unei baze de date se ţine cont de modul în care funcţionează afacerea modelată, datele care trebuie să fie memorate, relaţiile dintre acestea etc. Modul de utilizare a diferitelor date, modul în care acestea sunt relaţionate pot diferi de la o afacere la alta.

Regulile afacerii unei organizaţii se referă în esenţă la procesele şi fluxurile tuturor datelor şi activităţilor zilnice din cadrul organizaţiei. Cum funcţionează organizaţia? Care sunt activităţile sale?

Regulile afacerii acoperă următoarele aspecte ale unei organizaţii:

  Orice tip de politici organizaţionale de orice tip şi de la orice nivel al organizaţiei.

  Orice tip de formule de calcule (ca de exemplu modul de calcul al ratelor pentru diverse împrumuturi, modul de calcul al salariilor etc)

  Orice tip de reguli impuse de lege sau reguli interne ale organizaţiei.

Regulile simple ale afacerii pot fi implementate în modelul bazei de date prin intermediul relaţiilor dintre entităţi. Acest tip de reguli se numesc **reguli structurale**.

Alte reguli ale afacerii pot fi implementate folosind regulile de integritate despre care am discutat în paragraful anterior. Există totuşi reguli pentru implementarea cărora va trebui să scriem programe speciale folosind limbaje specializate specifice SGBD-ului utilizat. Acest tip de reguli se numesc numite **reguli procedurale**. În Oracle acest tip de programe se vor scrie folosind limbajul PL/SQL (Procedural Language/Structuded Query Languge) şi se numesc **declanşatoare** (**triggere**).

Există două tipuri de declanşatoare:

-          declanşatoare de aplicaţie care se execută când apar anumite evenimente la nivelul anumitor evenimente;

-          declanşatoare ale bazei de date care sunt lansate în execuţie când apar diverse evenimente asupra datelor (de exemplu la executarea unor comenzi ca **INSERT**, **UPDATE**, **DELETE**) sau la apariţia unor evenimente system (logarea la baza de date sau delogarea).

Orice declanşator poate avea rol de validare a unei operaţii, poate realiza diferite operaţii suplimentare, ca de exemplu diferite calcule, caz în care vom spune că e vorba de un declanşator de acţiune.

**Maparea tipurilor şi subtipurilor**

Nici un sistem de gestiune a bazelor de date nu suportă în mod direct supertipurile şi subtipurile. Putem adopta mai multe soluţii ale acestei probleme. Vom exemplifica aceste variante pentru schema din figura I.4.1, în care, pentru simplitate, vom presupune că nu avem nevoie de subentitatea **ALTUL**.

**Varianta 1**. Vom crea o tabelă pentru supertip şi câte o tabelă pentru fiecare subtip. Diagramele de tabelă în acest caz vor fi:

**Tabelul I.4.1.**Tabela **ANGAJAŢI                               Tabelul I.4.2.**Tabela **SECRETARE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Id\_angajat | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Adresa | Varchar2 |  | \* |
| Data\_nasterii | Date |  | \* |
| Id\_departament | Number | Fk | \* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Id\_angajat | Number | Pk | \* |

**Tabelul I.4.3.**Tabela **MANAGERI                   Tabelul I.4.4.**Tabela **REPREZENTANŢI\_VÂNZĂRI**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Id\_angajat | Number | Pk | \* |
| Bonus | Number |  | \* |
| Id\_depart\_condus | Number | Fk | o |

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Id\_angajat | Number | Pk | \* |
| Zona\_vanzari | Varchar2 |  | \* |
| Permis\_conducere | Varchar2 |  | \* |

Am notat cu **Id\_depart\_condus** codul departamentului pe care îl conduce un manager, iar cu **Id\_departament** codul departamentului în care lucrează un anumit angajat.

Cheia primară a supertipului va fi inclusă în toate tabelele corespunzătoare subtipurilor şi va deveni cheia primară a acelei tabele.

Atributele şi cheile străine provenite din relaţiile de la nivelul supertipului vor fi memorate în tabela corespunzătoare supertipului. Atributele şi relaţiile de la nivel de subtip, se vor memora doar în tabela corespunzătoare subtipului respectiv.

Acest model este cel mai natural dar poate crea multe probleme privind eficienţa întrucât sunt necesare multe operaţii de interogare din tabele multiple, pentru a obţine informaţii suplimentare despre toţi angajaţii.

**Varianta 2.** Vom crea câte o tabelă pentru fiecare subtip. Atributele şi cheile străine provenite din relaţiile de la nivelul supertipului vor fi introduse în fiecare tabelă astfel obţinută, acestea fiind moştenite de către fiecare subtip.

**Tabelul I.4.5.**Tabela **SECRETARE                                    Tabelul I.4.6.**Tabela **MANAGERI**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitate** |
| Id\_angajat | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Adresa | Varchar2 |  | \* |
| Id\_departament | Number | Fk | \* |
| Data\_nasterii | Date |  | \* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitate** |
| Id\_angajat | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Adresa | Varchar2 |  | \* |
| Data\_nasterii | Date |  | \* |
| Bonus | Number |  | \* |
| Id\_depart\_condus | Number | Fk | o |
| Id\_departament | Number | Fk | \* |

**Tabelul I.4.7.**Tabela **REPREZENTANŢI\_VÂNZĂRI**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitate** |
| Id\_angajat | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Adresa | Varchar2 |  | \* |
| Data\_nasterii | Date |  | \* |
| Id\_departament | Number | Fk | \* |
| Zona\_vanzari | Varchar2 |  | \* |
| Permis\_conducere | Varchar2 |  | \* |

**Varianta 3.** Vom crea o singură tabelă pentru supertip. Această tabelă va conţine toate coloanele corespunzătoare atributelor de la nivelul supertipului, dar şi toate coloanele corespunzătoare tuturor atributelor din toate subtipurile. Atributele de la nivelul supertipului îşi vor păstra opţionalitatea, însă atributele de la nivelul subtipurilor, vor fi toate introduse în tabelă, dar vor fi toate opţionale.

Relaţiile de la nivelul supertipului se transformă normal. Relaţiile de la nivelul subtipurilor se vor implementa cu ajutorul cheilor străine opţionale.

**Tabelul I.4.8.**Tabela **ANGAJAŢI**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Id\_angajat | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Adresa | Varchar2 |  | \* |
| Id\_departament | Number | Fk | \* |
| Data\_nasterii | Date |  | \* |
| Bonus | Number |  | o |
| Id\_depart\_condus | Number | Fk | o |
| Zona\_vanzari | Varchar2 |  | o |
| Permis\_conducere | Varchar2 |  | o |
| Tip\_angajat | Numeric |  | \* |

Am introdus un atribut suplimentar **Tip\_angajat**, cu ajutorul căruia vom codifica dacă un angajat este manager, secretară sau reprezentant de vânzări. Deoarece atributele de la nivelul subtipurilor sunt obligatorii pentru subtipul respectiv, va trebui să stabilim o regulă de integritate la nivel de înregistrare, care să verifice că pentru o înregistrare de un tip anume sunt completate câmpurile corespunzătoare. De exemplu, la adăugarea unui nou manager în tabela **ANGAJAŢI**, trebuie să verificăm dacă este completat câmpul bonus.

                Se observă că vor fi multe câmpuri cu valoarea null, ceea ce înseamnă o risipă de spaţiu de memorie.

**Tabelul I.4.9.**Tabela **ANGAJAŢI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Id\_angajat** | **Bonus** | **Id\_departament\_condus** | **Zona\_vanzari** | **Permis\_conducere** | **Tip\_angajat** | **…** |
| 10 | 125 | 5 | (null) | (null) | 1 |  |
| 121 | (null) | (null) | Transilvania | 568147 | 2 |  |
| 245 | (null) | (null) | (null) | (null) | 3 |  |
| … |  |  |  |  |  |  |

În acest tabel am codificat managerii cu 1, reprezentanţii de vânzări cu 2, iar secretarele cu 3. Aşadar această variantă de implementare este convenabilă când există puţine atribute şi relaţii la nivelul subtipurilor.

**. Maparea arcelor**

Pentru a mapa un arc vom crea atâtea chei străine câte relaţii există în arcul respectiv. Pentru modelul din figura I.4.2 vom obţine următoarele tabele:

**Tabelul I.4.10.**Tabela **CONTURI                         Tabelul I.4.11.**Tabela **PERSOANE\_FIZICE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| IBAN | Number | Pk | \* |
| Sold\_curent | Number |  | \* |
| Data\_deschiderii | Date |  | \* |
| Cnp | Number | Fk1 | o |
| Autorizatie\_functionare | Number | Fk2 | o |

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Cnp | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Prenume | Varchar2 |  | \* |
| Adresa | Varchar2 |  | \* |
| Telefon | Number |  | \* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Numele coloanei** | **Tip** | **Tip cheie** | **Opţionalitatea** |
| Autorizatie\_functionare | Number | Pk | \* |
| Nume | Varchar2 |  | \* |
| Adresa | Varchar2 |  | \* |
| Telefon | Number |  | \* |
| Fond\_social | Number |  | \* |

**Tabelul I.4.12.**Tabela **FIRME**Deşi relaţiile din arc sunt obligatorii, cheile străine corespunzătoare au fost setate ca fiind opţionale, deoarece pentru fiecare înregistrare trebuie să avem completată una din cele două chei străine, iar cealaltă cheie străină trebuie să rămână necompletată (principiul exclusivităţii). Va trebui să implementăm o condiţie de integritate care să verifice această condiţie.