

Ministerul Educației, Cercetării și Tineretului

Carmen Popescu

# Manual de **INFORMATICĂ** PENTRU CLASA A XII-A

(filiera *teoretică*, profil *real*, specializarea: *matematică-informatică*)  
și (filiera *vocațională*, profil *militar MApN*, specializarea *matematică-informatică*)

Aprobat prin ordinul MEdCT nr. 1561/81 din 23.07.2007

**L&S Info-mat**

**Copyright 2007-2016 © L&S INFO-MAT**

**Toate drepturile asupra acestei lucrări aparțin editurii L&S INFO-MAT.  
Reproducerea integrală sau parțială a textului din această carte este posibilă  
doar cu acordul în scris al editurii L&S INFO-MAT.**

Manualul a fost aprobat prin Ordinul Ministrului Educației, Cercetării și Tineretului nr. 1561 / 81 din 23.07.2007 în urma evaluării calitative și este realizat în conformitate cu programa analitică aprobată prin Ordin al ministrului Educației și Cercetării nr. 5959 din 22.12.2006.

Referenți științifici:

Prof. Gradul I, **Maria Canter**, Sibiu  
Prof. Gradul I, **Anca Voineag**, Sibiu

**Tiparul executat la S.C. LuminaTipo S.R.L.**  
Str. Luigi Galvani nr. 20 bis, Sector 2, București,  
office@luminatipo.com

**Editura L&S INFO-MAT:**



**Adresa:** Str. Stânjeneilor nr. 6, bl. 30, sc. A, et. 1, apt. 11, sector 4, București;  
**Mobil:** 0722-573701; 0749.99.77.07;  
**E-mail:** comenzi@ls-infomat.ro;  
**Web Site:** www.ls-infomat.ro.

# Cuprins

## PARTEA I: Proiectarea bazelor de date

<b>I.1. Proiectarea bazelor de date. Noțiuni introductive.....</b>	<b>11</b>
1. Date, informații, cunoștințe .....	12
2. Colectarea și analizarea datelor. Modelul conceptual .....	13
3. Entități. Instanțe. Atribute. Identificator unic.....	14
Aplicații .....	16
4. Relații între entități .....	17
Convenții de reprezentare a relațiilor.....	18
Tipuri de relații .....	19
Relații ierarhice. Relații recursive .....	21
Relații redundante.....	23
5. Rezolvarea relațiilor many-to-many .....	24
Test de autoevaluare .....	27
Test de evaluare 1.....	29
Test de evaluare 2.....	30
Aplicații .....	31
<b>I.2. Normalizarea datelor.....</b>	<b>33</b>
1. Ce este normalizarea? .....	34
2. Prima formă normală.....	35
3. A doua formă normală .....	37
4. A treia formă normală .....	38
5. Exemplu de normalizare .....	39
Aplicații .....	42
<b>I.3. Implementarea modelului conceptual .....</b>	<b>45</b>
1. Modele de baze de date .....	46
2. Baze de date relaționale.....	47
Aplicații .....	49

<b>3. Maparea relațiilor .....</b>	<b>50</b>
Maparea relațiilor one-to-many.....	50
Maparea relațiilor one-to-one.....	51
Maparea relațiilor recursive .....	52
<b>4. Maparea relațiilor barate .....</b>	<b>53</b>
<b>5. Exemplu complet de mapare .....</b>	<b>54</b>
<b>Aplicații .....</b>	<b>56</b>
<b>6. Operații specifice prelucrării bazelor de date.....</b>	<b>56</b>
<b>7. Reguli de integritate .....</b>	<b>57</b>
<b>8. Programe de validare și de acțiune .....</b>	<b>58</b>
<b>Test de autoevaluare .....</b>	<b>59</b>
<b>I.4. Elemente avansate de proiectare a bazelor de date.....</b>	<b>61</b>
1. Tipuri și subtipuri .....	62
2. Maparea tipurilor și a subtipurilor .....	63
Aplicații.....	66
3. Relații exclusive (arce) .....	66
4. Maparea arcelor .....	68
Aplicații.....	69
5. Nontransferabilitate.....	69
6. Modelarea datelor istorice .....	70
Aplicații.....	75
<b>I.5. Dezvoltarea profesională în domeniul IT.....</b>	<b>77</b>
1. Evaluarea aptitudinilor și a intereselor .....	78
2. Identificarea meseriilor de interes .....	83
3. Evaluarea posibilelor cariere.....	85
4. Scrisoarea de intenție .....	87
5. Scrierea curriculumului vitae .....	89
6. Pregătirea și susținerea interviului.....	91
Exemple de întrebări frecvente în interviurile la angajare .....	93

<b>I.6. Managementul de proiect.....</b>	<b>95</b>
1. Ce este un proiect ? .....	96
2. Etape în realizarea unui proiect .....	96
3. Principiile lucrului în echipă.....	98
4. Pregătirea și susținerea unei prezentări .....	99
Teme de proiect .....	101

## **PARTEA II: Programarea bazelor de date**

<b>II.1. Interogări simple. Sortarea datelor.....</b>	<b>107</b>
1. Noțiuni introductive.....	108
2. Elemente de bază ale SQL .....	113
3. Interogarea tabelor. Comanda <b>SELECT</b> .....	115
Aliasul unei coloane.....	118
Eliminarea liniilor duplicate .....	120
Filtrarea liniilor. Clauza <b>WHERE</b> .....	121
4. Sortarea datelor. Clauza <b>ORDER BY</b> .....	123
5. Afișarea primelor <i>n</i> linii.....	127
Aplicații.....	130
Joc.....	131
<b>II.2. Funcții singulare .....</b>	<b>134</b>
1. Tipuri de funcții.....	135
2. Tabela <b>DUAL</b> .....	135
3. Funcții asupra șirurilor de caractere .....	136
Combinarea funcțiilor asupra șirurilor de caractere.....	140
4. Funcții numerice .....	141
5. Funcții asupra datelor calendaristice .....	145
Aritmetica datelor calendaristice.....	146
Funcții cu date calendaristice .....	147
6. Funcții de conversie.....	150
Transformarea din dată calendaristică în șir de caractere .....	150
Transformarea din șir de caractere în dată calendaristică .....	153
Formatul <b>RR</b> și formatul <b>YY</b> .....	153

Transformarea din număr în șir de caractere .....	155
Transformarea din șir de caractere în număr .....	156
<b>7. Funcții de uz general .....</b>	<b>156</b>
<b>8. Funcții și expresii condiționale .....</b>	<b>158</b>
<b>Aplicații .....</b>	<b>159</b>
<b>II.3. Interogări multiple .....</b>	<b>161</b>
1. Produsul cartezian .....	163
2. Equijoin .....	165
3. Nonequijoin .....	167
4. Self Join .....	168
5. OuterJoin .....	169
6. Operatorii UNION, INTERSECT, MINUS .....	175
Test de evaluare .....	178
Aplicații .....	182
<b>II.4. Gruparea datelor .....</b>	<b>185</b>
1. Studiu de caz .....	186
2. Funcții de grup .....	187
3. Gruparea datelor. Clauza GROUP BY .....	192
Reguli de folosire a clauzei GROUP BY .....	194
4. Selectarea grupurilor. Clauza HAVING .....	195
Aplicații .....	200
Jocuri .....	202
<b>II.5. Subinterogări .....</b>	<b>205</b>
1. Subinterogări simple .....	207
2. Subinterogări multiple .....	209
Subinterogări multiple cu operatorul IN .....	210
Subinterogări multiple cu ALL .....	212
Subinterogări multiple cu ANY .....	213
Subinterogări multiple cu EXISTS .....	216
Subinterogări multiple în clauza FROM .....	216
Test de autoevaluare .....	217
Aplicații .....	219

## **II.6. Crearea și modificarea structurii tabelelor. Constrângeri. 222**

<b>1. Crearea tabelelor .....</b>	<b>223</b>
Definirea valorilor implicite pentru coloane .....	224
<b>2. Definirea constrângerilor .....</b>	<b>225</b>
Restricția NOT NULL.....	226
Restricțiile PRIMARY KEY și UNIQUE.....	227
Restricția FOREIGN KEY .....	229
Restricția CHECK .....	233
<b>3. Modificarea structurii unei table .....</b>	<b>234</b>
Adăugarea unei noi coloane .....	234
Ștergerea unei coloane.....	245
Modificarea unei coloane.....	236
Adăugarea unei constrângeri.....	236
Ștergerea unei constrângeri .....	237
Activarea/dezactivarea unei constrângeri.....	237
<b>Test de autoevaluare .....</b>	<b>238</b>
<b>Aplicații.....</b>	<b>241</b>

## **II.7. Introducerea și actualizarea datelor din table .....**

<b>1. Adăugarea datelor în table .....</b>	<b>243</b>
<b>2. Ștergerea datelor dintr-o tabelă .....</b>	<b>247</b>
<b>3. Modificarea datelor dintr-o tabelă.....</b>	<b>248</b>
<b>Aplicații.....</b>	<b>250</b>
<b>Aplicații recapitulative .....</b>	<b>251</b>

## **II.8. Vederi (views).....**

<b>1. Crearea și ștergerea vederilor.....</b>	<b>255</b>
<b>2. Actualizarea datelor prin intermediul vederilor .....</b>	<b>256</b>
Inserarea datelor prin intermediul vederilor .....	258
Ștergerea datelor prin intermediul vederilor .....	259
Modificarea datelor prin intermediul vederilor.....	260
Restricții privind utilizarea vederilor .....	261
<b>Aplicații.....</b>	<b>262</b>

<b>II.9. Secvențe. Indecși. Sinonime.....</b>	<b>263</b>
<b>1. Secvențe .....</b>	<b>264</b>
Crearea și ștergerea secvențelor.....	264
Utilizarea secvențelor .....	266
Modificarea secvențelor .....	267
<b>2. Indecși .....</b>	<b>268</b>
<b>3. Sinonime.....</b>	<b>269</b>
Test de autoevaluare .....	270
 <b>II.10. Alocarea și revocarea drepturilor.</b>	
<b>Gestiunea tranzacțiilor .....</b>	<b>273</b>
<b>1. Drepturi și roluri .....</b>	<b>274</b>
<b>02. Drepturile de sistem .....</b>	<b>275</b>
Acordarea drepturilor de sistem.....	276
<b>3. Drepturile la nivel de obiect.....</b>	<b>277</b>
Acordarea drepturilor la nivel de obiect .....	277
<b>4. Gestiunea rolurilor .....</b>	<b>278</b>
<b>5. Gestiunea tranzacțiilor.....</b>	<b>280</b>
Aplicație .....	285
 <b>II.11. Realizarea proiectelor .....</b>	<b>286</b>
1. Crearea tabelelor bazei de date.....	287
2. Crearea aplicației și a paginii principale .....	290
3. Adăugarea câmpurilor calculate unui formular sau raport .....	293
4. Crearea listelor de valori.....	296
Aplicații.....	301
 <b>II.12. Aplicații recapitulative .....</b>	<b>302</b>
 <b>Bareme de corectare și notare.....</b>	<b>311</b>



# **I. Proiectarea bazelor de date**





# Proiectarea bazelor de date.

## Noțiuni introductive

I.1

### 1. Proiectarea bazelor de date. Noțiuni introductive

2. Normalizarea datelor
3. Implementarea modelului conceptual
4. Elemente avansate de proiectare a bazelor de date
5. Dezvoltarea profesională în domeniul IT
6. Managementul de proiect

În acest capitol veți afla:

- ✓ ce este modelul conceptual și care este rolul său
- ✓ ce este un ERD
- ✓ ce este o entitate și cum se reprezintă ea într-un ERD
- ✓ ce este o instanță
- ✓ ce sunt și cum se stabilesc attributele unei entități
- ✓ care sunt tipurile de attribute și cum se reprezintă ele în ERD
- ✓ cum se stabilesc relațiile între entități
- ✓ care sunt caracteristicile unei relații
- ✓ cum se citește o relație
- ✓ ce tipuri de relații pot exista între entități și cum se reprezintă ele în ERD
- ✓ cum se rezolvă relațiile many-to-many

### I.1.1. Date. Informații. Cunoștințe

Auzim adesea vorbindu-se despre “Era informațiilor” sau “societate informațională” sau “tehnologia informației” însă de multe ori cuvântul “informație” este folosit fără a-i înțelege clar sensul, diferența dintre date, informații, cunoștințe.

În general, conținutul gândirii umane operează cu următoarele concepte:

- **Date** – constau în material brut, fapte, simboluri, numere, cuvinte, poze fără un înțeles de sine stătător, neintegrate într-un context, fără relații cu alte date sau obiecte. Ele se pot obține în urma unor experimente, sondaje, etc.
- **Informații** – prin prelucrarea datelor și găsirea relațiilor dintre acestea se obțin informații care au un înțeles și sunt integrate într-un context. Datele organizate și prezentate într-un mod sistematic pentru a sublinia sensul acestora devin informații. Pe scurt informațiile sunt date prelucrate. Informațiile se prezintă sub formă de rapoarte, statistici, diagrame, etc.
- **Cunoștințele** sunt colecții de date, informații, adevăruri și principii învățate, acumulate de-a lungul timpului. Informațiile despre un subiect reținute, înțelese și care pot fi folosite în luarea de decizii, formează judecăți și opinii, devin cunoștințe. Cu alte cuvinte, cunoștințele apar în momentul utilizării informației.
- **Înțelepciunea** este un nivel superior de înțelegere a faptelor și informațiilor. Vorbim despre înțelepciune atunci când pe baza informațiilor și cunoștințelor pe care le deținem putem discerne între bine și rău, formulăm opinii, păreri personale etc. Înțelepciunea este o caracteristică a oamenilor, calculatoarele neputând opera decât cu primele trei concepte.

Pentru a clarifica aceste concepte, să luăm un exemplu.

**Date:**

"42"    "iepuri"    "4.00pm"  
"76"    "mere"    "0740112233"  
"20euro"    "mare"

**Informații:**

Sunt **42 mere** în această cutie și fiecare măr este ronțait de către **iepuri**.

Costul biletului până la **mare** este de **20euro** și călătoria durează **76** minute cu trenul.

Numărul meu de telefon este **0740112233**. Sună-mă la ora **4.00pm**!

Aceste informații adaugă un context și un sens datelor.

**Cunoștințe:**

În ultimii cinci ani, recolta de mere din Moldova a crescut cu 10% în fiecare an. Se prevede că și în acest an recolta va crește cu încă 10% și de aceea trebuie să găsim o piață de desfacere pentru 10% mai multe mere.

Informațiile din ultimii câțiva ani au fost folosite pentru a estima creșterea producției de mere și necesitatea unei piețe mai mari de desfacere. Predicția făcută este *cunoștință*, cu alte cuvinte, folosirea informațiilor deținute.

## I.1.2. Colectarea și analizarea datelor. Modelul conceptual

Primul pas în realizarea unei aplicații de baze de date este analiza datelor și realizarea unei scheme conceptuale (model conceptual) a acestora.

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

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

Informațiile necesare realizării modelului conceptual se obțin folosind metode convenționale precum interviuarea oamenilor din cadrul organizației și studierea documentelor folosite.

Odată obținute aceste informații ele trebuie reprezentate într-o formă convențională care să poată fi ușor înțeleasă de toată lumea. O astfel de reprezentare este **diagrama entități-relații**, numită și **harta relațiilor**, sau **ERD-ul** (**Entity Relationship Diagram**). Aceste scheme sunt un instrument util care ușurează comunicarea dintre specialiștii care proiectează bazele de date și programatori pe de o parte și beneficiari, pe de altă parte. Aceștia din urmă pot înțelege cu ușurință o astfel de schemă, chiar dacă nu sunt cunoscători în domeniul IT.

În concluzie, putem sublinia câteva caracteristici ale ERD-urilor:

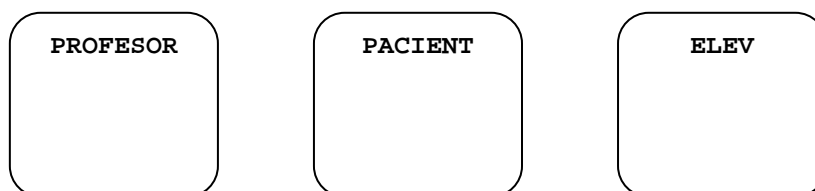
- sunt un instrument de proiectare;
- sunt o reprezentare grafică a unui sistem de date;
- oferă un model conceptual de înalt nivel al bazelor de date;
- sprijină înțelegerea de către utilizatori a datelor și a relațiilor dintre acestea
- sunt independente de implementare.

În cele ce urmează vom prezenta principalele elemente care intră în componența unui ERD precum și convențiile de reprezentare a acestora.

### I.1.3. Entități. Instanțe. Atribute. Identificator unic

O **entitate** este un lucru, obiect, persoană sau eveniment care are semnificație pentru afacerea modelată, despre care trebuie să colectăm și 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 operație, sau poate fi o noțiune abstractă.

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



**Figura I.1.1.** Exemple de entități și modul de reprezentare

O entitate este de fapt o clasă de obiecte și pentru orice entitate există mai multe **instanțe** ale sale. O instanță a unei entități 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 instanță a entității **ELEV**.

După cum se vede, pentru a preciza o instanță a unei entități, trebuie să specificăm unele caracteristici ale acestui obiect, să-l descriem (precizăm de exemplu numele, clasa, școala, etc.). Așadar, după ce am identificat entitățile

trebuie să descriem aceste entități în termeni reali, adică să le stabilim **atributele**. Un atribut este orice detaliu care servește la identificarea, clasificarea, cuantificarea, sau exprimarea stării unei instanțe a unei entități. Atributele sunt informații specifice ce trebuie cunoscute și memorate.

De exemplu, atributele entității **ELEV** sunt nume, prenume, adresa, număr de telefon, adresa de e-mail, data nașterii, etc.

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

Un atribut poate fi **obligatoriu** sau **opțional**. Dacă un atribut este obligatoriu, pentru fiecare instanță a entității respective trebuie să avem o valoare pentru acel atribut, de exemplu, este obligatoriu să cunoaștem numele elevilor. Pentru un atribut opțional putem avea instanțe pentru care nu cunoaștem valoarea atributului respectiv. De exemplu, atributul **email** al entității **ELEV** este opțional, un elev putând să nu aibă adresă de e-mail.

Un atribut obligatoriu este precedat în ERD de un asterisc \*, iar un atribut opțional va fi precedat de un cerculeț o.



**Figura I.1.2.** Entitatea **ELEV**

Atributele care definesc în mod unic instanțele unei entități se numesc **identificatori unici (UID)**. UID-ul unei entități poate fi compus dintr-un singur atribut, precum codul numeric personal ce poate fi un identificator unic pentru entitatea **ELEV**. În alte situații, identificatorul unic este compus dintr-o combinație de două sau mai multe atribute.

De exemplu combinația dintre titlu, numele autorului și data apariției poate forma unicul identificator al entității **CARTE**. Oare combinația titlu și 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 entități vor fi precedate de semnul diez # (figura I.1.2 și I.1.3). **Atributele din UID sunt întotdeauna obligatorii**, însă semnul # este suficient, nu mai trebuie pus și un semn asterisc în fața acestor atribute.

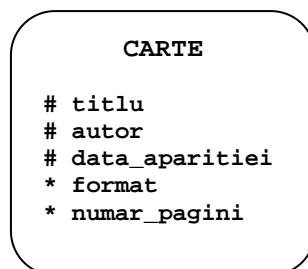


Figura I.1.3. Entitatea CARTE

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 nașterii”) acesta este un atribut **non-volatil**. Evident, este de preferat să folosim atribute non-volatile atunci când acest lucru este posibil.



## Aplicații

Identificați entitățile pentru următoarele scenarii. Identificați apoi pentru fiecare entitate atributele sale, stabiliți opționalitatea acestora și precizați unicul identificator al fiecărei entități.

*Indicație.* Subliniați substantivele care au semnificație pentru afacerea descrisă. Un substantiv va fi subliniat doar la prima sa apariție. Dintre aceste substantive veți alege apoi entitățile.

1. Pentru a se abona la diverse reviste, persoanele doritoare trebuie să furnizeze numele, adresa și un număr de telefon. Fiecare revistă este identificată prin titlul, numărul volumului și data apariției. Abonații semnează pentru abonare un contract pe o anumită perioadă de timp specificată prin data de început a abonamentului și data finală. Bineînțeles că o persoană se poate abona la mai multe reviste în același timp.

2. La o firmă de calculatoare există mai multe departamente, fiecare fiind identificat printr-un cod. Pentru fiecare departament se cunoaște numele departamentului precum și managerul acestuia. Fiecare departament dispune de mai multe birouri, situate în una din clădirile firmei. Un birou poate fi dotat cu mai multe imprimante, dar este posibil ca în anumite birouri să nu existe nici o imprimantă. Firma ține o evidență a firmelor care oferă tonere și cartușe pentru fiecare tip de imprimantă în parte. Se știe pentru fiecare astfel de firmă la ce preț furnizează fiecare tip de toner sau cartuș. E posibil ca o firmă să nu dispună de tonere sau cartușe pentru toate tipurile de imprimante pe care le deține firma. Pentru fiecare firmă se cunoaște perioada de onorare a unei comenzi.



3. Despre angajații unei firme se cunoaște numele, titlul, numărul de telefon de la birou. Angajații pot fi implicați într-o serie de proiecte ce se desfășoară în cadrul firmei. Despre fiecare proiect se cunoaște numele, data la care a demarat proiectul și se poate cunoaște o dată la care se va finaliza proiectul. La fiecare proiect lucrează un singur angajat, însă un angajat poate fi implicat în mai multe proiecte. Fiecare angajat are un manager, cu excepția directorului. Managerii pot fi și ei implicați în proiecte.

4. O companie de teatru dorește să memoreze într-o bază de date informații despre spectacolele pe care le susține și despre actorii săi. Un spectacol are loc într-o anumită zi și la o anumită oră și la acel spectacol se joacă o anumită piesă de teatru. O piesă de teatru nu este întotdeauna jucată de aceiași actori. Un actor poate juca în mai multe piese de teatru. Un actor poate avea diferite abilități. El știe de exemplu să cânte sau să danseze. Aceste abilități trebuie să fie memorate în baza de date. O piesă are un coordonator care trebuie să se asigure că piesa este jucată profesional. Acest coordonator trebuie să fie și el actor.

## I.1.4. Relații între entități

În lumea reală, obiectele nu există izolat. Percepem obiectele din lumea reală doar în conexiune cu alte obiecte, de exemplu vom spune 'pământul se învâрте în jurul soarelui', 'el este medic', etc.

Așadar, după ce ați identificat care sunt entitățile și atributele acestor entități este timpul să punem în evidență relațiile care există între aceste entități, modul în care acestea comunică între ele. O **relație** este o asociere, legătură, sau conexiune existentă între entități și care are o semnificație pentru afacerea modelată.

Orice relație este bidirecțională, legând două entități sau o entitate cu ea însăși. De exemplu, elevii studiază mai multe materii, o materie e studiată de către elevi.

Orice relație este caracterizată de următoarele elemente:

- numele relației
- opționalitatea relației
- gradul (cardinalitatea) relației.

Să luăm ca exemplu relația existentă între entitățile **JUCĂTOR** și **ECHIPĂ**. Vom spune:

Un **JUCĂTOR** joacă într-o **ECHIPĂ**.

- **Numele relației** este: *joacă*.

- Pentru a stabili **opționalitatea** relației trebuie să răspundem la următoarele întrebare: Un jucător **trebuie** să joace într-o echipă? Se poate ca un jucător să nu joace în nicio echipă?

Dacă acceptăm că toți jucătorii trebuie să joace într-o echipă relația este obligatorie sau mandatorie și vom spune:

Un JUCĂTOR **trebuie** să joace într-o ECHIPĂ.

Dacă însă acceptăm că există jucători care nu joacă în nicio echipă (de exemplu li s-a terminat contractul și în momentul de față nu mai joacă la nicio echipă), atunci relația este opțională. În acest caz vom spune:

Un JUCĂTOR **poate** juca la o ECHIPĂ.

- **Cardinalitatea** relației este dată de numărul de instanțe ale entității din partea dreaptă a relației care pot intra în relație cu o instanță a entității din partea stângă a relației. Adică va trebui să răspundem la întrebări de genul: La câte echipe poate juca un jucător? Răspunsurile posibile sunt **unul și numai unul**, sau **unul sau mai mulți**. Vom spune:

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

sau

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

Cea mai realistă variantă a relației dintre JUCĂTOR și ECHIPĂ este așadar:

Un JUCĂTOR poate să joace la o ECHIPĂ și numai una.

Am precizat însă mai înainte că orice relație este bidirecțională. Relația dintre ECHIPĂ și JUCĂTOR o putem enunța astfel:

La o ECHIPĂ trebuie să joace unul sau mai mulți JUCĂTORI.

## Convenții de reprezentare a relațiilor

În cadrul diagramei entități-relații, o relație va fi reprezentată printr-o linie ce unește cele două entități.

Deoarece o relație este bidirecțională, linia ce unește cele două entități este compusă din două segmente distincte, câte unul pentru fiecare entitate. Tipul segmentului ce pleacă de la o entitate ne va indica **opționalitatea** relației dintre această entitate și entitatea aflată în cealaltă parte a relației. Dacă acest segment este continuu este vorba de o relație obligatorie, o linie întreruptă indică o relație opțională.

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

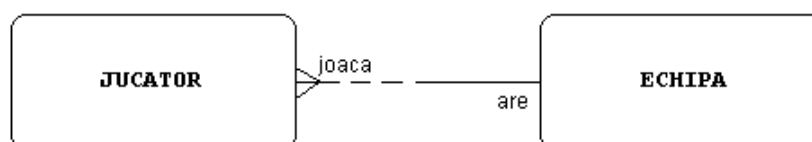


Figura I.1.4. Reprezentarea relațiilor

Modul în care o linie se termină spre o entitate este important. Dacă se termină printr-o linie simplă, înseamnă că o instanță și numai una a acestei entități este în relație cu o instanță a celeilalte entități. În exemplul anterior, linia de la JUCĂTOR la ECHIPĂ se termină în partea dinspre ECHIPĂ cu o **linie simplă**, deci un jucător joacă la **o** echipă **și numai una**.

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

Caracteristica relației	Valoare	Mod de reprezentare
<b>Numele relației</b>	un verb	se scrie deasupra relației
<b>Opționalitatea</b>	relație obligatorie (TREBUIE)	linie continuă ————
	relație opțională (POATE)	linie întreruptă -----
<b>Cardinalitatea</b>	una și numai una	linie simplă ———
	una sau mai multe	picior de cioară <=

## Tipuri de relații

Variantele de relații ce pot exista între două entități sunt prezentate mai jos:

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

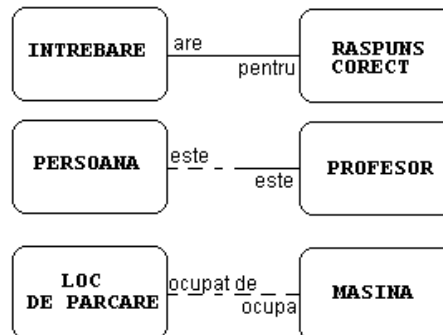


Figura I.1.5. Relații one-to-one

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

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

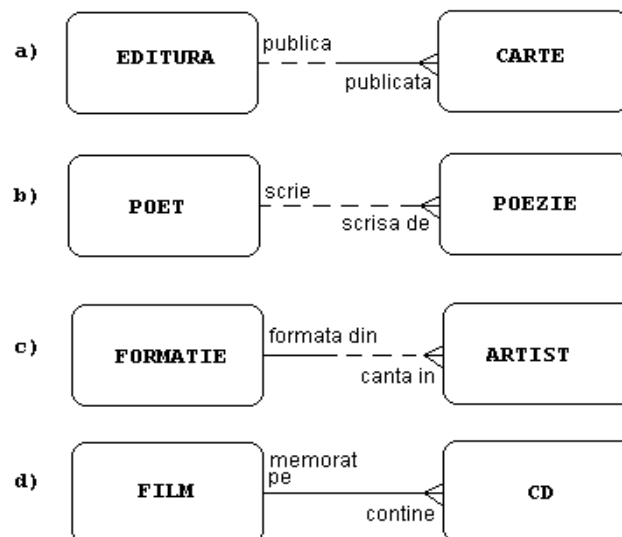


Figura I.1.6. Relații one-to-many

- **relații many-to-many** – aceste tipuri de relații apar în prima fază a proiectării bazei de date, însă ele trebuie să fie ulterior eliminate. Figura I.1.7 prezintă câteva exemple de relații many-to-many. La punctul **b** am considerat că un curs poate apărea pe oferta de cursuri a unei facultăți, însă poate să nu fie aleasă de niciun student de aceea un curs **poate** fi urmat de unul sau mai mulți studenți. Invers, este posibil ca un student să fi terminat studiile și să se pregătească pentru susținerea examenului de licență și de aceea el nu mai frecventează nici un curs. La punctul **c**, un profesor angajat al unei școli **trebuie** să predea cel puțin o disciplină. Iar o disciplină din planul de învățământ trebuie să fie predată de cel puțin un profesor.

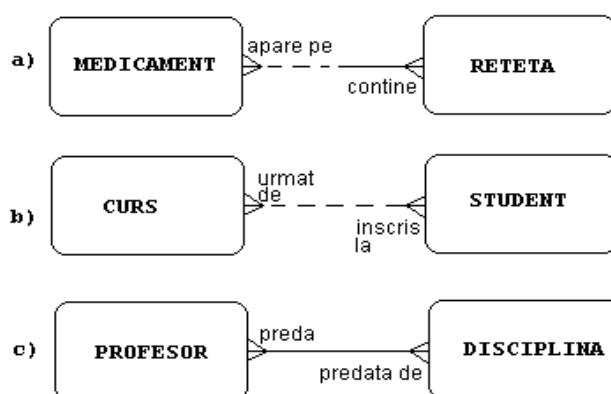


Figura I.1.7. Relații many-to-many

## Relații ierarhice. Relații recursive

Haideți să analizăm care este structura personalului într-o firmă oarecare. În figura I.1.8 este prezentată doar o parte din organigrama acesteia:

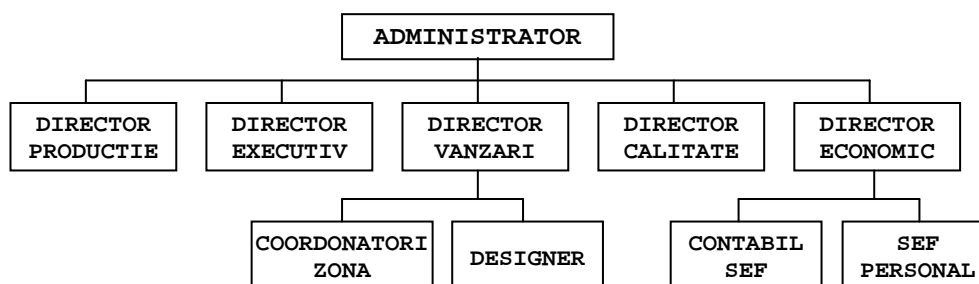
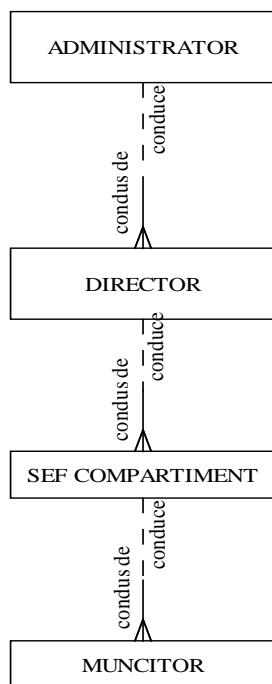


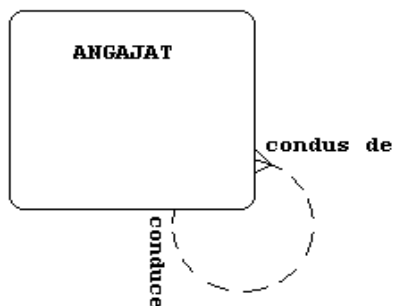
Figura I.1.8. Organigrama unei firme

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 și probabil există foarte multe atribute comune tuturor acestor entități ca de exemplu nume, prenume, adresă, telefon, e-mail, data nașterii, etc. Vom putea de aceea modela această structură cu ajutorul unei singure entități numită **ANGAJAT**. Însă fiecare angajat poate fi condus de către un alt angajat. Așadar vom avea o relație de la entitatea **ANGAJAT** la ea însăși. O astfel de relație se numește **relație recursivă**.



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

## Relații redundante

Atunci când o relație poate fi dedusă din alte relații, spunem că acea relație este redundantă. Să considerăm exemplul din figura I.1.11.

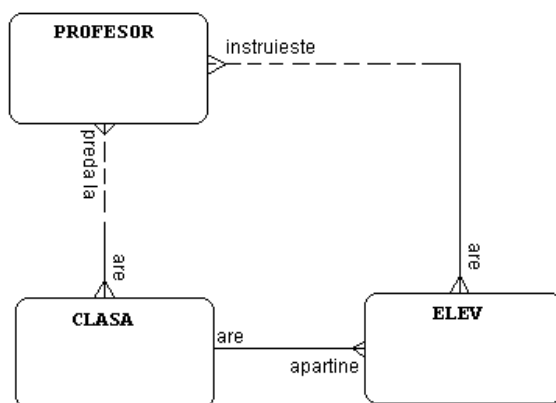


Figura I.1.11. Relații redundante

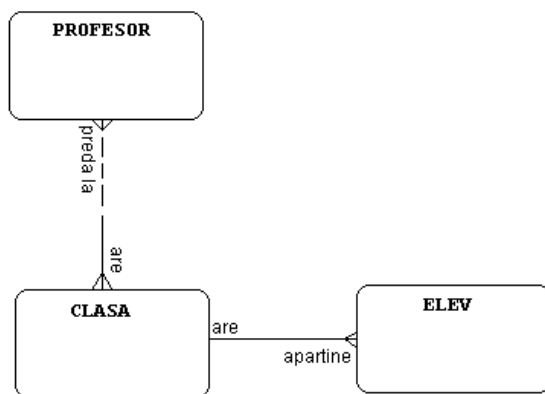


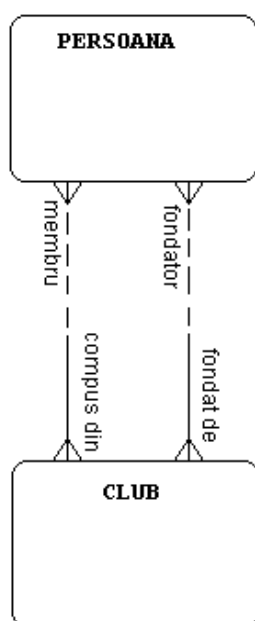
Figura I.1.12. Eliminarea relației redundante

Se observă că un elev face parte dintr-o clasă, iar la acea clasă predau mai mulți profesori. Așadar relația dintre profesor și elev nu mai este necesară deoarece putem deduce profesorii care îi predau unui elev, aflând profesorii clasei din care face parte elevul. Această relație poate fi deci eliminată, ca în figura I.1.12.

Trebuie totuși acordată mare atenție acestui tip de relații. Pentru situația anterioară, se pune întrebarea dacă un elev nu poate avea un profesor care nu predă la clasa în care învață? Desigur acest lucru depinde de situația pe care o modelăm. Dacă de exemplu ne propunem să memorăm date despre toți profesorii care îl instruiesc pe un elev, în cadrul orelor de curs, dar ne interesează de asemenea

profesorii care îndrumă activitățile extracurriculare ale elevilor, atunci e posibil ca un elev să aibă și alți profesori decât cei de la clasă. În astfel de situații vom păstra totuși relația dintre profesor și elev, adică vom opta pentru schema din figura I.1.11.

Atenție și la situația în care două entități pot fi legate prin mai multe relații diferite. Acestea nu sunt neapărat redundante. De exemplu, în figura I.1.13 sunt prezentate două relații diferite între entitățile **PERSOANA** și **CLUB**. O persoană poate fi membră a mai multor cluburi și poate fi fondatorul unor cluburi. Faptul că o persoană a fondat un anumit club nu înseamnă obligatoriu că este membru al acelui club. De asemenea faptul că o persoană este membru al unui club nu înseamnă că este fondatorul lui. Așadar cele două relații nu se implică una pe cealaltă.



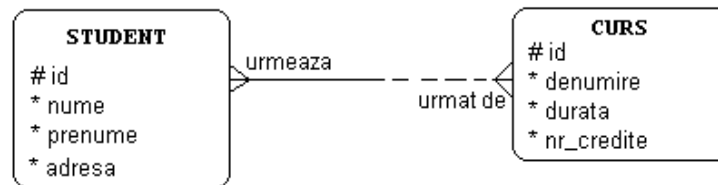
**Figura I.1.13.** Relații multiple între entități

## I.1.5. Rezolvarea relațiilor many-to-many

După cum am precizat mai devreme relațiile 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 relația din figura I.1.14 dintre entitățile **STUDENT** și **CURS**. Se știe că orice curs se termină în general cu un examen. Unde vom memora nota studentului la fiecare examen?





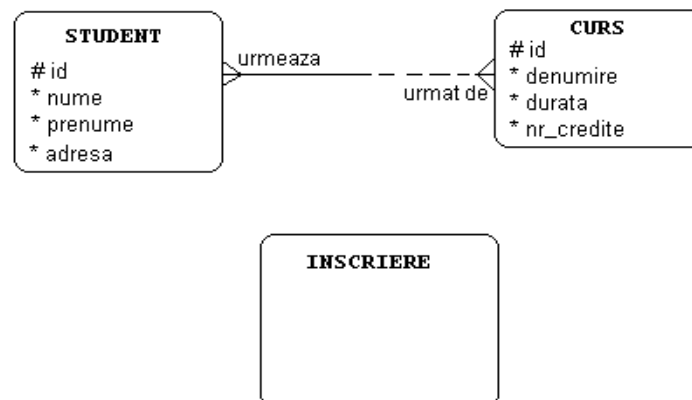
**Figura I.1.14.** Exemplu de relație

Dacă încercăm să introducem atributul **NOTA** la entitatea **STUDENT**, nu vom ști cărei materii corespunde acea notă, întrucât unei instanțe a entității **student** îi corespund mai multe instanțe ale entității **CURS**. Invers, dacă încercăm să memorăm nota în cadrul entității **CURS**, nu vom ști cărui student îi aparține acea notă.

Rezolvarea unei relații many-to-many constă în introducerea unei noi entități numită **entitate de intersecție**, pe care o legăm de entitățile originale prin câte o relație one-to-many.

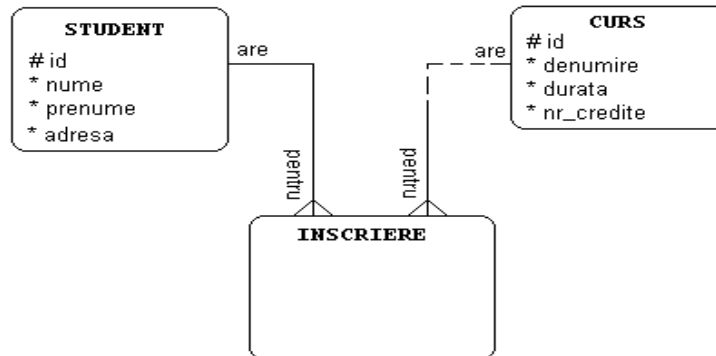
Pașii în rezolvarea unei relații many-to-many sunt următorii:

- se găsește entitatea de intersecție, pentru exemplul nostru vom introduce entitatea **INSCRIERE**.



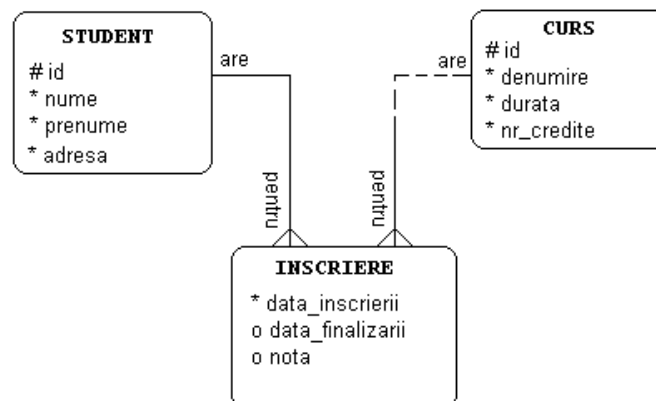
**Figura I.1.15.** Rezolvarea relațiilor many-to-many, pasul 1

- crearea noilor relații
  - opționalitatea: relațiile care pleacă din entitatea de intersecție sunt întotdeauna obligatorii în această parte. În partea dinspre entitățile originale, relațiile vor păstra opționalitatea relațiilor inițiale.
  - cardinalitatea: ambele relații sunt de tip one-to-many, iar partea cu many va fi întotdeauna înspre entitatea de intersecție.
  - numele noilor relații.



**Figura I.1.16.** Rezolvarea relațiilor many-to-many, pasul 2

- adăugarea de atribute în cadrul entității de intersecție, dacă acestea există. În exemplul nostru ne poate interesa să zicem data la care s-a înscris un student la un curs, data la care a finalizat cursul precum și nota obținută la sfârșitul cursului.



**Figura I.1.17.** Rezolvarea relațiilor many-to-many, pasul 3

- stabilirea identificatorului unic pentru entitatea de intersecție: dacă entitatea de intersecție nu are un identificator unic propriu, atunci acesta se poate forma din identificatorii unici ai entităților inițiale la care putem adăuga atribute ale entității de intersecție.

În exemplul nostru, identificatorul unic al entității de intersecție este format din id-ul studentului, id-ul cursului și data înscrierii la curs.

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

Vom vedea mai târziu că uneori nu putem bara ambele relații dinspre entitatea de intersecție.

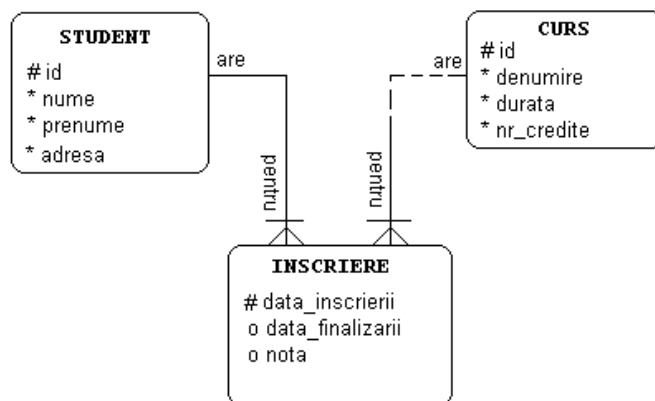


Figura I.1.18. Rezolvarea relațiilor many-to-many, pasul 4



## Test de autoevaluare

1. O bază de date va memora orarul unei universități. Fiecare curs este parte a unui modul, fiecărui curs îi este asociat exact un profesor. La fiecare curs participă mai mulți studenți.

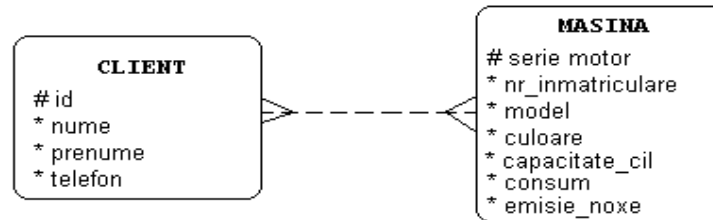
Fiecare poziție din orar corespunde unei zile a săptămânii și unei anumite ore. Fiecare poziție din orar durează exact o oră. Un curs poate dura mai multe ore consecutive, însă nici un curs nu poate apărea în zile diferite, sau la ore diferite neconsecutive ale aceleiași zile.

Fiecare profesor și fiecare student pot avea mai multe ore de curs la care participă în decursul unei săptămâni.

Care dintre următoarele variante **NU** este o soluție posibilă a acestei probleme?

- Se stabilește o relație one-to-many între **CURS** și **POZITIE\_ORAR**.
- Se stabilește o relație many-to-many între **CURS** și **POZITIE\_ORAR**.
- Pentru fiecare curs vom avea un atribut *start* care reține ora de începere a cursului și un atribut *durată* care memorează numărul de poziții consecutive din orar "ocupate" de acel curs.
- Pentru fiecare curs vom avea două atribute *primul* și *ultimul* care memorează prima și respectiv ultima poziție din orar ocupată de acel curs.

2. Fie următoarea hartă a relațiilor:



Cum se citește corect relația dintre **CLIENT** și **MAȘINĂ**?

- a) Fiecare **CLIENT** poate să închirieze o **MAȘINĂ** și numai una
- b) Fiecare **CLIENT** trebuie să închirieze o **MAȘINĂ** și numai una.
- c) Fiecare **CLIENT** poate să închirieze una sau mai multe **MAȘINI**.
- d) Fiecare **CLIENT** trebuie să închirieze una sau mai multe **MAȘINI**.

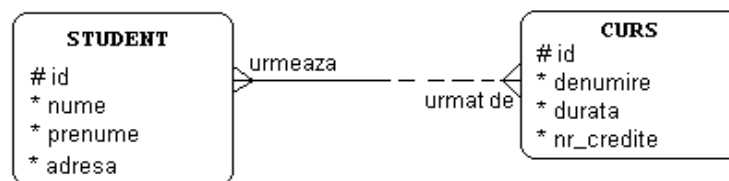
3. Numele unei entități este de obicei:

- a) un verb
- b) un substantiv
- c) un adverb
- d) orice cuvânt


4. Care dintre următoarele variante **NU** poate reprezenta un atribut al entității **PANTOF**?

- a) culoare
- b) mărime
- c) model
- d) clasa

5. Care dintre următoarele fraze poate fi citită din schema de mai jos?



- a) Un student poate să urmeze mai multe cursuri.
- b) Un curs poate fi urmat de mai mulți studenți.
- c) Un student trebuie să urmeze un singur curs.
- d) Un curs trebuie să fie urmat de un student.

6. Ce semnificație are piciorul de cioară (  ) în cadrul unui ERD?

- a) relația este obligatorie
- b) relația este opțională
- c) pot exista una sau mai multe instanțe ale entității lângă care apare semnul în relație cu o instanță a celeilalte entități
- d) niciuna dintre variantele anterioare.

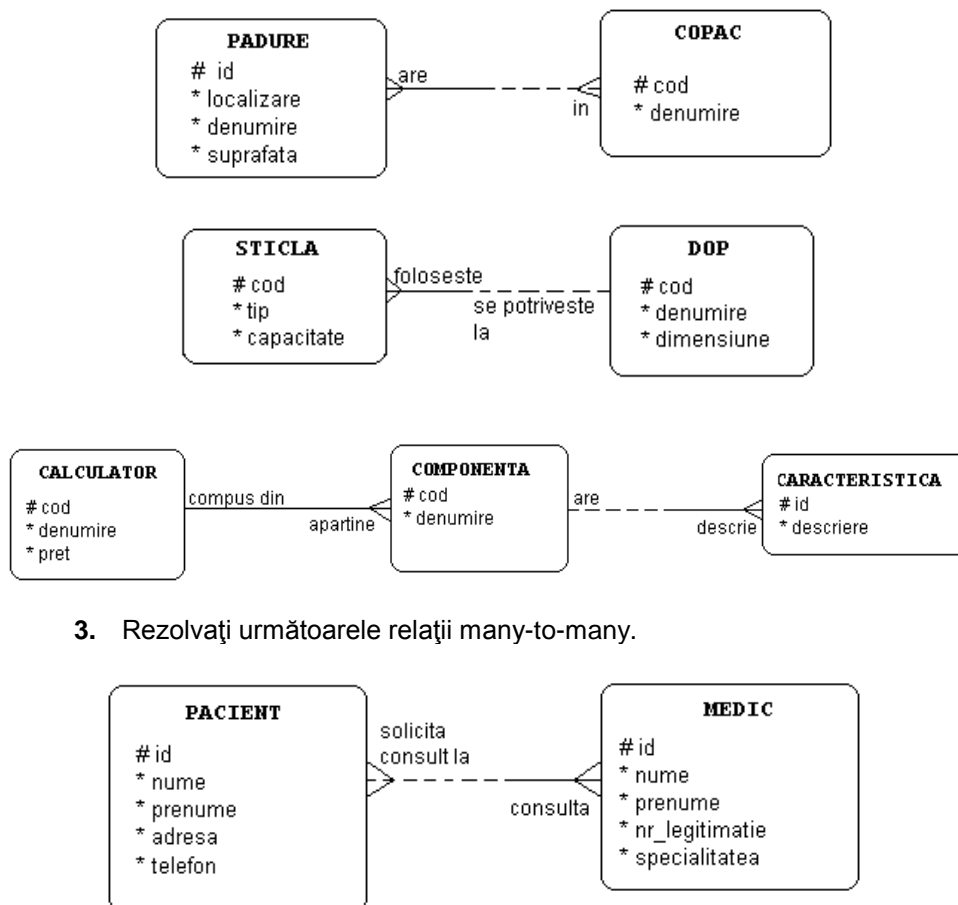


## Test de evaluare 1

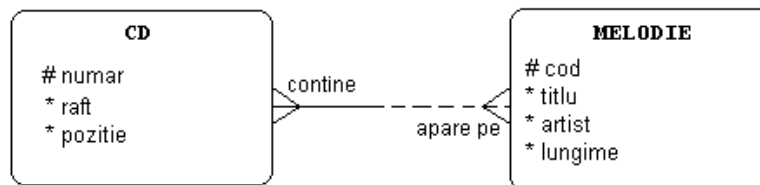
1. Completați în tabelul următor, în prima coloană, câte un exemplu de entitate a cărui atribut este specificat în coloana a doua.

Entitate	Atribut
	culoare
	nr_calorii
	volum

2. Citiți, în ambele sensuri, următoarele relații. Din ce attribute este compus UID-ul fiecărei entități?



3. Rezolvați următoarele relații many-to-many.



4. Dați două exemple de relații one-to-many.

(vezi baremul de corectare la pagina 311)

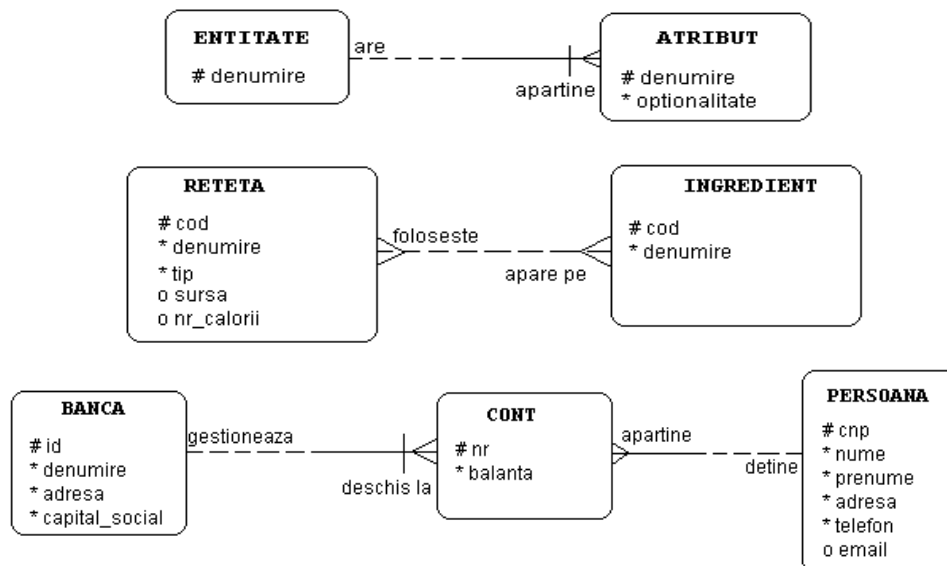


## Test de evaluare 2

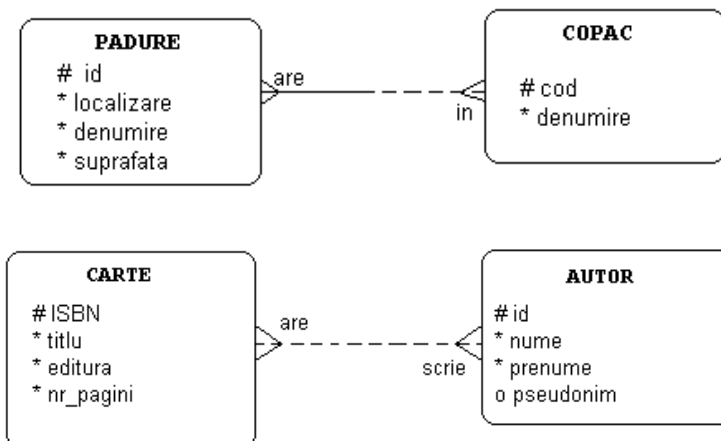
- Completați în tabelul următor, în prima coloană, câte un exemplu de entitate pentru care în coloana a doua este dat un exemplu de instanță.

Entitate	Instanță
	Brad
	Roșu
	Monitor Samsung 17"

- Citiți următoarele relații. Din ce atribute este compus UID-ul fiecărei entități?



3. Rezolvați următoarele relații many-to-many. Stabiliți cel puțin un atribut pentru entitatea de intersecție. Stabiliți UID-ul entității de intersecție.



4. Dați două exemple de relații one-to-many.

(vezi baremul de corectare la pagina 311)



## Aplicații

1. Reluați aplicațiile de la paginile 16-17 și stabiliți relațiile dintre entitățile de la fiecare exercițiu. Rezolvați apoi eventualele relații many-to-many. Verificați să nu existe relații redundante în schemele obținute.

**Pentru scenariile de la punctele 2-6 determinați entitățile, attributele acestora, relațiile dintre entități. Desenați harta relațiilor pentru fiecare exercițiu în parte.**

2. O firmă produce mai multe tipuri de mașini, un model fiind caracterizat printr-un nume, mărimea motorului și un sufix care indică gradul de lux al acesteia (de exemplu XL, GL). Fiecare model este construit din mai multe părți, fiecare parte putând fi folosită pentru construirea mai multor modele de mașini. Fiecare parte are o descriere și un cod. Fiecare model de mașină este produs de exact o fabrică a firmei, fabrică ce se poate găsi în una din țările UE. O fabrică poate produce mai multe modele de mașini și mai multe tipuri de părți componente. De asemenea fiecare tip de parte componentă poate fi produsă de o singură fabrică a firmei.

3. O universitate are în componența sa mai multe facultăți, fiecare facultate având mai multe departamente. Fiecare departament oferă studenților mai multe cursuri. Un profesor poate lucra la un singur departament al unei singure facultăți. Fiecare curs are mai multe secțiuni, iar o secțiune poate să facă parte din mai multe cursuri. Un profesor poate preda mai multe secțiuni, din același curs sau din cursuri diferite, dar o secțiune poate fi predată de mai mulți profesori.

**4.** La o facultate este nevoie să se memoreze date despre studenți, cursuri și secțiunile fiecărui curs. Fiecare student are un nume, un număr de identificare, adresa de acasă, adresa temporară, pentru cei care nu fac facultatea în localitatea lor. Un student poate opta să urmeze un curs întreg sau doar anumite secțiuni ale unui curs. De asemenea el poate urma mai multe cursuri și/sau secțiuni de curs simultan. Un curs poate avea mai multe secțiuni dar o secțiune poate fi parte a mai multor cursuri.

**5.** Angajații unei firme sunt asignați la diferitele departamente din cadrul firmei. Dorim ca în baza de date să memorăm pentru fiecare angajat departamentul la care lucrează acum, dar și departamentul la care a lucrat prima dată, la angajarea în firmă.

**6.** O companie de transport deține mai multe autobuze. Fiecare autobuz este alocat unei anumite rute, pe o anumită rută putând exista mai multe autobuze. Fiecare rută trece prin mai multe orașe.

Unul sau mai mulți șoferi sunt însărcinați pentru fiecare porțiune dintr-o rută, dată prin orașul de unde preia cursa și orașul în care predă cursa altui șofer. Așadar pe o rută se pot schimba șoferii unui autobuz. Un șofer poate conduce mai multe autobuze.

În unele orașe există garaje în care autobuzele pot staționa. Fiecare autobuz este identificat prin numărul de înregistrare și are o anumită capacitate. Fiecare rută este identificată printr-un număr. Șoferii sunt identificați printr-un id și se cunoaște despre aceștia numele, adresa și uneori, numărul de telefon.



1. Proiectarea bazelor de date. Noțiuni introductive
2. Normalizarea datelor
3. Implementarea modelului conceptual
4. Elemente avansate de proiectare a bazelor de date
5. Dezvoltarea profesională în domeniul IT
6. Managementul de proiect

În acest capitol veți afla:

- ✓ care sunt anomaliile care pot apărea la o bază de date
- ✓ ce înseamnă normalizarea
- ✓ care sunt formele normale
- ✓ care sunt regulile pe care trebuie să le respecte o entitate pentru a se afla în una dintre formele normale 1NF, 2NF și respectiv 3NF
- ✓ cum puteți aduce un ERD la a treia formă normală

## I.2.1. Ce este normalizarea?

Normalizarea este o tehnică de proiectare a bazelor de date prin care se elimină (sau se evită) anumite anomalii și inconsistențe ale datelor. O bază de date bine proiectată nu permite ca datele să fie redundante, adică aceeași informație să se găsească în locuri diferite sau să se memoreze în baza de date informații care se pot deduce pe baza altor informații memorate în aceeași bază de date. Anomaliile care pot să apară la o bază de date nenormalizată sunt următoarele:

- **anomalii la actualizarea** datelor – închipuiți-vă că la secretariatul școlii voastre sunt memorate într-o tabelă informațiile despre toți elevii școlii: nume, adresă, telefon etc. De asemenea, la biblioteca școlii, există fișele voastre tot într-o tabelă. Aceste fișe conțin numele, prenumele, adresa, telefonul, data înscrierii la bibliotecă, etc. Acum câteva zile v-ați schimbat domiciliul. Noua adresă trebuie modificată la secretariat, în fișa de la bibliotecă, și în toate locurile în care această informație apare. Dacă modificarea nu se produce într-unul dintre aceste locuri (fie că ați uitat să anunțați fie din alte motive) datele devin inconsistente.

Alt scenariu: la o bibliotecă se înregistrează într-o tabelă următoarele date despre cărți: ISBN, titlu, autor, preț, subiect, editură, adresa editurii. La un moment dat o editură își schimbă adresa. Bibliotecara va trebui să modifice adresa editurii respective, în înregistrările corespunzătoare tuturor cărților 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 inconsistență a datelor.

- **anomalii de inserare** – în exemplul anterior, nu vom putea memora adresa unei edituri, lucru inacceptabil dacă dorim să avem informații și despre edituri a căror cărți nu le avem în bibliotecă, eventual de la care dorim să facem comenzi.
- **anomalii de ștergere** – să presupunem că într-o tabelă memorăm următoarele informații: codul studentului, codul cursului, codul profesorului. La un moment dat, nici un student nu mai dorește să participe la un anumit curs. Ștergând toate înregistrările corespunzătoare cursului, nu vom mai putea ști niciodată cine predă acel curs.

Conceptul de normalizare a bazelor de date a fost pentru prima dată introdus de către Edgar Frank Codd<sup>1</sup>. Formele normale oferă indicații pe baza cărora puteți decide dacă un anumit ERD este bine proiectat, neexpus anomaliilor și inconsistențelor. În principiu, normalizarea implică descompunerea unei entități în două sau mai multe entități, prin compunerea cărora se pot obține exact aceleași informații.

---

<sup>1</sup> <http://www.acm.org/classics/nov95/toc.html>

Formele normale se aplică fiecărei entități în parte. O bază de date (sau un ERD) se găsește într-o anumită formă normală doar dacă toate entitățile se găsesc în acea formă normală.

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

## I.2.2. Prima formă normală

O entitate se găsește în prima formă normală dacă și numai dacă:

- nu există attribute cu valori multiple
- nu există attribute sau grupuri de attribute care se repetă.

Cu alte cuvinte toate attributele trebuie să fie atomice, adică să conțină o singură informație.

Dacă un atribut are valori multiple, sau un grup de attribute se repetă, atunci trebuie să creați o entitate suplimentară pe care să o legați de entitatea originală printr-o relație de 1:m. În noua entitate vor fi introduse attributele sau grupurile de attribute care se repetă.

Să considerăm entitatea din figura I.2.1, referitoare la notele elevilor unei clase. Câteva observații referitoare la această entitate: câte discipline studiază un elev? Câte perechi (disciplina, nota) va trebui să aibă entitatea **ELEV**? Să spunem că știm exact care este numărul maximum de discipline ce pot fi studiate de către un elev. Ce se întâmplă dacă în anul viitor școlar 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 și 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 restricție conform acestui model fiind că un elev nu va putea primi în aceeași zi, la aceeași materie, mai multe note.

Să considerăm un alt exemplu. Pentru managementul unui proiect este important să știm pentru fiecare membru al echipei care sunt abilitățile de care dispune, pentru a ști în ce mod să atribuim sarcinile în cadrul grupului. Într-o primă etapă am proiectat o entitate **ANGAJAT**, care are un atribut **abilități**, ca în figura I.2.3.

Însă se știe că fiecare angajat are mai multe abilități pe care dorim să le memorăm. Așadar, atributul **abilități** nu respectă prima formă normală. De aceea vom crea o nouă entitate **ABILITATE** în care vom memora toate abilitățile fiecărui angajat (figura I.2.4.).

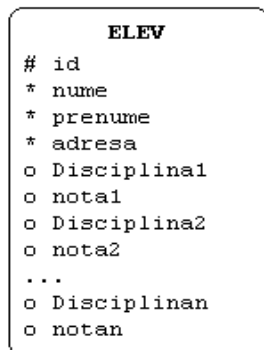


Figura I.2.1.

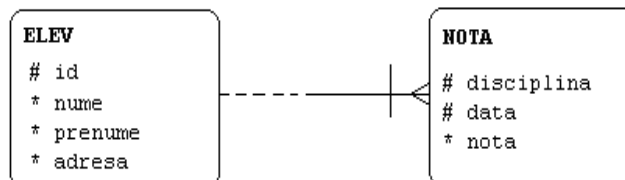


Figura I.2.2

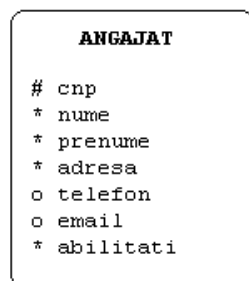


Figura I.2.3.

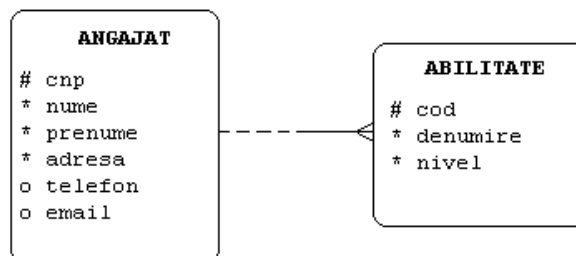


Figura I.2.4.

Un alt exemplu de încălcare a regulilor primei forme normale, puțin mai "ascuns", este prezentat în figura I.2.5. De ce? Pentru că adresa este de tipul "str. Florilor, bl. 45, sc. A, ap. 28, etaj 3, Brașov, cod 123123", formă care de fapt conține mai multe informații elementare. În mod normal acest atribut ar trebui "spart" în mai multe atribute ca în figura I.2.6.

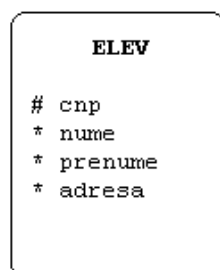


Figura I.2.5.

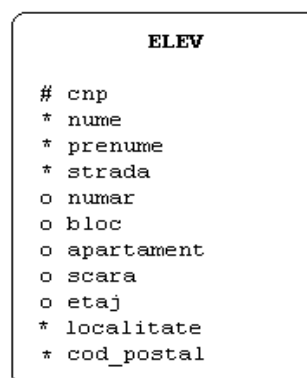


Figura I.2.6.

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

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

### I.2.3. A doua formă normală

O entitate se găsește în a doua formă normală dacă și numai dacă se găsește în prima formă normală și 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 angajații unui departament într-o entitate ca mai jos:

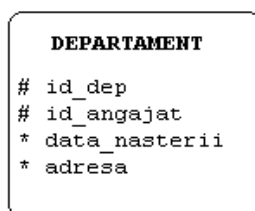


Figura I.2.7. Entitatea DEPARTAMENT

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

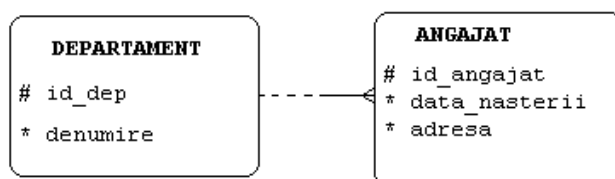


Figura I.2.8.

O situație mai specială este în cazul relațiilor barate, când trebuie ținut seama că UID-ul unei entități este compus din atribute din entitatea respectivă plus un atribut sau mai multe atribute provenite din relația barată. Să considerăm următorul exemplu:

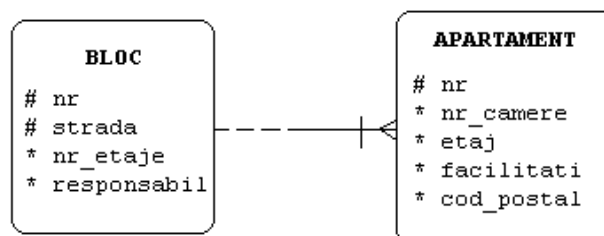


Figura I.2.9.

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

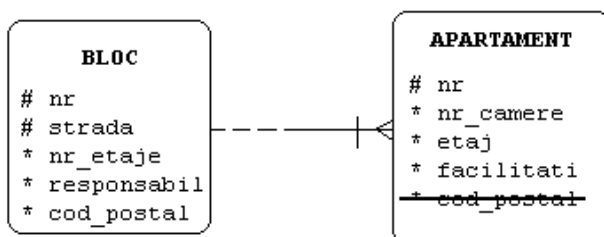


Figura I.2.10.

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

## I.2.4. A treia formă normală

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

Luăm ca exemplu entitatea **CARTE** din figura I.2.11. Atributul **biografie\_autor** nu depinde de **ISBN** ci de atributul **autor**. Nerezolvarea acestei situații duce la memorarea de date redundante, deoarece biografia unui autor va fi memorată pentru fiecare carte scrisă de autorul respectiv. Rezolvarea acestei situații constă în crearea unei noi entități **AUTOR**, pe care o legăm de entitatea **CARTE** printr-o relație **1:m** (figura I.2.12).

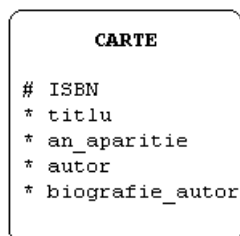


Figura I.2.11.

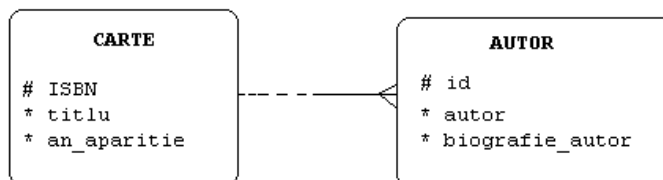


Figura I.2.12.

**Atenție!** Acest model este corect doar dacă se acceptă că o carte are un singur autor. Lăsăm ca temă rezolvarea situației în care o carte poate avea mai mulți autori. În această situație apare o relație many-to-many, pe care trebuie să o rezolvați.

## I.2.5. Exemplu de normalizare

Vom considera următorul scenariu: am fost solicitați să proiectăm o bază de date în care să memorăm datele despre toate operațiile dintr-o clinică privată. Pentru fiecare operație efectuată se memorează codul pacientului, numele și adresa sa, numele chirurgului, codul operației efectuate, data la care a avut loc, denumirea operației, tratamentul administrat după operație.

Într-o primă fază am putea crea o singură entitate cu toate aceste informații (figura I.2.13.). Vom rafina acest model pentru a-l aduce până la forma normală 3NF.

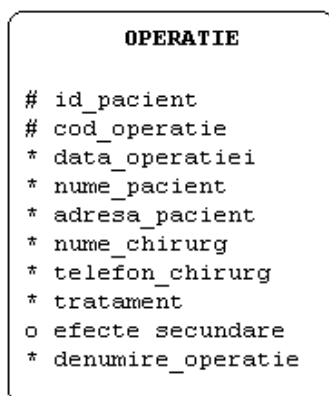
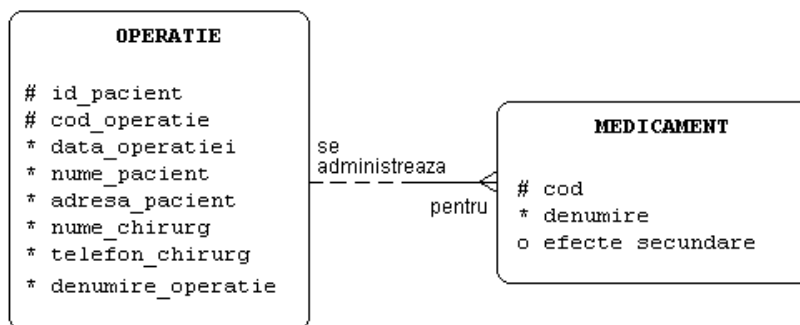


Figura I.2.13. Exemplu de entitate

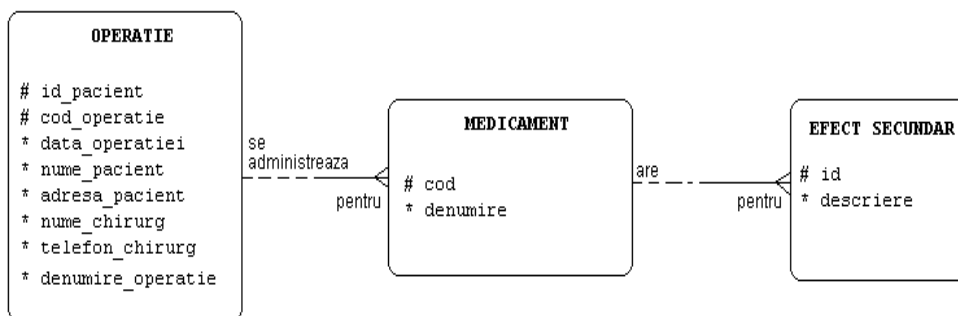
**Prima formă normală.** Este evident că în general un tratament nu constă doar dintr-un singur medicament, ci din mai multe medicamente, fiecare cu efectele sale

secundare. Așadar vom crea o nouă entitate **MEDICAMENT**, pe care o legăm de entitatea operație printr-o relație de 1:m (figura I.2.14).



**Figura I.2.14.** Crearea unei noi entități, **MEDICAMENT**

Nici acest model nu este însă în forma normală 1NF, pentru că fiecare medicament poate avea mai multe efecte secundare. De aceea, vom crea o entitate în care să memorăm efectele secundare ale medicamentelor (figura I.2.15).



**Figura I.2.15.** Prima formă normală

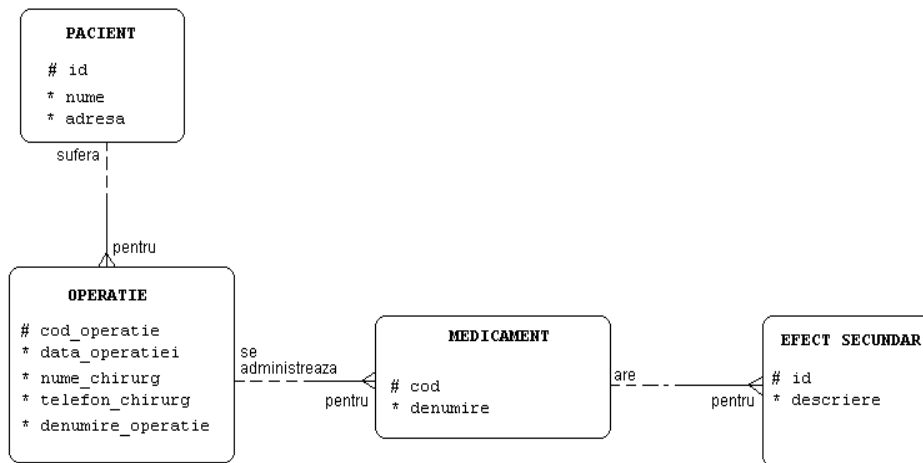
În acest moment putem considera ERD-ul în prima formă normală.

**Forma normală 2NF.** Pentru a aduce schema anterioară în a doua formă normală trebuie să rezolvăm două probleme:

- numele și adresa pacientului observăm că nu depind de întregul UID (**id\_pacient + cod\_operatie**) ci doar de o parte a acestuia și anume de **id\_pacient**.
- Denumirea operației depinde doar de **cod\_operatie** nu de întregul UID.

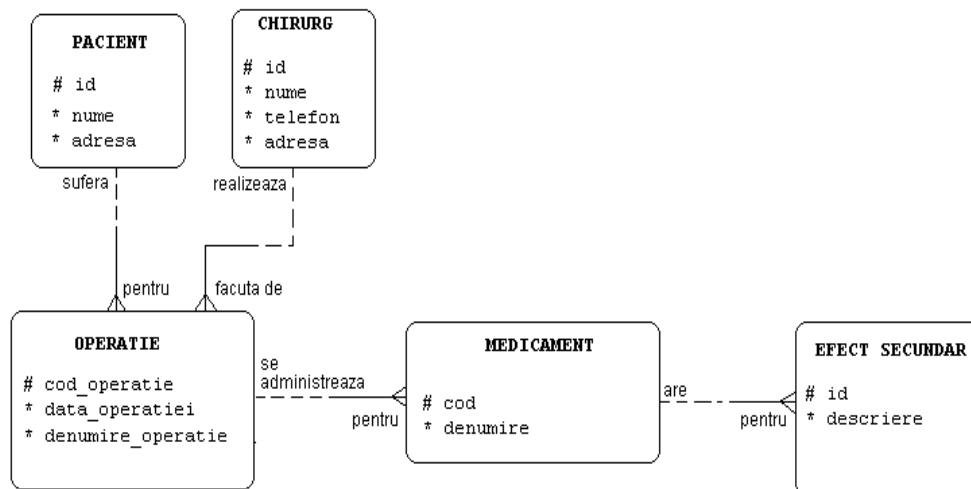
Vom crea o nouă entitate în care vom memora toate datele despre pacient:





**Figura I.2.16.** A doua formă normală

**Forma normală 3NF.** Observăm pe schema anterioară că telefon\_chirurg nu depinde de cod\_operatie ci doar de nume\_chirurg. Așadar, datele despre chirurg va trebui să le memorăm într-o nouă entitate **CHIRURG**.



**Figura I.2.17.** A treia formă normală

Această schemă este acum în forma a treia normală.



## Aplicații<sup>2</sup>

1. Care dintre următoarele enunțuri NU este un exemplu de redundanță?

- a) O relație între două entități care poate fi dedusă din altă relație.
- b) O valoare dintr-o bază de date care poate fi obținută direct pe baza altei valori.
- c) Două atribute din baza de date care au aceeași valoare.
- d) O valoare din baza de date care poate fi obținută efectuând diferite calcule asupra altor valori.
- e) Niciuna dintre variantele anterioare.

2. Care dintre următoarele cerințe NU sunt necesare pentru ca o entitate să se găsească în a treia formă normală?

- a) Trebuie să se găsească în a doua formă normală.
- b) Fiecare atribut care nu face parte din UID trebuie să depindă de întregul UID.
- c) Nu trebuie să existe dependențe tranzitive.
- d) Niciuna dintre variantele anterioare.

3. Care este cea mai avansată formă normală în care se găsește entitatea alăturată?

- a) 1NF
- b) 2NF
- c) 3NF
- d) ERD-ul nu este normalizat.

### PROGRAMARE

```
# cod_pacient
# con_doctor
* denumire_spital
* adresa_spital
* data_programarii
* ora_programarii
```

4. În ce formă normală se găsește fiecare dintre următoarele entități?

a)

```
ANGAJAT
# id
* nume
* job_id
```

b)

```
ANGAJAT
# id
* nume
* job_id
* descriere_job
```

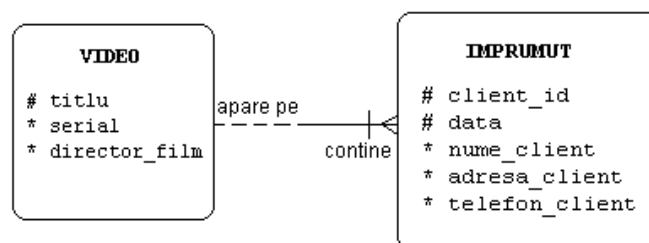
c)

```
ANGAJAT
# id
* nume
# proiect
* nr_ore_lucrate_la_proiect
```

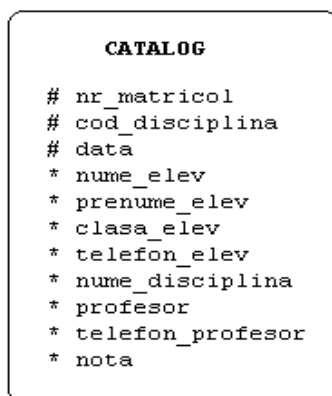
<sup>2</sup> O mare parte dintre aplicațiile din această secțiune sunt preluate și adaptate de pe site-ul <http://db.grussell.org> cu acordul domnului dr. Gordon Russell.

5. Un magazin vinde o gamă variată de pantofi de diferite mărimi și modele. Un model este identificat printr-un cod. Fiecare model are o descriere și aceeași descriere se poate aplica mai multor modele. Atributul **vanzare\_saptamanala** va memora numărul de pantofi de un anumit model și o anumită mărime vânduți săptămâna anterioară (de exemplu, 25 de perechi model 17, mărimea 39). Atributul **valoare\_lunara\_model** reprezintă valoarea totală a pantofilor vânduți pentru fiecare model în parte, indiferent de model. Desenați un ERD în forma normală 3NF, conținând toate aceste informații.

6. Se dă următoarea schemă a unei baze de date existente într-o videotecă. Presupunând că videoteca dispune de un singur exemplar din fiecare film video, stabiliți în ce formă normală se găsește acest ERD. Dacă el nu se găsește în forma normală 3NF, faceți modificările necesare pentru aducerea sa la forma normală 3NF.



7. Plecând de la următoarea entitate, desenați un ERD în forma normală 3NF.



8. Desenați ERD-ul pentru următorul scenariu și aduceți-l în forma normală 3NF: Într-o clădire se găsesc mai multe birouri. Fiecare birou este identificat unic printr-un număr. În fiecare birou se găsește un singur telefon. Un telefon poate fi de două tipuri: telefon interior (cu care nu se pot face apeluri în afara clădirii) și telefon exterior, cu care se pot face apeluri atât în interiorul clădirii cât și cu exteriorul. Fiecare telefon are un număr unic. Într-un birou pot lucra mai mulți angajați, pentru fiecare cunoscându-se numele, prenumele, adresa, e-mail-ul, data nașterii și data angajării. Se știe că un angajat poate lucra într-un singur birou.

9. Modificați ERD-ul de la problema anterioară în ipoteza că într-un birou pot exista mai multe telefoane, folosite în comun de către toți angajații care lucrează în acel birou.

10. Aduceți modificările necesare entității alăturate astfel încât să obțineți un ERD în forma normală 3NF. Entitatea reține informații despre angajații unei agenții de plasare a forței de muncă, care oferă personal cu normă întreagă sau cu program redus, pentru diferite hoteluri din întreaga țară. Se memorează numărul de ore lucrate de fiecare angajat în diferite hoteluri. Se știe că numărul de contract este întotdeauna dependent de codul hotelului dar nu și invers.

SERVICII
# cnp
# nr_contract
* nr_ore
* nume_angajat
* cod_hotel
* localitate_hotel

11. O firmă de consultanță în domeniul software-ului dorește să păstreze într-o bază de date următoarele informații despre angajații săi și proiectele la care aceștia lucrează: codul angajatului, numele și adresa acestuia, salariul, codul actualului post ocupat de angajat, istoricul tuturor posturilor ocupate în timp de către angajat, locația biroului, numărul de telefon, codul și denumirea proiectului la care lucrează angajatul, codul, numele și data la care trebuie finalizată sarcina concretă în cadrul proiectului, codul și denumirea departamentului în care lucrează.

Se știe că numărul de telefon depinde de locația biroului și pot exista mai mulți angajați în cadrul aceluiași birou. De asemenea pot exista mai multe telefoane în același birou. Sarcinile în cadrul proiectului sunt numerotate unic. Se știe că un angajat poate lucra simultan la mai multe sarcini în cadrul aceluiași proiect sau pentru proiecte diferite, însă un angajat lucrează într-un singur departament.

Proiectați un ERD în forma normală 3NF corespunzător acestui scenariu.

## Implementarea modelului conceptual

1.3

1. Proiectarea bazelor de date. Noțiuni introductive
2. Normalizarea datelor
3. Implementarea modelului conceptual
4. Elemente avansate de proiectare a bazelor de date
5. Dezvoltarea profesională în domeniul IT
6. Managementul de proiect

În acest capitol veți afla:

- ✓ ce modele de baze de date există
- ✓ care sunt principalele caracteristici ale bazelor de date relaționale
- ✓ cum se transformă modelul conceptual al bazei de date (ERD-ul) în modelul fizic (tabelele bazei de date)
- ✓ care sunt principalele operații care se pot efectua asupra bazelor de date
- ✓ care este rolul regulilor de integritate și care sunt principalele reguli de integritate
- ✓ ce sunt programele de validare și acțiune

### I.3.1. Modele de baze de date

Bazele de date au fost concepute pentru stocarea volumelor mari de informații relativ omogene între care se pot stabili anumite relații. O bază de date este deci o colecție structurată de **date** aflate în interdependență, date care pot fi consultate pentru a răspunde diferitelor interogări. Înregistrările returnate ca răspuns la o interogare devin **informații** care pot fi utilizate în luarea unor decizii ulterioare.

Sistemul complex de programe care permite descrierea, organizarea, memorarea, regăsirea, administrarea și securizarea informațiilor dintr-o bază de date se numește **sistemul de gestiune a bazelor de date** (SGBD). Memorarea datelor conținute de bazele de date se face pe suporturile de memorie internă sau externă folosite de calculatoare. SGBD este un software special asociat bazelor de date care asigură interfața între o bază de date și utilizatorii ei, rezolvând toate cererile de acces la datele memorate.

Pentru orice bază de date poate fi dată o descriere a datelor și obiectelor memorate, precum și relațiile existente între aceste obiecte. O astfel de descriere se numește **schema** bazei de date.

Există mai multe modele de baze de date, acestea diferențiindu-se în funcție de modul de organizare a schemei bazei de date.

Un model de bază de date nu este doar un mod de structurare a datelor, el definește de asemenea un set de operații care pot fi realizate cu datele respective.

Cele mai cunoscute modele de baze de date sunt următoarele:

- **Modelul tabelar**, în care toate datele sunt memorate sub forma unui singur tabel, un tablou bidimensional de date.
- **Modelul ierarhic** – datele sunt organizate sub forma unor structuri arborescente, există deci o rădăcină cu mai mulți dependenți, care la rândul lor pot avea alți dependenți. IMS (Information Management System) produs de IBM este un exemplu de SGBD bazat pe acest tip de model.
- **Modelul rețea** este un model performant, dar complicat. O bază de date de tip rețea reprezintă o colecție de noduri și legături, fiecare nod putând fi legat de oricare altul. Legăturile trebuie stabilite având tot timpul în minte interogările posibile și acțiunile viitoare probabile.
- **Modelul relațional** reprezintă cel mai utilizat model de stocare a datelor, în care datele sunt organizate sub formă de tabele între care există diverse legături.

- **Modelul obiectual**, destinat să suporte modele de obiecte complexe (organizare de tip heap cu referințe între componente), este oarecum asemănător rețelei, iar prin faptul că pentru accesare directă, stochează o hartă a ierarhiilor și relațiilor claselor de obiecte, are ascendent și în modelul ierarhic. Modelul obiectual se pretează pentru înmagazinarea informațiilor complexe: attribute descriptive asociate datelor multimedia, documentelor, desenelor, arhivelor etc.
- **Modelele hibride** sunt mixturi ale modelelor prezentate anterior, din care cel mai semnificativ este modelul relațional-obiectual, obținut prin extensii ale modelului de organizare tabelar și izvorât din tendința spre universalitate a bazei de date (entități complexe și de naturi diferite, evoluând în condiții eterogene).

### I.3.2. Baze de date relaționale

Bazele de date relaționale au fost dezvoltate având în vedere în primul rând utilizatorii finali. Acest model are la bază teoria matematică a relațiilor, ceea ce a făcut posibilă tratarea algoritmică a proiectării bazelor de date și problema normalizării datelor.

Modelul relațional este un model simplu, bazat pe algebra relațională, care a făcut posibilă dezvoltarea limbajelor relaționale sub forma unui software specializat ce asistă procesul de implementare a bazelor de date. Astfel de limbaje sunt SQL-ul (Structured Query Language) și QBE (Query By Example).

În momentul de față există multe sisteme performante de gestiune a bazelor de date relaționale precum Oracle, DB2, MySQL, Informix, etc.

În cazul bazelor de date relaționale mari și foarte mari, s-au impus SGBD-uri precum Oracle, DB2 și Informix. Acestea au la bază tehnologia client-server.

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

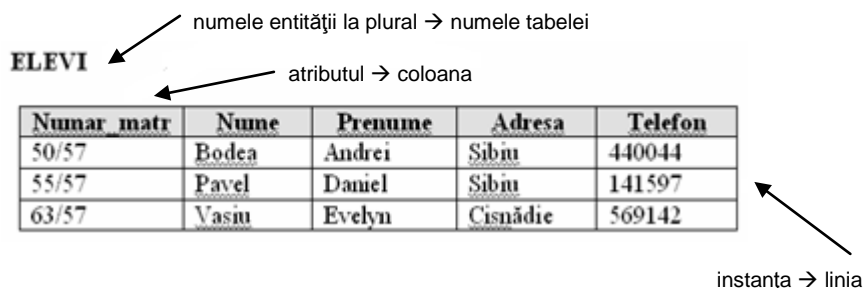
Prima etapă a acestui proces constă în crearea tabelelor bazei de date. Astfel:

- fiecărei entități îi va corespunde câte un tabel. Spre deosebire de entitate, un tabel va avea numele un substantiv la plural. De exemplu entitatea **ANGAJAT** se va transforma în tabela **ANGAJAȚI**, entitatea **ELEV** în tabela **ELEVI**, etc.
- Fiecare atribut al unei entități va deveni o coloană a tablei. Fiecare coloană va memora date de același tip.

- Fiecare instanță a unei entități se va transforma într-un rând (sau înregistrare) a tabelului corespunzător.
- Unicul identificator al entității devine cheia primară a tabelului. Coloana sau combinația de coloane care identifică în mod unic toate liniile unui tabel se numește **cheie primară**.

Deci, orice tabelă are linii și coloane și conține datele organizate conform anumitor structuri. În limbajul bazelor de date, coloanele se numesc câmpuri. Fiecare coloană reprezintă un câmp cu o denumire unică, de un anumit tip (șir de caractere, numeric, dată calendaristică, etc.), având o dimensiune prestabilită. Rândurile tabelului se numesc înregistrări.

Vom vedea pe parcursul următorului paragraf cum mapăm relațiile dintre entități.



**Figura I.3.1. Maparea entităților**

Informațiile despre o tabelă a bazei de date vor fi prezentate folosind **diagramele de tabelă** care sunt niște tabele de forma celui de mai jos, în care vom nota numele coloanelor pe care le va avea tabela bazei de date, notăm dacă o coloană face parte din cheia primară, caz în care vom scrie un **pk** (primary key) în coloana a treia, sau dacă face parte din cheia străină, caz în care vom scrie în coloana a doua un **fk** (foreign key), iar în ultima coloană vom nota dacă atributul este opțional sau obligatoriu. Pentru aceasta vom folosi aceleași simboluri ca și în cazul ERD-ului. Asupra cheilor străine vom reveni în paragraful următor.

În tabelul I.3.1 puteți vedea diagrama tabelului CĂRȚI, corespunzătoare entității CARTE. Se observă că deocamdată nu avem nici o cheie străină, deoarece cheia străină provine din relațiile în care entitatea este implicată. Cum deocamdată această entitate nu are nici o relație cu nicio altă entitate, nu vom avea nicio cheie străină.

**Tabelul I.3.1.**

Numele coloanei	Tip	Tip cheie	Opționalitatea
titlu	Varchar2	Pk	*
autor	Varchar2	Pk	*
data_apariției	Date		*
format	Varchar2		*
nr_pagini	Number		*



Coloana a doua a tabelului este completată cu tipul pe care îl au datele din acea coloană. În tabelul următor sunt prezentate principalele tipuri de date pe care le pune la dispoziție Oracle:

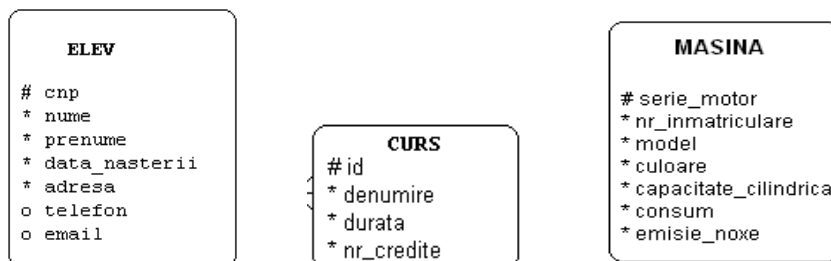
**Tabelul I.3.2.**

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



## Aplicații

Completați diagramele de tabelă pentru următoarele entități:



### I.3.3. Maparea relațiilor

#### Maparea relațiilor one-to-many

Să începem cu un exemplu. Vom considera ERD-ul din figura I.3.2.

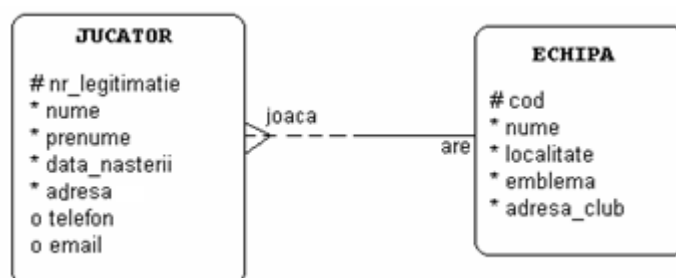


Figura I.3.2. Exemplu de ERD

Să ne reamintim cum se citește relația dintre cele două entități:

Fiecare **JUCĂTOR** poate juca la o **ECHIPĂ** și numai una.

La fiecare **ECHIPĂ** trebuie să joace unul sau mai mulți **JUCĂTORI**.

Observăm că nu putem memora toți jucătorii care joacă la o echipă în cadrul tabelii **ECHIPA**, deoarece ar trebui să introducem o coloană cu valori multiple. Invers însă, putem cu ușurință să memorăm, pentru fiecare jucător, echipa la care joacă, deoarece acesta nu poate juca decât la o singură echipă.

Oare cum putem memora echipa la care joacă un jucător? Răspunsul este destul de simplu. Vom memora pentru fiecare jucător codul echipei la care joacă. Adică diagrama de tabel corespunzătoare entității **JUCĂTOR** va fi următoarea:

Tabelul I.3.3.

Numele coloanei	Tip	Tip cheie	Opționalitatea
Nr_legitimatie	Number	Pk	*
Nume	Varchar2		*
Prenume	Varchar2		*
Data_nasterii	Date		*
Adresa	Varchar2		*
Telefon	Number		o
Email	Varchar2		o
cod echipa	Number	Fk	o

De pe tabela anterioară puteți deduce încă un element important al mapării relațiilor: dacă relația pe partea many este opțională atunci și coloanele cheii străine vor fi opționale. Ce înseamnă acest lucru? Cum un jucător poate la un moment dat să nu joace la nici o echipă, câmpul `cod_echipa` va rămâne necompletat în dreptul lui (va avea valoarea NULL). Dacă însă relația este obligatorie pe partea many, atunci coloanele ce fac parte din cheia străină vor fi obligatorii.

În general, la maparea unei relații de tip one-to-many, vom introduce în tabela corespunzătoare entității de pe partea many a relației, cheia primară a entității de pe partea one a relației. Câmpurile astfel introduse se vor numi **cheie străină** (foreign key).

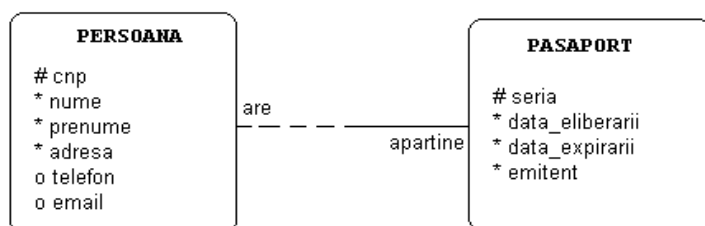
Așadar:

- cheia străină a unei tabeli este cheia primară din tabela referință;
- cheia străină este întotdeauna introdusă în tabela corespunzătoare entității din partea many a relației.

## Maparea relațiilor one-to-one

Dându-se două entități **A** și **B** legate între ele printr-o relație one-to-one, este evident că putem include cheia primară **A** în cadrul tabeli **B**, dar putem proceda la fel de bine și invers, incluzând cheia primară a tabeli **B** în cadrul tabeli **A**, deoarece fiecărei instanțe a entității **A** îi corespunde cel mult o instanță a entității **B**, dar și invers, oricărei instanțe a entității **B** îi corespunde cel mult o instanță a entității **A**.

Pentru relația din figura I.3.3 de exemplu putem memora, pentru fiecare persoană, seria de pașaport, dar și invers, pentru fiecare pașaport, putem memora cnp-ul deținătorului.



**Figura I.3.3.** Exemplu de relație

Decizia depinde de specificul afacerii modelate. Dacă de exemplu ne interesează în primul rând persoanele și abia apoi datele de pe pașapoarte, atunci vom adopta probabil prima variantă, a memorării seriei de pașaport în cadrul

tabelei **PERSOANE**, dacă însă baza de date este destinată evidenței pașapoartelor, atunci probabil vom adopta varianta a doua.

Uneori este convenabil să memorăm cheia străină în ambele părți ale relației, în exemplul nostru pentru fiecare pașaport să memorăm cnp-ul persoanei care îl deține, dar și pentru fiecare persoană să memorăm seria de pașaport.

## Maparea relațiilor recursive

Dacă vom privi o relație recursivă ca pe o relație de tipul one-to-many între o entitate și ea însăși, atunci acest caz se reduce la ceea ce deja am discutat. Să exemplificăm relația din figura I.3.4. Relația recursivă din această figură poate fi privită ca o relație între două entități identice, ca în figura I.3.5.

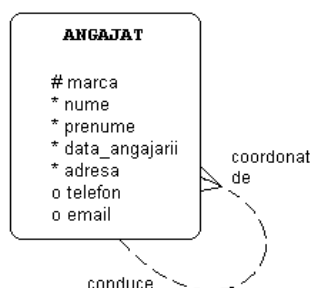


Figura I.3.4.

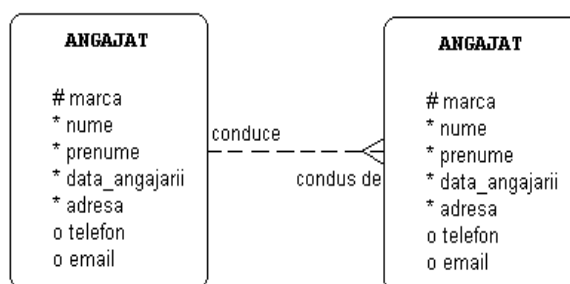


Figura I.3.5.

Așadar vom introduce în cadrul tablei **ANGAJAȚI**, marca șefului său. Diagrama de tabelă va arăta ca mai jos:

Tabelul I.3.4.

Numele coloanei	Tip	Tip cheie	Opționalitatea
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 relațiilor barate

Relațiile barate se transformă în urma mapării în străină în tabela aflată în partea many a relației, la fel ca la maparea oricărei relații one-to-many. Bara de pe relație exprimă faptul că acele coloane ce fac parte din cheia străină vor deveni parte a cheii primare a tabelului din partea many a relației barate.

Pentru exemplul din figura I.3.6, cheia primară a tabelului **ATRIBUT** va fi formată din coloanele **denumire\_atribut** și **denumire\_entitate**, aceasta din urmă fiind de fapt cheie străină în tabela **ATRIBUT**.

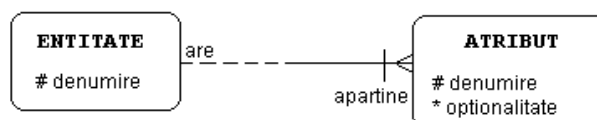


Figura I.3.6. Maparea relațiilor barate

Tabelul I.3.5. Tabela **ENTITĂȚI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
denumire	Varchar2	Pk	*

Tabelul I.3.5. Tabela **ATRIBUT**

Numele coloanei	Tip	Tip cheie	Opționalitatea
denumire_atribut	Varchar2	Pk	*
denumire_entitate	Varchar2	<b>Pk, Fk</b>	*
optionalitate	Varchar2		*

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

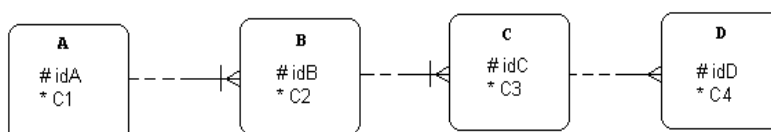


Figura I.3.7. Relații barate în cascadă

Tabelul I.3.6. Tabela **A**

Numele coloanei	Tip cheie	Opționalitatea
idA	Pk	*
C1		*

Tabelul I.3.7. Tabela B

Numele coloanei	Tip cheie	Opționalitatea
idB	Pk	*
C2		*
idA	Pk, Fk	*

Tabelul I.3.8. Tabela c

Numele coloanei	Tip cheie	Opționalitatea
idC	Pk	*
C3		*
idA	Pk, Fk	*
idB	Pk, Fk	*

Tabelul I.3.9. Tabela D

Numele coloanei	Tip cheie	Opționalitatea
idD	Pk	*
C4		*
idA	Fk	*

### I.3.5. Exemplu complet de mapare

Vom exemplifica operația de mapare pe ERD-ul din figura următoare. Ceea ce trebuie subliniat este că în fiecare tabelă pot apărea mai multe chei străine, câte una pentru fiecare relație care are partea many spre entitatea respectivă.

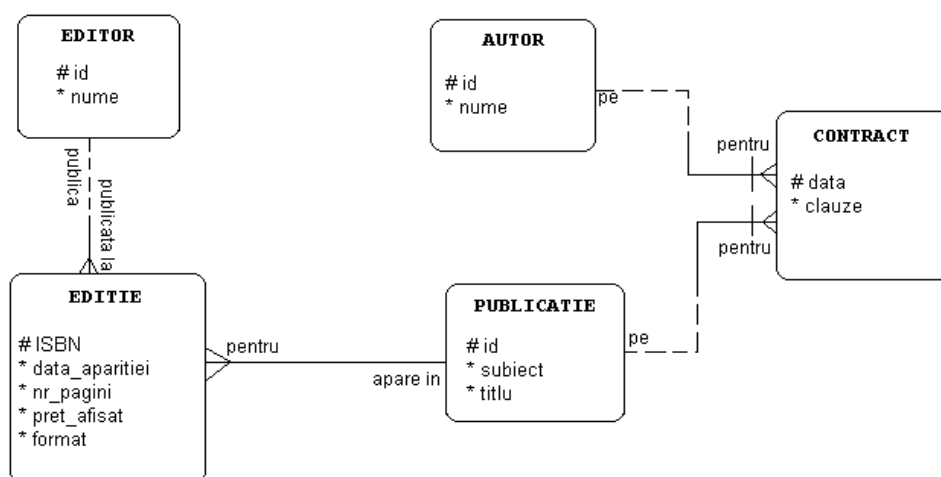


Figura I.3.7. Exemplu de ERD

Diagramele de tabelă corespunzătoare fiecărei entități sunt următoarele:

**Tabelul I.3.10. Tabela EDITORI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id	Number	Pk	*
Nume	Varchar2		*

**Tabelul I.3.11. Tabela EDIȚII**

Numele coloanei	Tip	Tip cheie	Opționalitatea
ISBN	Number	Pk	*
Data_apariției	Date		*
Nr_pagini	Number		*
Pret_afișat	Number		*
Format	Varchar2		*
Id_editor	Number	Fk1	*
Id_publicatie	Number	Fk2	*

**Tabelul I.3.12. Tabela PUBLICAȚII**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id	Number	Pk	*
Subiect	Varchar2		*
Titlu	Varchar2		*

**Tabelul I.3.13. Tabela AUTORI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id	Number	Pk	*
Nume	Varchar2		*

**Tabelul I.3.14. Tabela CONTRACTE**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_autor	Number	Pk, Fk	*
Id_publicatie	Number	Pk, Fk	*
Data	Date	Pk	*
Clauze	Varchar2		*

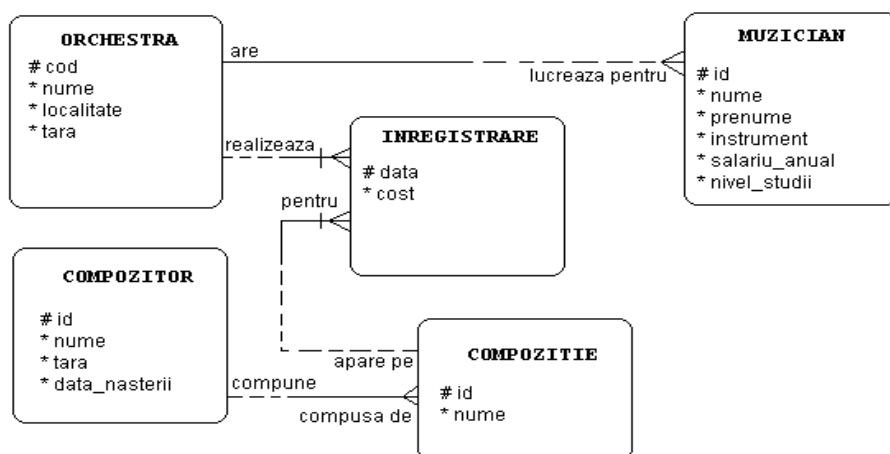
Se observă în ultima diagramă că în cazul relațiilor barate, cheia străină face parte din cheia primară a entității respective.



## Aplicații

1. Reluați exercițiile propuse pe parcursul capitolelor anterioare și realizați maparea ERD-urilor obținute.

2. Realizați maparea următoarei hărți a relațiilor.



### I.3.6. Operații specifice prelucrării bazelor de date

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

- definirea structurii bazei de date
- încărcarea datelor în baza de date (adăugarea de noi înregistrări la baza de date)
- accesul la date pentru:
  - interogare (afișarea datelor, sortarea lor, calcule statistice, etc.)
  - ștergere
  - modificare
- întreținerea bazei de date:
  - refacerea bazei de date prin existența unor copii de siguranță
  - repararea în caz de incident
  - colectarea și re folosirea spațiilor goale



- posibilitatea de reorganizare a bazei de date prin:
  - restructurarea datelor
  - modificarea accesului la date
- securitatea datelor.

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

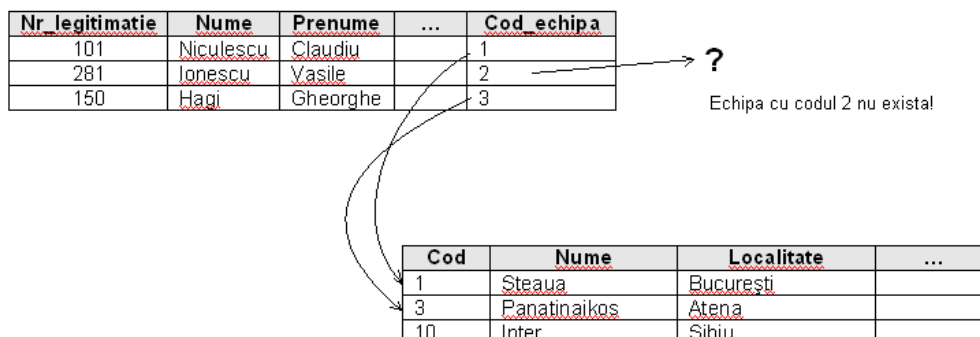
### I.3.7. Reguli de integritate

Detalierea caracteristicilor pe care trebuie să le prezinte un SGBD pentru a fi considerat relațional a fost realizată de către E. F. Codd în 1985 sub forma a 13 reguli. Una dintre aceste reguli precizează că restricțiile 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 și valide. Aceasta înseamnă că dacă există orice regulă sau restricție asupra unei entități, atunci datele introduse în baza de date respectă aceste restricții. 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 entităților** – indică faptul că nici o coloană ce face parte din cheia primară nu poate avea valoarea NULL. În plus, pentru fiecare înregistrare, cheia primară trebuie să fie unică.
- **Integritatea de domeniu** – acest tip de reguli permite ca într-o anumită coloană să se introducă doar valori dintr-un anumit domeniu. De exemplu, putem impune ca salariul unui angajat să fie cuprins între 4500 și 5000 RON.
- **Integritatea referențială** – este o protecție 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** și **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 tabelii **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 integrității referențiale

Situații de încălcare a integrității referențiale 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 **ștergerea** unei înregistrări. De exemplu se șterge înregistrarea corespunzătoare unei anumite echipe (echipa se desființează). Înregistrările jucătorilor care au jucat la acea echipă vor încălca integritatea referențială, deoarece se vor referi la o echipă care nu mai există. Soluțiile posibile sunt ca la ștergerea unei echipe, toți jucătorii care au activat la acea echipă să fie și ei șterși din baza de date (ștergere în cascadă) sau valoarea câmpului **cod\_echipă** pentru acei jucători să fie setată la **NULL**, ceea ce înseamnă că acei jucători nu activează la nicio echipă.

## I.3.8. Programe de validare și de acțiune

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

Regulile afacerii unei organizații se referă în esență la procesele și fluxurile tuturor datelor și activităților zilnice din cadrul organizației. Cum funcționează organizația? Care sunt activitățile sale?

Regulile afacerii acoperă următoarele aspecte ale unei organizații:

- Orice tip de politici organizaționale de orice tip și de la orice nivel al organizației.
- Orice tip de formule de calcul (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 organizației.

Regulile simple ale afacerii pot fi implementate în modelul bazei de date prin intermediul relațiilor dintre entități. 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 totuși reguli pentru implementarea cărora va trebui să scriem programe speciale folosind limbaje specializate specifice SGBD-ului utilizat. Aceste tipuri de reguli se numesc **reguli procedurale**. În Oracle, acest tip de programe se vor scrie folosind limbajul PL/SQL (Procedural Language/Structured Query Language) și se numesc **declanșatoare (triggere)**.

Există două tipuri de declanșatoare:

- declanșatoare de aplicație care se execută când apar anumite evenimente la nivelul anumitor evenimente;
- declanșatoare ale bazei de date care sunt lansate în executare când apar diverse evenimente asupra datelor (de exemplu, la executarea unor comenzi ca **INSERT**, **UPDATE**, **DELETE**) sau la apariția unor evenimente sistem (logarea la baza de date sau delogarea).

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



## Test de autoevaluare<sup>1</sup>

1. Când mapați un ERD, care dintre următoarele afirmații NU este adevărată?
  - a) Fiecare entitate este mapată într-o tabelă
  - b) Fiecare atribut este mapat într-o coloană a tabelului corespunzător
  - c) Fiecare entitate în parte este mapată într-o linie din tabelul corespunzător
  - d) Fiecare relație one-to-many se transformă într-o cheie străină.

---

<sup>1</sup> Întrebările din acest test sunt preluate, cu acordul domnului dr. Gordon Russell, din subiectele de la examenul de baze de date de la Universitatea Napier din Edinburgh (<http://db.grussell.org/resources/exams.html>)

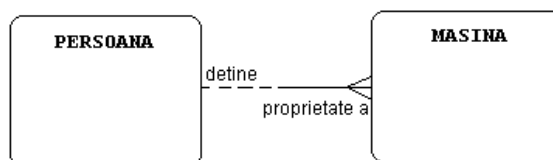
2. Într-o tabelă, o instanță a unei entități este mapată ca

- a) o relație many-to-many
- b) o linie din tabel
- c) o coloană din tabel
- d) un atribut
- e) un index.

3. Două entități A și B se găsesc într-o relație one-to-one care este opțională la ambele capete. Care dintre următoarele variante este o soluție corectă de mapare?

- a) Combinarea celor două entități A și B într-o singură relație
- b) Crearea a două tabele separate și includerea unei chei străine atât în tabela A cât și în tabela B.
- c) Combinarea celor două entități A și B într-o singură tabelă
- d) Utilizarea unei chei primare care să fie o combinație a cheilor primare din A și B

4. Referitor la următorul ERD, care dintre afirmațiile de mai jos corespund unei variante corecte de mapare?



- a) Se preia cheia primară din tabela **PERSOANE** și se adaugă ca și cheie străină la tabela **MASINI**.
- b) Se preia cheia primară din tabela **MASINI** și se adaugă ca și cheie străină la tabela **PERSOANE**.
- c) Se creează o singură tabelă cu informațiile din ambele entități.
- d) Se preiau cheile primare din ambele tabele și se introduc într-o nouă tabelă numită **PROPRIETARI**.
- e) Oricare dintre variantele de mai sus este corectă.

5. O bază de date conține următoarele tabele:

**Tabelul I.3.15. Tabela DEPARTAMENTE**

DepNo	Departament
1	IT
2	Electric
3	Geografie
4	Istorie
5	Business

**Tabelul I.3.16. Tabela ANGAJATI**

IdAngajat	NumeAngajat
1	Ionescu
2	Georgescu
3	Vasilescu
4	Marinescu
5	Andreescu

**Tabelul I.3.17. Tabela ANGAJARI**

IdAngajat	DepNo
1	1
3	2
4	1
3	3
1	2
2	5

Desenați ERD-ul din care s-au obținut prin mapare aceste tabele.

(vezi baremul de corectare la pagina 311)

1. Proiectarea bazelor de date. Noțiuni introductive
2. Normalizarea datelor
3. Implementarea modelului conceptual
4. Elemente avansate de proiectare a bazelor de date
5. Dezvoltarea profesională în domeniul IT
6. Managementul de proiect

În acest capitol veți afla:

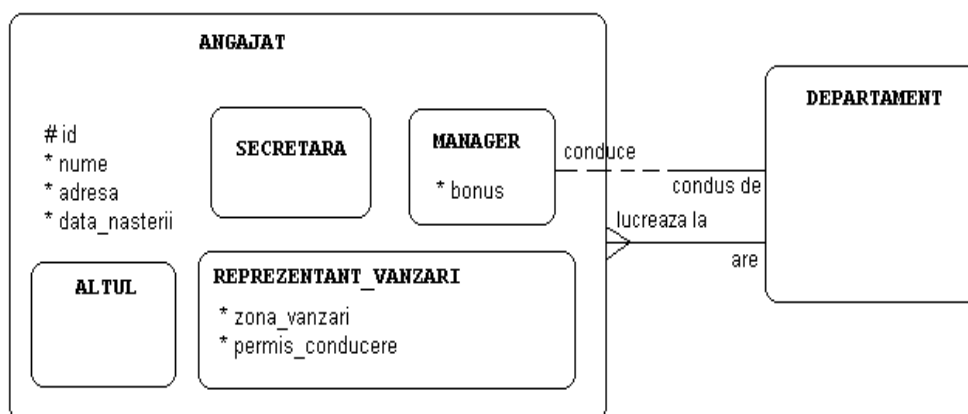
- ✓ cum se modelează clasificările obiectelor cu ajutorul subtipurilor și supertipurilor
- ✓ care sunt posibilele variante de mapare a subtipurilor și supertipurilor
- ✓ ce sunt relațiile exclusive și cum se mapează
- ✓ ce sunt relațiile nontransferabile
- ✓ cum se modelează datele care se modifică în timp

## I.4.1. Tipuri și subtipuri

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

Un **subtip** sau o **subentitate** este o clasificare a unei entități care are caracteristici comune cu entitatea generală, precum atribute și relații. Subtipurile se reprezintă în cadrul hărții relațiilor ca entități în interiorul altei entități. Atributele și relațiile comune tuturor subtipurilor se vor reprezenta la nivelul **supertipului** sau **superentității**. Atributele și relațiile supertipului vor fi moștenite de către subtipuri.

Un subtip poate avea la rândul său alte subtipuri incluse.



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

Subtipurile trebuie să respecte două reguli importante:

- trebuie să acopere toate cazurile posibile de instanțe ale supertipului, cu alte cuvinte, orice instanță a supertipului trebuie să aparțină unui subtip. De multe ori ERD-urile includ un subtip "ALTUL" pentru a acoperi toate situațiile și pentru a permite viitoare dezvoltări ale modelului.
- subtipurile trebuie să se excludă reciproc. Această regulă se traduce în cazul exemplului de mai sus în faptul că un angajat nu poate fi, de exemplu și manager și secretară în același timp.

## I.4.2. Maparea tipurilor și subtipurilor

Niciun sistem de gestiune a bazelor de date nu suportă în mod direct supertipurile și subtipurile. Putem adopta mai multe soluții 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 și câte o tabelă pentru fiecare subtip. Diagramele de tabelă în acest caz vor fi:

**Tabelul I.4.1. Tabela ANGAJAȚI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*
Nume	Varchar2		*
Adresa	Varchar2		*
Data_nasterii	Date		*
Id_departament	Number	Fk	*

**Tabelul I.4.2. Tabela SECRETARE**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*

**Tabelul I.4.3. Tabela MANAGERI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*
Bonus	Number		*
Id_depart_condus	Number	Fk	o

**Tabelul I.4.4. Tabela REPREZENTANȚI\_VÂNZĂRI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
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 și va deveni cheia primară a acelei tabele.

Atributele și cheile străine provenite din relațiile de la nivelul supertipului vor fi memorate în tabela corespunzătoare supertipului. Atributele și relațiile 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 eficiența, întrucât sunt necesare multe operații de interogare din tabele multiple, pentru a obține informații suplimentare despre toți angajații.

**Varianta 2.** Vom crea câte o tabelă pentru fiecare subtip. Atributele și cheile străine provenite din relațiile de la nivelul supertipului vor fi introduse în fiecare tabelă astfel obținută, acestea fiind moștenite de către fiecare subtip.

**Tabelul I.4.5. Tabela SECRETARE**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Id_angajat	Number	Pk	*
Nume	Varchar2		*
Adresa	Varchar2		*
Id_departament	Number	Fk	*
Data_nasterii	Date		*

**Tabelul I.4.6. Tabela MANAGERI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
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 REPREZENTANȚI\_VÂNZĂRI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
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 conține toate coloanele corespunzătoare atributelor de la nivelul supertipului, dar și toate coloanele corespunzătoare tuturor atributelor din toate subtipurile. Atributele de la nivelul supertipului își vor păstra opționalitatea, însă atributele de la nivelul subtipurilor, vor fi toate introduse în tabelă, dar vor fi toate opționale.

Relațiile de la nivelul supertipului se transformă normal. Relațiile de la nivelul subtipurilor se vor implementa cu ajutorul cheilor străine opționale.

**Tabelul I.4.8. Tabela ANGAJAȚI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
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 dacă pentru o înregistrare de un tip anume sunt completate câmpurile corespunzătoare. De exemplu, la adăugarea unui nou manager în tabela **ANGAJAȚI**, trebuie să verificăm dacă este completat câmpul bonus.

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

**Tabelul I.4.9. Tabela ANGAJAȚI**

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, reprezentanții de vânzări cu 2, iar secretarele cu 3. Așadar această variantă de implementare este convenabilă când există puține atribute și relații la nivelul subtipurilor.



## Aplicații

Pentru următoarele scenarii desenați harta relațiilor și apoi realizați maparea sa. Atenție! ERD-ul obținut trebuie să fie adus în a treia formă normală.

1. O firmă închiriază diferite tipuri de mașini identificate printr-un cod unic. Fiecare mașină este caracterizată prin greutatea sa. Camioanele au o capacitate maximă admisă, iar autoturismele se caracterizează prin număr de locuri. Pentru persoanele care închiriază mașini se va memora numele, prenumele și numărul permisului de conducere. O persoană închiriază o mașină pentru o anumită perioadă de timp, dar nu poate închiria simultan mai mult de o mașină.

2. O firmă de software lucrează cu două tipuri de clienți: firme și persoane fizice. Pentru fiecare client, firma poate avea de realizat mai multe proiecte. Fiecare proiect are un cod unic și un șef de proiect. Despre o firmă se cunoaște denumirea, adresa, capitalul social, iar despre persoanele fizice se cunosc codul numeric personal, numele și prenumele, adresa de acasă, adresa de la serviciu, adresa de e-mail. Angajații firmei lucrează la un moment dat la un singur proiect, dar la un proiect pot lucra mai mulți angajați.

3. Într-o firmă pentru fiecare angajat se cunosc numele, prenumele, numărul de telefon, și un id asociat în mod unic la angajare. Personalul din firmă se împarte în personal de secretariat, personal tehnic, și personal de conducere. Pentru angajații de secretariat dorim să memorăm viteza de dactilografie și care sunt procesoarele de text cunoscute. Pentru personalul tehnic dorim să știm care este nivelul de studii. Pentru personalul de conducere dorim să știm ce angajați se găsesc în subordinea fiecăruia.

### I.4.3. Relații exclusive (arce)

În unele situații, relațiile se pot exclude reciproc, adică dintr-un grup de relații, la un moment dat doar una dintre ele poate avea loc. De exemplu, un cont anume la o bancă este deținut, fie de o persoană fizică, fie de o firmă, dar nu de ambele tipuri de clienți simultan.

Un grup de relații exclusive este reprezentat în harta relațiilor printr-un arc peste relațiile care fac parte din respectivul grup, ca în figura I.4.2.

Toate relațiile ce fac parte din grupul de relații exclusive trebuie să aibă aceeași opționalitate. Un arc aparține unei singure entități, adică va include doar relații care pleacă de la o aceeași entitate.

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

Există două tipuri de relații exclusive:

- relații exclusive **obligatorii** în care toate relațiile ce fac parte din arcul respectiv sunt obligatorii, ceea ce înseamnă că de fiecare dată, una dintre relații are obligatoriu loc. Este și cazul din figura I.4.2. Evident că un cont trebuie să fie deținut de o persoană fizică sau de o firmă, o a treia variantă neexistând.

- relații exclusive **opționale** caz în care toate relațiile ce fac parte din arc sunt opționale. În acest caz, de fiecare dată are loc cel mult una dintre relații, existând varianta ca pentru o instanță a entității căreia aparține arcul să nu aibă loc niciuna dintre relațiile din grupul respectiv. În figura I.4.3, este exemplificată situația 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 școlii prevăd ca un elev să nu participe la două astfel de activități extrașcolare. Relațiile fiind opționale, înseamnă că un elev are libertatea de a decide să nu participe la nici o activitate extrașcolară.

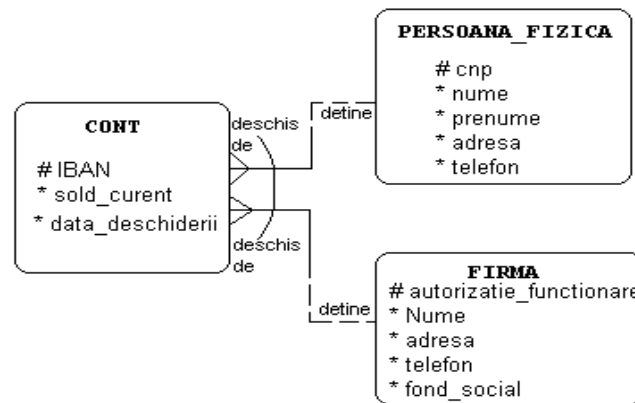


Figura I.4.2. Relații exclusive obligatorii

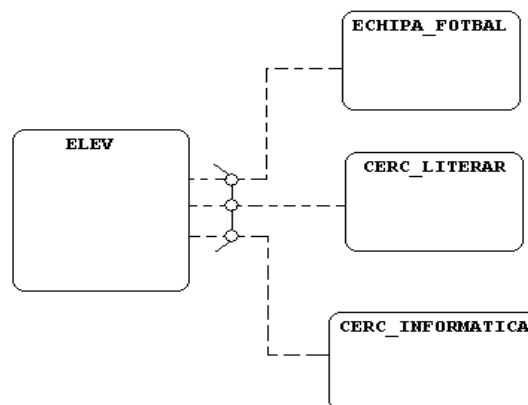


Figura I.4.3. Relații exclusive opționale

## I.4.4. Maparea arcelor

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

**Tabelul I.4.10. Tabela CONTURI**

Numele coloanei	Tip	Tip cheie	Opționalitatea
IBAN	Number	Pk	*
Sold_curent	Number		*
Data_deschiderii	Date		*
Cnp	Number	Fk1	o
Autorizatie_functionare	Number	Fk2	o

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

Numele coloanei	Tip	Tip cheie	Opționalitatea
Cnp	Number	Pk	*
Nume	Varchar2		*
Prenume	Varchar2		*
Adresa	Varchar2		*
Telefon	Number		*

**Tabelul I.4.12. Tabela FIRME**

Numele coloanei	Tip	Tip cheie	Opționalitatea
Autorizatie_functionare	Number	Pk	*
Nume	Varchar2		*
Adresa	Varchar2		*
Telefon	Number		*
Fond_social	Number		*

Deși relațiile din arc sunt obligatorii, cheile străine corespunzătoare au fost setate ca fiind opționale, deoarece pentru fiecare înregistrare trebuie să avem completată una din cele două chei străine, iar cealaltă cheie străină trebuie să rămână necompletată (principiul exclusivității). Va trebui să implementăm o condiție de integritate care să verifice această condiție.



## Aplicații

Reluați exercițiile de la pagina 66 și implementați o soluție folosind de data aceasta arcele. Realizați apoi maparea schemei obținute.

Realizați maparea următorului ERD complex:

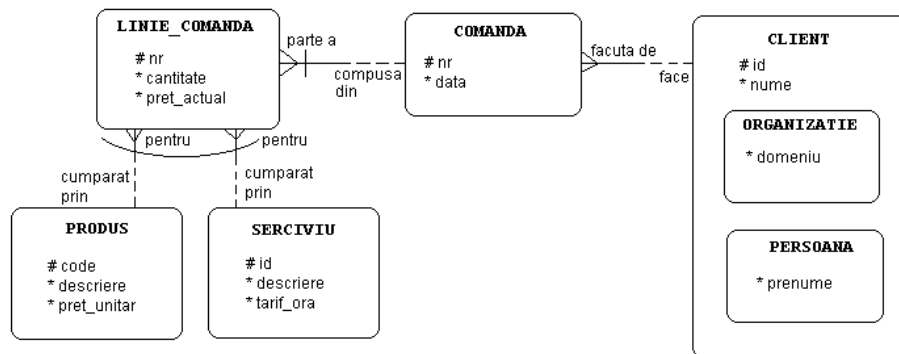


Figura I.4.4. ERD propus ca exercițiu

## I.4.5. Nontransferabilitate

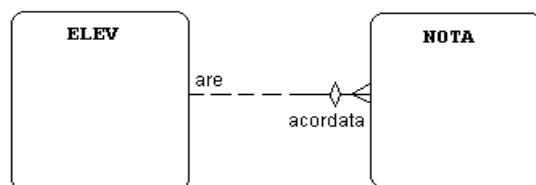
Spunem că o relație este nontransferabilă dacă o asociație între două instanțe ale celor două entități, odată stabilită, nu mai poate fi modificată. Nontransferabilitatea unei relații se reduce la faptul că valorile cheii străine corespunzătoare relației respective nu pot fi modificate.

Nontransferabilitatea anumitor relații poate proveni din reguli speciale ale afacerii modelate. În general, relațiile ce se referă la informații financiare sau la informații care pot fi expuse fraudelor pot fi setate ca fiind nontransferabile.

Condiția de nontransferabilitate a unei relații este asigurată prin program. De aceea trebuie să documentăm această restricție.

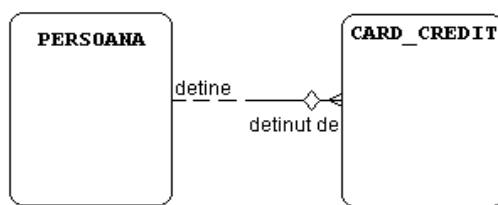
În ERD, o relație nontransferabilă se notează cu un romb pe linia corespunzătoare relației, înspre entitatea a cărei cheie străină nu este permis să o modificăm (adică în partea cu many a unei relații one-to-many).

În figura I.4.5 este dat un exemplu de relație 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.** Relații nontransferabile

Un alt exemplu poate fi dat relativ la cardurile bancare. Se poate oare modifica proprietarul unui card bancar? Evident că nu. Acest caz este modelat de ERD-ul din figura I.4.6.



**Figura I.4.6.** Relații nontransferabile

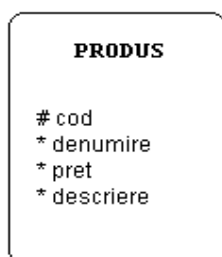
## I.4.6. Modelarea datelor istorice

Viața înseamnă schimbare, orice lucru se schimbă de-a lungul timpului și nu doar obiectele se modifică în timp, ci chiar și relațiile dintre aceste obiecte se schimbă. Prietenii se pot rupe, se leagă alte prietenii, oamenii își schimbă locul de muncă etc.

Și datele dintr-o bază de date pot suferi modificări de-a lungul timpului. Să luăm un exemplu. De ce ar fi important să reținem datele istorice? Imaginați-vă că sunteți un investitor important. Aveți disponibilă o sumă mare de bani pe care doriți să o investiți în acțiuni. Cum decideți ce acțiuni cumpărați? Evident veți studia piața, veți încerca să vedeți ce acțiuni se vând mai bine, care sunt rezultatele diferitelor firme de pe piață pe o anumită perioadă de timp, și veți încerca, pe baza acestor date să faceți previziuni privind viitorul firmelor și acțiunilor respective. Vă veți baza așadar pe istoricul firmelor și acțiunilor respective.

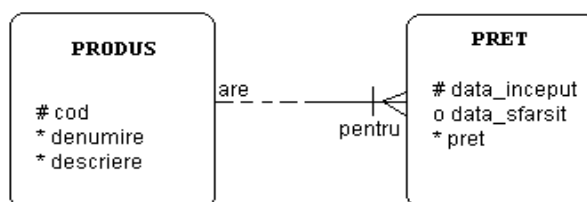
Alt exemplu, sunteți managerul unei importante echipe de fotbal și doriți să vă întăriți echipa. Decideți așadar să achiziționați câțiva jucători noi pentru echipă. Cum decideți ce jucători să aduceți? Veți studia evoluția jucătorilor vizati, care a fost prestația lor în ultima perioada și încercați să „ghiciți” care este potențialul acestor jucători. Și aici aveți nevoie de date istorice, echipele la care a jucat fiecare sportiv, goluri marcate, accidentări suferite (s-ar putea ca, deși este un jucător foarte valoros, să fi suferit numeroase accidentări ceea ce ar putea crea probleme în viitor).

Prețul produselor poate suferi modificări destul de des. Factorii care duc la aceste modificări pot fi dintre cei mai diverși: rata, inflația, anotimpul, etc. Așadar atributul **preț** din cadrul entității **produs** se modifică de-a lungul timpului. Dacă nu ne interesează decât prețul actual al fiecărui produs, modelul este foarte simplu, ca cel din figura I.4.7:



**Figura I.4.7.** Exemplu de model simplu

Dacă însă pentru afacerea modelată este important să reținem un istoric al prețurilor pentru fiecare produs, atunci atributul preț se va transforma într-o nouă entitate (figura I.4.8).

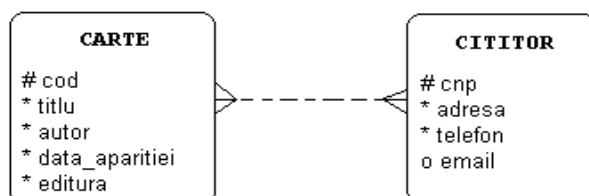


**Figura I.4.8.** Relația dintre cele două entități

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

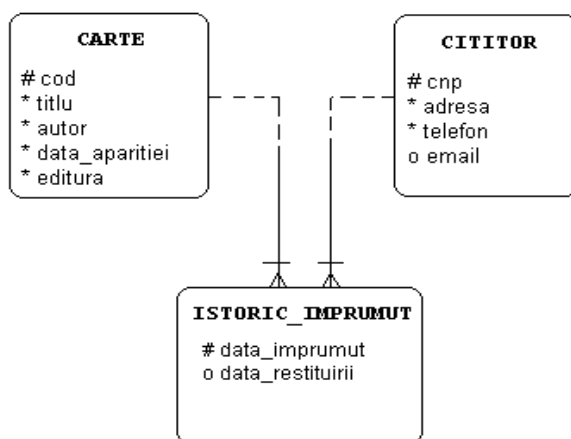
Vom considera acum o situație puțin mai dificilă. Să presupunem că dorim să modelăm o bază de date pentru o bibliotecă. Evident este important de reținut un istoric al tuturor împrumuturilor, deoarece pe baza acestora, se pot afla domeniile de interes ale cititorilor și astfel vom ști ce achiziții de carte să facem în viitor, vom putea determina uzura cărților astfel încât să le putem înlocui, etc.

Într-o primă fază vom obține o relație de many-to-many între entitățile **CARTE** și **CITITOR**. Fiecare carte poate fi împrumutată de mai mulți cititori (evident nu în același timp), și fiecare cititor poate împrumuta mai multe cărți (figura I.4.9).



**Figura I.4.9.** Exemplu de relație many-to-many

Să rezolvăm această relație many-to-many. Aplicând ceea ce am învățat în capitolele anterioare vom obține schema din figura I.4.10.



**Figura I.4.10.** Cazul 1 de barare incorectă a relațiilor

Dacă mapăm acest ERD, diagrama corespunzătoare tabeli **ISTORIC\_IMPRUMUTURI** va fi următoarea:

**Tabelul I.4.13.**

Numele coloanei	Tip	Tip cheie	Opționalitatea
cod_carte	Number	Pk, Fk	*
cnp	Number	Pk, Fk	*
data_imprumut	Date	Pk	*
data_restituirii	Date		o

Cheia primară a tabeli este formată din combinația coloanelor **cod\_carte**, **cnp** cititor, și **data\_imprumut**. Combinația valorilor celor trei coloane trebuie să fie unică pentru fiecare înregistrare în parte. Înregistrările din



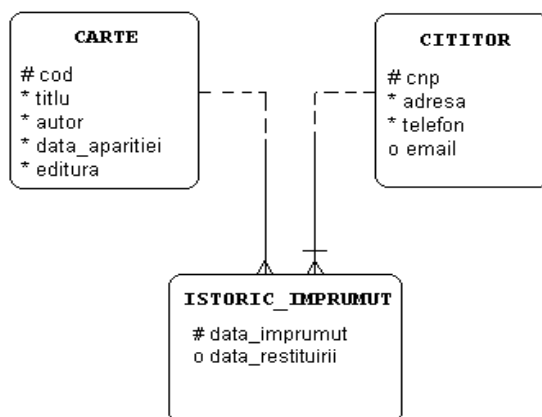
tabelul de mai jos sunt deci două înregistrări valide pentru această tabelă, ele diferind prin valoarea coloanei **cnp**. Dar cum interpretăm valorile celor două înregistrări? Înseamnă că o aceeași carte (cu codul 50214) poate fi împrumutată la aceeași dată (05.04.2007) de către două persoane diferite, lucru care nu este normal în ipoteza că o carte este împrumutată pentru cel puțin o zi.

**Tabelul I.4.14.**

cod_carte	cnp	data_imprumut	data_restituirii
50214	1890502323932	05.04.2007	
50214	2970523256587	05.04.2007	

Ce este de făcut în acest caz? Înseamnă că nu am ales în mod corect cheia primară. Încercăm și alte variante de barare a celor două relații dinspre entitatea de intersecție.

Încercăm să barăm doar relația dintre entitățile **ISTORIC\_ÎMPRUMUTURI** și **CITITOR** ca în figura I.4.11.



**Figura I.4.11.** Cazul 2 de barare incorectă a relațiilor

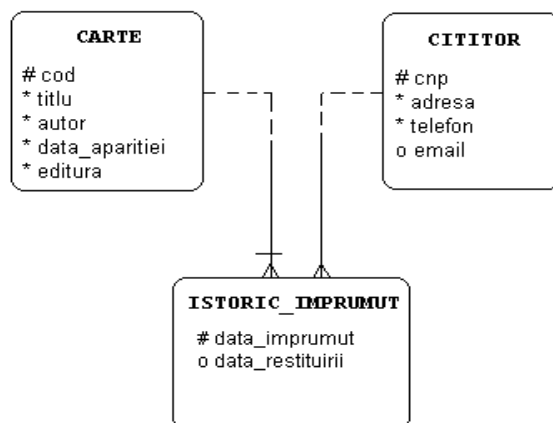
În această nouă situație cheia primară este formată din câmpurile **data\_imprumut** și **cnp**-ul cititorului. Asta înseamnă că primele două înregistrări din tabelul următor nu pot exista simultan în tabela **ISTORIC\_ÎMPRUMUTURI**, adică la o anumită dată, un cititor nu poate împrumuta decât o singură carte, lucru nerealist în cazul celor mai multe biblioteci. În plus situația anormală din cazul anterior se păstrează și în acest caz, adică doi cititori pot împrumuta în aceeași zi o aceeași carte (prima și a treia înregistrare de mai jos pot exista simultan în tabelă).

**Tabelul I.4.15.**

cod_carte	cnp	data_imprumut	data_restituirii
50214	1890502323932	05.04.2007	
35101	1890502323932	05.04.2007	
50214	2970523256587	05.04.2007	

Deci nici această situație nu este cea corectă. Încercăm să nu barăm nici una dintre relații, dar noua situație este chiar mai rea. Cheia primară a tabelului **ISTORIC\_IMPRUMUTURI** fiind **data\_imprumut**, adică la o anumită dată biblioteca împrumută o singură carte.

Singura variantă rămasă este bararea doar a relației dintre **CARTE** și **ISTORIC\_IMPRUMUTURI** (figura I.4.12):



**Figura I.4.12.** Bararea corectă a relațiilor

Să verificăm dacă acest caz este cel corect. Cheia primară este acum combinația coloanelor **cod\_carte** și **data\_imprumut**. Poate un cititor împrumuta două cărți la aceeași dată? Adică cele două înregistrări din tabelul I.4.16 pot exista simultan în tabela **ISTORIC\_IMPRUMUTURI**? Răspunsul este DA, combinația celor două coloane, pentru cele două înregistrări fiind unică.

**Tabelul I.4.16.**

cod_carte	cnp	data_imprumut	data_restituirii
50214	1890502323932	05.04.2007	
35101	1890502323932	05.04.2007	

Pot doi cititori diferiți să împrumute aceeași carte în aceeași zi, adică cele două înregistrări de mai jos pot exista simultan în tabela **ISTORIC\_IMPRUMUTURI**? Se observă că valorile celor două coloane din cheia primară coincid pentru cele două înregistrări, ceea ce nu este permis de regula de integritate a cheii primare. Deci cele două înregistrări nu sunt permise simultan în baza de date ceea ce este foarte bine.

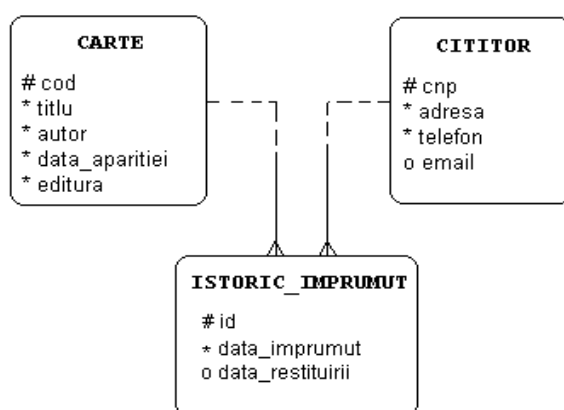
**Tabelul I.4.17.**

cod_carte	cnp	data_imprumut	data_restituirii
50214	1890502323932	05.04.2007	
50214	2970523256587	05.04.2007	

Se observă ușor și că un același cititor poate împrumuta o aceeași carte la două date diferite.

Tragem în final concluzia că modelul din figura I.4.12 este cel corect.

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



**Figura I.4.13.** Introducerea unei chei artificiale



## Aplicații

1. Angajații unei firme sunt repartizați la diverse departamente. În timp un angajat poate fi mutat de la un departament la altul. Firma dorește să țină o evidență a departamentelor la care a lucrat sau lucrează în prezent fiecare angajat. Dorim să știm data la care a fost repartizat un angajat la un departament și până la ce dată a lucrat la acel departament. Realizați ERD-ul corespunzător acestui scenariu.

2. Fie ERD-ul din figura I.4.14, referitor la cursele unei companii de transport aerian.

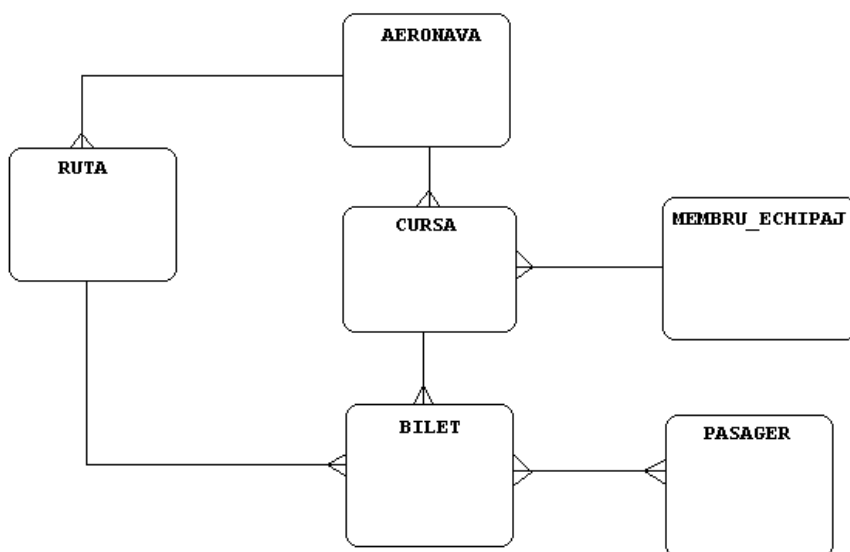
a) Completați cu atribute fiecare entitate a acestei scheme. Stabiliți identifiatorii unici ai fiecărei entități.

b) Stabiliți opționalitatea fiecărei relații din acest ERD.

c) Rezolvați relația many-to-many dintre entitățile **BILET** și **PASAGER**.

d) Stabiliți dacă vreo relație din ERD este nontransferabilă.

- e) Verificați dacă schema obținută este în forma normală 3.
- f) Câte echipaje poate avea o aeronavă?
- g) Câte rute poate acoperi un bilet al unui pasager?
- h) Dacă pentru un membru al echipajului se cunosc numele, adresa și numărul licenței, iar pentru un pasager se știe numele și adresa, cum se poate modifica ERD-ul anterior, astfel încât un membru al echipajului să fie un simplu pasager (la altă cursă de exemplu) ?



**Figura I.4.14.** Exemplu de ERD propus pentru *Aplicația 2*

## Dezvoltarea profesională în domeniul IT

1.5

1. Proiectarea bazelor de date. Noțiuni introductive
2. Normalizarea datelor
3. Implementarea modelului conceptual
4. Elemente avansate de proiectare a bazelor de date
5. Dezvoltarea profesională în domeniul IT
6. Managementul de proiect

În acest capitol veți afla:

- ✓ care sunt etapele planificării carierei
- ✓ cum să vă evaluați aptitudinile și interesele
- ✓ cum să aflați care sunt meseriile care vi se potrivesc
- ✓ cum să scrieți un curriculum vitae
- ✓ cum să vă pregătiți pentru un interviu de angajare

*„Un om este suma acțiunilor sale, a ceea ce a făcut și a ceea ce poate face.”  
Mahatma Gandhi*

Te afli în fața unui pas important din viața ta, acela în care ești pus să decizi, dacă încă nu ai făcut-o, un drum spre viitoarea ta carieră.

Stabilirea planurilor pentru viitor este o sarcină dificilă în societatea noastră. Procesul de autocunoaștere îți dezvăluie alternativele de succes, îți extinde paleta opțiunilor. Așa poți avea convingerea că te afli pe drumul cel bun.

Principalii pași pe care trebuie să îi aveți în vedere pentru a avea succes în planificarea viitoarei voastre cariere sunt pe scurt următorii:

1. Identificarea principalelor voastre aptitudini și interese
2. Identificarea meseriilor de interes
3. Evaluarea posibilelor cariere
4. Crearea curriculum-ului vitae, a scrisorii de intenție și a portofoliului personal
5. Pregătirea și susținerea interviului de angajare
6. Stabilirea unui plan de dezvoltare profesională

### **I.5.1. Evaluarea aptitudinilor și a intereselor**

În domeniul IT, ca în orice domeniu de activitate de altfel, există atât de multe ocupații și posibile cariere, încât îți poți pune întrebarea "Cum să-mi aleg oare cariera care mi se potrivește cel mai bine?"

În cele ce urmează veți învăța cum să identificați acele meserii în care abilitățile voastre să fie importante și care să se potrivească cel mai bine cu preferințele și abilitățile voastre.

Oricine știe cât de importante sunt abilitățile în obținerea sau păstrarea unui serviciu. Angajatorii urmăresc în CV-ul tău abilitățile relevante pentru postul solicitat. Ei vor cere detalii despre aceste abilități în timpul interviului.

Dar nu numai atât. Nu este destul să ai anumite abilități, trebuie să îți și placă să le folosești. De exemplu, poți avea foarte bune abilități de comunicare dar totuși nu-ți face mare plăcere să vorbești în public sau în fața unor persoane necunoscute.

Există un adevăr destul de greu de acceptat de mulți oameni, și pe care mulți îl realizează doar atunci când este deja târziu. Dacă ceea ce facem nu ni se potrivește oricât de mult ne străduim și oricât de mare ar fi salariul, nu vom putea fi decât un profesionist mediocru, lipsit de satisfacție profesională.



### Exercițiul 1

Notați pe caiet sau pe o foaie separată de hârtie cât mai multe dintre abilitățile pe care considerați că le aveți.

Câte abilități ați reușit să identificați? Probabil destul de puține. De ce oare? Nu aveți oare mai multe abilități? Răspunsul este cu siguranță NU. Orice persoană are în jur de 700 de abilități diferite, dar probabil nu ați fost niciodată obligat să vă gândiți la acestea.



### Exercițiul 2

Completați în tabelele următoare, în dreptul fiecărei propoziții, numărul care indică gradul în care sunteți de acord cu afirmația respectivă:

3 = sunt total de acord

2 = sunt de acord

1 = există puțin adevăr în această afirmație

0 = afirmația nu mi se potrivește

Abilități artistice	
	Sunt un artist amator
	Am talent muzical
	Îmi place să redecorez casa
	Îmi place să apar pe scenă
	Îmi place să fac fotografii și să filmez
	Îmi place să scriu povești, poezii, articole, sau eseuri
	Îmi place (mi-a plăcut) să particip la cursuri de balet sau de dans

	Îmi place să gătesc, să pregătesc și să aranjez masa
	Desenez cu ușurință portretul cuiva sau pot să fac o schiță bună unui obiect
	Să cânt (cu voce sau la un instrument) este pentru mine un hobby
	Am un bun simț estetic
	Am jucat într-o piesă de teatru
	Îmi place să mă exprim prin scris
	Pot să prepar mese gustoase mai bine decât majoritatea oamenilor
	Prind foarte ușor pașii unui dans nou

**Total pentru abilitățile artistice:**

Abilități de comunicare	
	Pot să explic cu ușurință, lucruri complicate altor oameni
	Îmi place să iau notițe și să scriu mici conspecte pentru ședințe
	Îmi place să vorbesc în public
	Mă descurc foarte bine să scriu instrucțiuni de utilizare a unui computer sau a unui dispozitiv
	Îmi place să caut și să examinez cauza și contextul evenimentelor sau problemelor și să le arăt și altora ce indică aceste evenimente
	Sunt capabil(ă) să ascult părerile altora și să le înțeleg problemele
	Îmi place să scriu scrisori redactorilor unor ziare sau unor oameni politici
	Știu să-mi susțin punctul de vedere într-o dezbatere
	Îmi place să creez fluturași publicitari pentru diferite evenimente
	Îmi place să îi învăț pe alții cum să conducă o mașină sau cum se joacă un anumit joc
	Reușesc ușor să fac diagrame care descompun procese complexe
	Îmi place să iau cuvântul la diverse întâlniri de grup
	Reușesc întotdeauna să găsesc cele mai potrivite cuvinte
	Îmi place să predau tinerilor
	Nu este o problemă pentru mine să înțeleg conținutul manualelor tehnice

**Total pentru abilitățile de comunicare:**

Abilități interpersonale	
	Sunt capabil(ă) să fac oamenii să simtă că înțeleg punctul lor de vedere
	Îmi place să colaborez cu alții
	Reușesc fac sugestii altor oameni, fără ca aceștia să se simtă criticați
	Îmi place să inițiez sau să particip la campanii de strângere de bani, mâncare, haine sau alte provizii pentru oamenii aflați în nevoie.
	Reușesc cu ușurință să ghicesc gândurile oamenilor
	Am foarte multă răbdare cu oamenii care fac ceva pentru prima dată
	Oamenii mă consideră o persoană deschisă și prietenoasă
	Îmi place să am grijă de prietenii, vecinii sau neamurile bolnave
	Știu să aplanez conflictele dintre prieteni sau membri ai familiei
	Îmi place să primesc musafiri și să fiu gazda lor
	Oamenii mă consideră un jucător de echipă



	Îmi place să întâlnesc oameni noi și să descopăr interesele comune
	Știu să conving oamenii să doneze bani pentru școală, echipe, sau diferite organizații
	Îmi place să dreszez animalele sau să am grijă de acestea
	Deseori știu ce să spun ca să dezamorsez o situație tensionată
	Mi-a plăcut să fiu îndrumător pentru grupuri de tineri

**Total pentru abilitățile interpersonale:**

<b>Abilități de conducere</b>	
	Știu să conving oamenii să lucreze împreună pentru un scop comun
	Tind să utilizez timpul în mod eficient
	Sunt capabil(ă) să iau o decizie importantă fără a mă consulta cu alții
	Îmi place să planific activitățile din școală sau ale unei organizații
	Nu mă feresc să îmi asum responsabilitatea atunci când lucrurile ies rău
	Mi-a făcut (sau îmi face) plăcere să fiu șeful cercetașilor sau a altor grupuri de acest gen
	Înțeleg ușor ce îi motivează pe ceilalți
	Oamenii au încredere să vorbesc în numele lor și să îi reprezint
	Îmi place să ajut la organizarea lucrurilor acasă, precum realizarea listei de cumpărături sau gestionarea bugetului
	Sunt o personalitate remarcată în grupul meu
	Sunt capabil(ă) să planific munca până la detaliu
	Sunt capabil(ă) să conving oamenii, discutând cu ei
	Îmi place să cumpăr cantități mari de mâncare sau alte produse pentru o organizație
	Mă pricep să identific abilitățile altor oameni
	Sunt capabil(ă) să văd dincolo de detalii, tabloul întreg
	Știu să împart munca și autoritatea cu alții, în loc să încerc să fac totul singur(ă)

**Total pentru abilitățile de conducere:**

<b>Abilități matematice</b>	
	M-am descurcat întotdeauna bine la cursurile de matematică
	Îmi face plăcere să țin evidența cecurilor pentru membrii familiei
	Pot să fac calcule mintale cu repeziciune
	Îmi place să calculez statistici sportive
	Calcularea impozitului pe venituri nu este un lucru dificil pentru mine
	Îmi place să ajut copiii la matematică
	Am participat sau intenționez să particip la cursuri de statistică sau algebră
	Îmi place să țin evidența cheltuielilor familiei

**Subtotal pentru abilitățile matematice:**

**x 2 (multiplicați acest subtotal cu 2 pentru a obține totalul)**

**Total pentru abilități matematice:**

Abilități practice	
	Sunt capabil să înțeleg modul de funcționare a diferitelor dispozitive mecanice
	Îmi place să fac călătorii lungi cu mașina sau motocicletă proprie
	Sunt capabil să înțeleg diagramele unor mașini sau a unor cablaje electrice
	Îmi place să instalez sau să repar echipamentul stereo sau computerul de acasă
	Dacă dezmembrez ceva, îmi amintesc cum am făcut-o și pot pune piesele la loc
	Sunt bun(ă) la rezolvarea problemelor tehnice sau repararea unor defectiuni
	Îmi place să construiesc machete de avioane, mașini sau bărci
	Am dexteritate manuală

**Subtotal pentru abilitățile practice:**

**x 2 (multiplicați acest subtotal cu 2 pentru a obține totalul)**

**Total pentru abilități practice:**

Abilități științifice	
	Îmi place să testez și să modific setările de la calculatorul meu pentru a-l face să funcționeze mai bine
	Sunt capabil să înțeleg cum funcționează organele și sistemele din corpul uman
	Îmi place să fac experimente la întâlnirile științifice
	Îmi place să citesc despre noutățile din știință și tehnologie
	Știu să scriu un program într-un limbaj de programare
	Îmi place să citesc reviste medicale sau științifice
	Urmăresc cu plăcere la televizor emisiuni cu caracter științific (de genul Discovery)
	Cele mai bune note le-am avut în școală la disciplinele științifice

**Subtotal pentru abilitățile științifice:**

**x 2 (multiplicați acest subtotal cu 2 pentru a obține totalul)**

**Total pentru abilități științifice:**

Copiază acum totalurile în tabelul următor

Total	Abilitatea
	Abilități artistice
	Abilități de comunicare
	Abilități interpersonale
	Abilități de conducere
	Abilități matematice
	Abilități practice
	Abilități științifice



### Exercițiul 3

În exercițiul anterior ați reușit să analizați abilitățile de care dispuneți. Este timpul acum să extrageți din această listă de abilități acele abilități pe care doriți sau preferați să le folosiți într-o viitoare carieră, cu alte cuvinte să descoperiți care sunt interesele voastre.

Pentru acest exercițiu parcurgeți din nou listele anterioare și păstrați doar acele abilități de care sunteți cel mai interesați să le folosiți, renunțând la acele abilități pe care, deși le aveți, ați prefera să nu le utilizați în cariera viitoare.

Calculați noile totaluri și treceți-le în tabelul următor.

Total	Abilitatea	Clasamentul abilităților preferate
	Abilități artistice	
	Abilități de comunicare	
	Abilități interpersonale	
	Abilități de conducere	
	Abilități matematice	
	Abilități practice	
	Abilități științifice	

## I.5.2. Identificarea meseriilor de interes

Este momentul să începeți munca de explorare. E timpul să adunați informații din cât mai multe surse posibile, și să aflați denumirile posibilelor meserii, care sunt cerințele pentru aceste meserii și orice alte informații de acest gen.

Care sunt sursele de informare? Există multe posibilități de informare, precum:

- ✓ **Ziare și site-uri de internet** – anunțurile de angajare din ziare și de pe internet pot fi o sursă importantă de informare. În acest mod, puteți afla care sunt cele mai cerute meserii de pe piață, care sunt cerințele pe care angajatorii le solicită viitorilor angajați etc. Nu uitați însă că ceea ce este acum "la modă" pe piață s-ar putea ca în 4-5 ani, când probabil veți intra pe piața muncii, să nu mai fie tot atât de cerute.
- ✓ **Agențiile de plasare a forței de muncă** – angajații acestor agenții vă pot consilia în căutarea unui loc de muncă potrivit calificărilor și deprinderilor voastre. Tot aici puteți afla informații utile despre tendința de pe piață, despre cererea de forță de muncă.

- ✓ **Persoanele care lucrează în domeniu** – probabil cele mai bune informații despre o meserie pot fi aflate de la persoane care au această meserie. Veți afla de la aceștia nu doar cerințele meseriei, dar puteți afla multe date despre mediul în care lucrează, despre programul de lucru, și multe alte aspecte utile în luarea unei decizii. De multe ori ceea ce ne imaginăm noi că înseamnă o meserie nu se potrivește deloc cu realitatea. Dacă putem intra în contact direct cu oamenii care lucrează în domeniu, dacă putem vizita locul acestora de muncă ne pot fi de mare ajutor.
- ✓ **Angajarea temporară pe timpul vacanțelor școlare**, în domenii cât mai apropiate de meseriile pe care le doriți, ar putea să vă ajute să vă dați seama mai bine dacă aceste meserii vi se potrivesc cu adevărat. În plus experiența acumulată în acest fel poate fi foarte valoroasă mai târziu, atunci când veți scrie un curriculum vitae. Aici putem include și munca de voluntariat, care este de multe ori privită cu ochi foarte buni de către angajatori.



#### **Exercițiul 4**

1. Folosind orice sursă de informație de care dispuneți, căutați **5** meserii din domeniul IT care v-ar putea trezi interesul și scrieți denumirile acestor meserii pe caiet sau pe o foaie de hârtie.
2. Pentru una dintre aceste meserii căutați o descriere cât mai detaliată a sa. Notați **5** calități cerute de această meserie.
3. Care sunt sarcinile și responsabilitățile meseriei selectate?
4. Care este nivelul de pregătire necesar pentru meseria selectată la punctul 2?
5. Care sunt condițiile de muncă ale acestei meserii (număr de ore zilnice de muncă, cerințe fizice ale postului, nivelul de stres, etc)?
6. Care sunt posibilitățile de avansare în carieră dintr-un astfel de post?

Puteți folosi pentru a rezolva această sarcină de lucru următoarele adrese de internet:

[http://www23.hrdc-drhc.gc.ca/ch/e/docs/ch\\_welcome.asp](http://www23.hrdc-drhc.gc.ca/ch/e/docs/ch_welcome.asp)

<http://hotjobs.yahoo.com>

<http://www.careeroink.com/career-reference/>

<http://www.go.ise.ro/>

<http://www.munca.ro/>

<http://www.bls.gov/k12/>

### I.5.3. Evaluarea posibilelor cariere

Ați reușit probabil să vă faceți o idee despre carierele din domeniul IT pe care le-ați putea urma și care sunt cerințele pentru fiecare dintre acestea. Pe de altă parte, ați realizat un inventar a ceea ce știți și doriți să faceți. Acum veți afla cum să comparați ceea ce oferiți voi și ceea ce vi se cere pe piața muncii. Veți învăța cum să vă dați seama dacă o meserie este potrivită pentru voi.

Înainte de a trece la treabă dorim să precizăm încă câțiva factori de care trebuie să țineți cont în alegerea unei meserii. Nu este suficient să fiți bine pregătit pentru o meserie, să aveți destule calități pentru aceasta, să vă placă să folosiți abilitățile solicitate. Trebuie să luați în calcul și următorii patru factori importanți:

- **Tendința pieței muncii** – diferitele fenomene economice și sociale, pot influența în timp cererea de forță de muncă. De exemplu, dacă înainte de 1989, inginerii erau la mare căutare, imediat după revoluție aceștia au intrat oarecum în umbră, mulți dintre ei fiind nevoiți să se reprofileze. Deși acum e foarte căutată o anumită meserie, peste câțiva ani s-ar putea să fie dificil să găsești un loc de muncă în meseria respectivă. De aceea când alegi o meserie, trebuie să studiezi statisticile și să afli care vor fi probabil meseriile căutate peste câțiva ani.
- **Lucrul în aer liber** – unele persoane preferă să lucreze în aer liber, alții dimpotrivă preferă munca în interior, ca de exemplu într-un birou, în spital, laborator, etc.
- **Munca fizică** – vă place și puteți desfășura muncă fizică grea?
- **Condiții periculoase de muncă** – mulți oameni suferă accidente la locul de muncă. Multe meserii presupun un anumit grad de pericolozitate. Întrebarea este dacă sunteți dispuși să vă asumați astfel de riscuri sau preferați un loc de muncă cu condiții sigure?



#### Exercițiul 5

Reveniți la tabelul obținut la exercițiul 3. Căutați în tabelul următor coloanele corespunzătoare abilităților aflate pe primul și pe al doilea loc în clasamentul abilităților voastre preferate și bifați acele meserii pentru care există un cerculeț umplut (●) în aceste două coloane. Aceste ocupații folosesc într-o foarte mare măsură ambele abilități. Am inclus în acest tabel doar meseriile din domeniul IT, sau domenii în care utilizarea calculatorului este importantă.

Parcurgeți din nou tabelul și urmăriți coloana corespunzătoare celei de a treia abilitate preferată de voi, conform exercițiului 3. Dacă **meseriile pe care le-ați marcat deja** au un cerculeț umplut (●) sau un cerculeț cu punct (⊙) în acea coloană, mai faceți încă o bifă în dreptul meseriilor respective. Dacă nici una dintre meseriile deja marcate nu au unul din semnele ● sau ⊙, căutați cerculețele goale (○) din coloana respectivă.

	Abilități personale							Caracteristici			
	Abilități artistice	Abilități de comunicare	Abilități interpersonale	Abilități de conducere	Abilități matematice	Abilități practice	Abilități științifice	Tendința pieței muncii	Muncă în aer liber	Muncă fizică	Condiții periculoase
Manager IT		⊙	⊙	⊙	●	●	●	○			
Administrator baze de date și sistem		⊙	○	○	●		●	○			
Inginer software		⊙	⊙	⊙	●	●	●	○			
Specialist suport IT și administrator de sistem		⊙	⊙	○	⊙	●	⊙	○			
Analist de sistem		⊙	○	○	●		●	○			
Desenator (grafician)	●	⊙	⊙	⊙				○		○	
Contabil sau auditor		○	○	⊙	●			○			
Inginer	⊙	⊙	○	⊙	●	●	●	⊙	○	○	○
Analist financiar și consilier financiar personal		●	●	○	●		○	●			
Manager financiar		●	●	●	●		○	⊙			
Analist managerial		●	⊙	○				○			
Administrator sau manager de birou		●	●	●	⊙	○		○			
Secretar		○	⊙		○			⊙			
Asistent secretar	○	●	●	○	○			○			
Profesor asistent	⊙	●	●	⊙	○		⊙				
Profesor	⊙	●	●	●	●			⊙			
Scriitor și editor	●	●	●	○	○		○				

Revedeți acum lista meseriilor marcate și vedeți dacă acele caracteristici speciale se potrivesc preferințelor voastre. În cele patru coloane corespunzătoare caracteristicilor speciale semnificația semnelor este următoarea:

- *Tendința pieței muncii* – dacă apare un cerculeț plin, meseria respectivă este foarte puternic influențată de fenomenele economice și sociale, dacă apare un cerculeț cu punct, meseria este într-o oarecare măsură influențată de aceste fenomene, cerculețul gol semnifică o mică influență a acestor fenomene, iar lipsa oricărui semn înseamnă că meseria este foarte stabilă, neinfluențată de fenomenele economice.
- *Munca în aer liber*
  - ● - meseria solicită muncă în aer liber
  - ⊙ - meseria necesită în oarecare măsură muncă în aer liber
  - ○ - meseria implică într-o mică măsură muncă în aer liber

- *Munca fizică*
  - ● - meseria solicită muncă fizică
  - ☉ - meseria necesită în oarecare măsură muncă fizică
  - ○ - meseria implică într-o mică măsură muncă fizică
- *Condiții periculoase de muncă*
  - ● - meseria implică condiții periculoase
  - ☉ - meseria implică într-o oarecare măsură condiții periculoase
  - ○ - meseria implică într-o mică măsură condiții periculoase

Notați în caiete meseriile care au rămas selectate în urma acestor selecții

## I.5.4. Scrisoarea de intenție

Scrisoarea de intenție este prima modalitate de a vă prezenta, de a vă face cunoscute "punctele forte" care vă fac un candidat favorit pentru jobul solicitat. Scrisoarea de intenție trebuie să conțină motivația, calificările și aptitudinile voastre și, nu în ultimul rând, trebuie să exprime disponibilitățile voastre față de compania la care intenționați să vă angajați. De regulă, este prima șansă de a face o impresie bună, iar o scrisoare concepută exclusiv pentru firma respectivă arată interesul deosebit pe care îl acordați acesteia.

CV-ul dvs. poate da o mulțime de informații despre voi, însă scrisoarea de intenție trebuie să-l determine pe cititor să se gândească un minut în plus dacă să vă aleagă pe voi și nu pe oricare alt candidat.

În acest sens, scrisoarea de intenție trebuie să conțină neapărat solicitarea efectivă a postului pe care îl doriți. Pentru aceasta, explicați în cuvinte puține și simple motivația pentru care doriți postul respectiv, care ar fi principalele calități ale dvs. care vă recomandă pentru ocuparea postului respectiv, ce doriți să realizați în cadrul firmei. Trebuie, de asemenea, să menționați ce anume din activitatea firmei (eventual prestigiul acesteia) va determinat să solicitați respectivul post.

Scrisoarea de intenție nu are un conținut standard, ci în general, trebuie să exprime interesul candidatului pentru postul vizat. Scrisoarea va fi adresată persoanei care se ocupă de angajări sau direct departamentului de resurse umane, dacă nu aveți informații complete. Dacă firma este mică, scrisoarea poate fi trimisă direct către manager (executiv sau general) ori președintele acesteia. Menționați disponibilitatea pentru un interviu de angajare, cât și faptul că respectiva scrisoare este însoțită de un CV.

Iată un model de scrisoare de intenție, potrivit unui proaspăt absolvent:

Ioana Popescu  
Str. Frunzelor, nr. 5  
Cluj, 73546  
0721 XXXXXX

D-I Ionescu Ion  
Firma...  
Str. Lalelelor nr.26  
Bucuresti, 35647  
Cluj, 5 iulie 2005

**Stimate domnule Ionescu,**

Personalitatea mea comunicativă și sociabilă, experiența în vânzări și studiile mele recent finalizate sunt argumente solide pentru candidatura mea la o poziție de broker de asigurări pentru [ Firma].

Am absolvit recent Universitatea [...] și am obținut o diplomă în [domeniul]. Pe durata ultimilor ani de studii am fost președintele / vicepreședintele / membrul asociației studențești [...] și am publicat lucrări de specialitate în [revista].

Deși abia am absolvit facultatea, nu sunt un proaspăt absolvent tipic. În anul trei am obținut o bursă Socrates/Leonardo la Universitatea [...] din [...]. În perioada studenției am avut propriile mele surse de venit, prin intermediul mai multor joburi: am făcut reclame la radio și sampling, am vândut abonamente de ziare. Toate acestea m-au ajutat să-mi completez pregătirea formală oferită de universitate.

Consider că am maturitatea, abilitățile și calitățile necesare pentru a începe o carieră în brokerajul de asigurări și mi-ar plăcea să lucrez în orașul meu natal, Cluj, unde știu că aveți deschisă o filială.

La sfârșitul lunii viitoare voi călători în București și mi-aș dori foarte mult să pot să vă întâlnesc pentru a discuta despre posibilitatea de a ocupa o poziție în cadrul firmei dumneavoastră. În continuarea acestei scrisori vă voi suna pentru a vedea dacă putem stabili o întâlnire.

În CV-ul care însoțește această scrisoare sunt specificate detalii legate de abilitățile mele profesionale.

Vă mulțumesc pentru timpul și atenția dumneavoastră.

Cu stimă,  
Ioana Popescu  
[semnătura]



## I.5.5. Curriculum vitae

Scrierea unui curriculum vitae poate fi o muncă dificilă pentru cei care nu sunt familiarizați cu acest tip de document, dar există o serie de tehnici care fac această muncă mai ușoară.

Scopul acestui document este să convingă angajatorul că ar trebui să te invite la un interviu. Este într-un anumit sens o formă de publicitate, prin care încerci să atragi atenția, să stârnești interesul, să descrii abilitățile și realizările tale cele mai deosebite și să inviți angajatorul să te contacteze.

E foarte important ca un curriculum vitae să fie cât mai concis, angajatorul nu va petrece foarte multă vreme citindu-l. Acest document va arăta angajatorului cine ești, ce știi, ce poți face și care sunt realizările tale.

Prima impresie lăsată de către curriculum vitae este foarte importantă. El trebuie să fie bine organizat, ușor de citit (font Arial sau Times New Roman, mărime 12 pct), corecte din punct de vedere gramatical.

Dacă adaugi și date calendaristice, ordinea acestora va fi cea ***invers cronologică***.

Secțiunile importante ale unui curriculum vitae sunt:

- ✓ **Date personale** – nume, adresă, număr de telefon, adresă de e-mail, adresa paginii de web dacă există.  
  
***Observații:*** dacă nu aveți o adresă de e-mail este momentul să vă creați una acum. Aveți însă grijă la id-ul ales. Acesta trebuie să fie unul serios, profesional.
- ✓ **Obiectivul de carieră/obiectivul pentru poziția solicitată**
- ✓ **Abilități speciale și calificări corespunzătoare obiectivului ales**
  - referiți-vă aici la abilitățile descoperite în prima secțiune a acestui capitol
  - Limbi străine cunoscute
  - Abilități de lucru cu calculatorul
  - Certificări obținute
  - Dacă nu ai experiență relevantă, accentuează aptitudinile pe care le-ai dezvoltat în termeni de relații interpersonale, organizaționale, etc., cunoștințele relaționate la aspectele postului la care aspiři.
- ✓ **Educație** – vei menționa aici școlile și calificările obținute. Nu este necesar să treci în CV școala primară și gimnaziul absolvit, decât dacă au fost institutii prestigioase. Menționează școala absolvită, diploma obținută, data și, dacă dorești, poți scrie și specializarea obținută (ex. București, Diploma

Bacalaureat, Liceul "Ion Neculce", 1999). Numele diplomelor obținute nu se abreviază. Toate etapele educaționale vor fi scrise în ordine inversă absolvirii lor (cea mai recentă fiind prima).

O sub-secțiune este cea a cursurilor care au relevanță în raport cu locul de muncă solicitat.

Proiecte - în această sub-secțiune pot fi scrise proiectele relevante, la care ai participat.

✓ **Experiență IT&C** - vor fi menționate cunoștințele teoretice, dar și cele practice.

✓ **Experiența profesională** - posturi ocupate până în momentul de față, poziția ocupată, perioada, responsabilități pe care le-ai avut.

Elevii/studenții vor menționa aici cercetări la care au participat, certificări obținute, premii și burse obținute (vor menționa numele premiului și numele instituției care a acordat premiul și data).

✓ **Activități de voluntariat, implicare în cadrul comunității, asociații la care sunteți membri**

✓ **Articole/cărți publicate**

✓ **Alte activități și interese** – activități extracurriculare, hobbyuri etc.

Din ce în ce mai mult în întreaga Europă, dar și la noi în țară este folosit formatul European al CV-ului. Formatul european este utilizat, în mod special, pentru locuri de muncă cu standarde superioare pentru angajați.



### **Exercițiul 6**

Redactați CV-ul personal, folosind formatul european pe care îl puteți descărca de pe internet de la adresa [http://www.go.ise.ro/uchazeni/cv\\_eu.rtf](http://www.go.ise.ro/uchazeni/cv_eu.rtf).



### **Exercițiul 7**

Descoperiți greșelile tipice în redactarea unui CV, folosind următoarele adrese de internet:

<http://www.cdm.uwaterloo.ca/retest.asp>

<http://www.cdm.uwaterloo.ca/retest2.asp>

Încercați să depistați și să corectați în CV-ul vostru eventualele greșeli, după modelul anterior.

## I.5.6. Pregătirea și susținerea interviului

Dacă ai fost invitat pentru un interviu, acesta reprezintă un prim succes. CV-ul pe care l-ai trimis a stârnit interesul instituției/firmei respective pentru persoana ta. Dar, de acum începe o altă competiție pe care, de asemenea, trebuie să o câștigi.

Prima condiție pentru a te prezenta cât mai bine este să nu te gândești la ceilalți candidați, oricât de mulți ar fi aceștia. Poți fi singurul care participă la interviu sau poți fi unul din mai mulți candidați și să câștigi.

Decizia celui care te interviează va fi luată numai în funcție de comportamentul tău.

Pregătirea pentru interviu constă în a învăța să porți o conversație cu angajatorul și să răspunzi la anumite întrebări. Scopul tău este să-l convingi pe acesta că tu ești persoana cea mai potrivită pentru postul respectiv, că ai abilitățile și deprinderile necesare.

Interviul este binevenit. Este ocazia unică să te întâlnești cu angajatorul și să arăți cine ești și ce vrei. Bineînțeles, trebuie să te pregătești pentru acest interviu, să adopți o strategie.

Încearcă să te gândești că interviul este de fapt o conversație între doi parteneri egal interesați de această conversație. Amândoi aveți informații importante de expus și de aflat.

Iată câteva sugestii de care ar fi bine să ții cont în pregătirea pentru interviu:

- Adoptă o ținută corespunzătoare, curată și îngrijită. Statisticile spun că 70% din mesajul pe care îl transmiți este exclusiv vizual. Alege să transmiți un mesaj elegant prin ținuta ta.

Pentru femei, cea mai potrivită este o ținută clasică, sobră, simplă dar de bun-gust. Taiorul și fusta sau pantalonul pot fi o variantă adecvată de îmbrăcăminte pentru interviu. Nu exagera cu accesoriile. Nu purta lanțușoare sau brățări zgomotoase, încărcate de pietre prețioase sau tot felul de medalioane. Încearcă să fii cât mai discretă, dar cu haine de calitate.

Evită aspectul neîngrijit, cu pete, scame sau ațe atârând din haină, poartă o geantă de proporții medii, eventual ia la interviu o servietă sau o mapă.

Nu abuzați cu machiajul și folosiți doar o bază care să vă acopere micile imperfecțiuni, nu folosiți culori tari la fardul de pleoape sau de obraz. Rujul trebuie să fie cât mai discret, cu un contur foarte fin.

Coafura trebuie să fie și ea simplă, cu părul ridicat, astfel încât să lase la vedere fața. Cu alte cuvinte, maschează defectele, dar menține atenția auditoriului la ceea ce ai de spus, nu la felul în care arăți.

Bărbații pot opta oricând pentru un costum clasic, bine croit, evitând, pe cât se poate culorile închise precum negru sau bleu-marin. Indiferent cât de cald ar fi afară, prezintă-te la interviu cu o cămașă cu mânecă lungă și cravată.

Folosește un parfum discret, fii atent(ă) la manichiură. Nu arunca servieta pe jos, aranjează-ți lucrurile cu grijă și angajatorii vor înțelege prin ținuta pe care o afișezi în timpul interviului că ești o persoană ordonată.

- Dezvoltă-ți capacitatea de comunicare.
- Pregătește pentru interviu un dosar pe care să îl ai la tine și care să conțină:
  - Date despre companie (rapoarte anuale, materiale privind vânzările, etc.).
  - Câteva copii suplimentare ale CV-ului și ale scrisorilor de recomandare.
  - O listă cu 10-15 întrebări pe care le poți pune în timpul interviului și care se bazează pe datele pe care le ai despre companie.
  - Pix, foi și orice alte lucruri care crezi că îți pot fi utile (acte de studii, foi matricole, etc.).
- În ziua dinaintea interviului sună la firmă pentru a confirma întâlnirea. Fii sigur că știi unde trebuie să ajungi și fă în așa fel încât să ajungi cu 10 minute înainte de ora fixată. Ia în calcul orice situație care poate duce la o eventuală întârziere și evit-o. Încearcă să nu ajungi în ultimul moment, agitat și transpirat.
- Verifică-ți ținuta înainte de interviu. Verifică-ți hainele, imaginea generală, aranjează-ți părul. Exersează zâmbetul!
- Ai grijă la limbajul corpului. Acesta poate da indicii cu privire la starea noastră de spirit. Nu evita contactul vizual cu interviatorul, dar nu îl fixa cu privirea. Contactul vizual e una dintre cele mai puternice forme de comunicare, dovedind încredere și putere. Nu fă mișcări bruște, nu-ți încrucișa brațele. Nu acoperi fața cu mâinile, ține-ți mâinile departe de față.
- Comportați-vă într-o manieră încrezătoare, dar nu sfidătoare. Nu evitați privirea nimănui, răspunsurile nu trebuie șoptite și nici mormăite, iar când este cazul, zâmbiți. Vorbiți la obiect, nu încercați să fiți prea spiritual sau jovial, dar și a fi prea retras poate fi o greșeală.
- Extrem de utilă este capacitatea de a asculta. Ofertele de muncă sunt pentru cei care știu să asculte, să găsească înțelesuri ascunse și să răspundă la întrebări pe scurt într-un mod convingător.
- Demonstrează entuziasm și interes sincer.

- Subliniază-ți abilitățile și capacitățile. Descrie valoarea ta și beneficiile pe care le oferi. Arată cum poți tu contribui la 1) creșterea vânzărilor, 2) reducerea costurilor, 3) îmbunătățirea productivității, 4) rezolvarea problemelor organizatorice.
- Ia notițe în timpul interviului. Te poți folosi de aceste notițe mai târziu, în timpul interviului. Dacă nu ești sigur poți cere permisiunea mai întâi.
- Lasă interviuatorul să aducă în discuție problema salariului. Odată propusă o sumă, aceasta poate fi negociată.
- Fi pregătit să răspunzi la toate întrebările, chiar și la cele incomode. Înainte de interviu, pregătește un răspuns pentru fiecare întrebare care ar putea fi o problemă pentru tine și exersează acest răspuns până vei fi sigur pe el. Majoritatea întrebărilor care pot apărea în cadrul unui interviu pot fi prevăzute. În consecință, poți să-ți pregătești din timp răspunsurile.
- Întotdeauna, în următoarele 24 de ore de după interviu, trimite o scrisoare de mulțumire. Aceasta te poate detașa de ceilalți candidați.

## Exemple de întrebări frecvente în interviurile pentru angajare

- ✓ Cum ați descrie cariera dumneavoastră de până acum?
- ✓ Care era activitatea dvs. la locul de muncă anterior?
- ✓ Ce vă plăcea (nu vă plăcea) la această activitate?
- ✓ Care au fost elementele noi pe care le-ați învățat acolo?
- ✓ De ce ați părăsit locul de muncă anterior? (atenție - nu fiți prea critic la adresa fostului șef)
- ✓ Descrie-te pe tine însuși.
- ✓ De ce te interesează locul acesta de muncă?
- ✓ Cum te-ar descrie prietenii tăi?
- ✓ Ați mai fost și la alte interviuri?
- ✓ Ce știți despre firma noastră? De ce ați ales-o? (merită să știți ceva despre firmă – informați-vă din timp)
- ✓ Ce v-ar plăcea să obțineți în muncă (în carieră)? Cum vă vedeți (ce v-ar plăcea să faceți) peste 5 (10) ani?

- ✓ Care vă sunt punctele forte (slabe), la ce sunteți bun (mai puțin bun)?
- ✓ Care sunt cunoștințele și aptitudinile pe care le-ați aplica la noul loc de muncă?
- ✓ De ce ar trebui să vă încredințăm dvs. acest loc de muncă? Ce beneficiu ne-ați aduce?
- ✓ Ce tip de muncă vă face să fiți încrezător?
- ✓ Sunteți dispus să lucrați peste programul normal?
- ✓ Sunteți capabil și dispus să lucrați ocazional cu un volum mare de muncă?
- ✓ Știți să vă odihniți sau să vă relaxați? Cum vă petreceți vacanțele? Care sunt interesele dumneavoastră de timp liber? Vă place sportul?
- ✓ Care este starea sănătății dumneavoastră?

***Fii tu însuți !***

***Fii cinstit !***

***Fii pozitiv !***

1. Proiectarea bazelor de date. Noțiuni introductive
2. Normalizarea datelor
3. Implementarea modelului conceptual
4. Elemente avansate de proiectare a bazelor de date
5. Dezvoltarea profesională în domeniul IT
6. Managementul de proiect

În acest capitol veți afla:

- ✓ care sunt etapele în realizarea unui proiect
- ✓ care sunt avantajele lucrului în echipă
- ✓ care sunt regulile pe care trebuie să le aibă în vedere un bun lider de echipă
- ✓ cum să pregătiți și să susțineți o prezentare publică

## I.6.1. Ce este un proiect ?

Un proiect este o secvență de acțiuni intercorelate, ce se derulează într-o perioadă de timp clar definită și delimitată, acțiuni orientate către îndeplinirea unor obiective cu caracter unic și precis.

Putem defini managementul de proiect ca fiind efortul planificării, organizării și mobilizării resurselor pentru un scop dat. Managementul de proiect este de fapt un instrument, un set de metode și tehnici care ne ajută să atingem cu eficacitate scopurile și obiectivele propuse.

## I.6.2. Etape în realizarea unui proiect

Orice proiect trece printr-o serie întreagă de etape. Ceea ce este important de reținut este faptul că orice proiect este un proces iterativ, în sensul că orice etapă poate fi repetată de mai multe ori, în funcție de necesitățile de redefinire a anumitor cerințe.

Etapetele principale pe care le parcurgem pentru realizarea unui proiect sunt următoarele:

- Definirea proiectului
  - Validarea proiectului – în această etapă vor fi analizate toate documentele prezente în propunerea de proiect. Această analiză va duce fie la confirmarea și acceptarea proiectului, fie va duce la respingerea, sau eventual regândirea acestuia, în cazul în care propunerea de proiect nu a fost bine realizată, și resursele au fost subestimate.
  - Definirea proiectului – constă în enunțarea problemei, stabilirea scopului proiectului, stabilirea unei liste a posibilelor soluții etc. Orice proiect propus poate avea elemente pe care cel care l-a propus nu a considerat necesar să le detalieze, însă echipa de proiect poate avea nevoie de informații suplimentare pentru a înțelege corect enunțul proiectului.
  - Identificarea surselor de finanțare a proiectului
  - Obținerea aprobărilor de realizare a proiectului
- Organizarea proiectului
  - Stabilirea obiectivelor



- Stabilirea grupului țintă - când încercați să stabiliți care este grupul țintă al unui proiect trebuie să răspundeți la următoarele întrebări: Cine trebuie să știe despre proiect? Cine va folosi acest proiect? Asupra cui vor avea impact rezultatele proiectului? Cine finanțează acest proiect? Cine aprobă acest proiect? Cine livrează proiectul? Cine va trebui să fie instruit?
- Stabilirea cerințelor proiectului – este un proces iterativ care poate implica negocierea. Fiind conducătorul proiectului, știți ce este posibil și ce nu în ceea ce privește scopul proiectului, timpul alocat, costurile, cerințele de calitate etc. Un proiect care nu îndeplinește cerințele utilizatorului este din start un eșec. Asigurați-vă că în urma negocierilor obțineți acordul utilizatorului.
- Stabilirea infrastructurii proiectului – infrastructura se referă la elemente precum instrumente de comunicare în cadrul echipei (telefoane mobile, PDA, laptop, etc.), spațiu de lucru pentru membrii echipei (birou), echipamente de birou, laboratoare, etc.
- Stabilirea sistemului de calitate a proiectului
- Formarea echipei de proiect
- Planificarea proiectului – planul proiectului prezintă desfășurarea normală, ideală a unui proiect. Acesta oferă reperele necesare evaluării situației proiectului. Fără existența unui plan dinainte stabilit, exercitarea controlului este practic imposibilă. Se iau decizii cu privire la elemente cheie ale proiectului cum ar fi: obiectivele, activitățile, resursele și implementarea proiectului.
- Elaborarea proiectului – în această etapă se stabilesc detaliile tehnice și de design, se revizuiesc schemele tehnice, se revizuiesc criteriile de cost și performanță. Această etapă vizează conturarea unui model al aplicației. Se proiectează bazele de date, se realizează interfețele aplicației (rapoarte, ecrane, meniuri etc), se proiectează prelucrările automate etc.
- Dezvoltarea proiectului
- Implementarea proiectului – constă în instalarea produsului și testarea sa, instruirea viitorilor utilizatori. Se testează produsul în condiții reale, și se înlătură eventualele erori depistate în funcționare. Tot în această fază, personalul implicat în proiect este redus. Se încep activitățile de publicitate.
- Finalizarea proiectului – se elaborează rapoartele finale, se eliberează personalul angajat în derularea proiectului.

### I.6.3. Principiile lucrului în echipă

Colaborarea înseamnă mai ales implicarea oamenilor în crearea propriilor soluții la problemele cu care se confruntă. În cadrul unui proces colaborativ, oamenii au ocazia să afle punctul de vedere și perspectiva celorlalți și să gândească împreună soluții la problemele comune.

Colaborarea este o artă. Sunteți uneori puși în situația de a vorbi și colabora cu persoane pe care nu le cunoașteți, cu care nu sunteți de acord, sau pe care nu le agreeați.

Mărimea echipei de proiect este un factor important care poate influența modul de conducere al acesteia. Într-o echipă prea mare pot apărea probleme de comunicare, dificultăți în luarea deciziilor. În general, se acceptă că o echipă, pentru a putea fi eficientă, nu trebuie să depășească un număr de 10 persoane.

În cazul proiectelor cu număr mare de persoane implicate, participanții la proiect trebuie organizați în mai multe echipe satelit intercorelate, coordonate de o echipă principală al cărei rol principal este să asigure o comunicare facilă și eficientă între toate echipele satelit.

Fiecare membru al unei echipe de proiect trebuie să cunoască ceea ce se așteaptă de la el. Trebuie stabilite de la început rezultatele așteptate de la fiecare membru al echipei în parte, acțiunile ce trebuie executate pentru obținerea rezultatului, relațiile de subordonare în cadrul echipei, metodele de măsurare a performanței fiecărui membru al echipei.

Liderul echipei are în primul rând rolul de a stabili, menține și proteja un proces colaborativ care permite tuturor să participe neîngrădit la munca grupului. Pentru aceasta el poate realiza câteva acțiuni importante:

- să creeze o viziune, să identifice scopurile pe care vrea ca echipa să le atingă. Să definească modul în care dorește ca echipa să fie percepută de persoanele din afara ei.
- să ajute oamenii să se simtă în largul lor venind spre el cu probleme. O comunicare proastă poate constitui sentința de moarte a unei echipe. Chiar și cel mai bun lider de echipă nu poate corecta o problemă despre existența căreia nu știe.
- să conducă prin exemplu. Liderul nu poate determina o echipă să aibă o motivație pozitivă pentru munca lor dacă el nu are o astfel de motivație.
- să ajute grupul să stabilească un set de reguli de comportament acceptat și respectat de toată lumea, și care să încurajeze respectul reciproc, participarea și încrederea.
- să încurajeze participarea largă.

- să medieze conflictele și disputele.
- să mențină abordarea colaborativă în rezolvarea problemelor și luarea deciziilor, să se asigure că un individ, sau subgrup nu vor acționa fără acordul grupului mai mare din care fac parte.
- să ajute grupul să identifice și să obțină resursele necesare pentru a-și putea îndeplini sarcinile propuse.

## I.6.4. Pregătirea și susținerea unei prezentări

În momentul finalizării unui proiect va trebui să prezentați rezultatul muncii voastre în fața clientului. Susținerea unei prezentări în public este pentru oricine o sursă de stres. Mulți oameni ar prefera să evite complet această problemă, dar de multe ori este imposibil de evitat. Indiferent că lucram singuri sau în echipă, vom putea fi puși în situația de a face o prezentare publică.

Adevărul este că susținerea unei prezentări publice, nu trebuie să fie un motiv de stres. Atât timp cât aveți în minte câteva principii de bază, vorbitul în public va deveni o experiență plăcută pentru voi.

Iată câteva principii de care trebuie să țineți seama pentru a depăși dificultățile legate de vorbitul în public:

- Vorbitul în public NU este un motiv real de stres.  
Mulți oameni erau la început îngroziți de ideea de a vorbi în public. Le tremurau genunchii, vocea, gândurile o luau razna. Totuși ei au învățat să elimine definitiv teama de a vorbi în public.
- Nu îți propune să fi genial sau perfect, chiar dacă așa ți se pare că ar trebui. Ți se poate întâmpla să greșești, ți se poate întâmpla să uiți porțiuni întregi din ceea ce vroiai să spui, poți să nu glumești deloc și totuși prezentarea ta să aibă succes.  
Totul depinde de cum definești tu și auditorul tău succesul. Auditorul tău nu așteaptă de la tine să fii perfect. Cu cât încerci mai tare să fii perfect cu atât mai mult vei adânci anxietatea pe care o ai și efectul va fi contrar celui scontat. Ceea ce este important atunci când faci o prezentare în fața unui public este să oferi ceva auditorului. Atât timp cât auditorul rămâne cu ceva de pe urma prezentării, ei o vor considera un succes.
- Nu încerca să transmiți prea multe informații într-o prezentare. Nu trebuie să transmiți "tone" de informații și detalii, oferă publicului doar ceea ce el dorește cu adevărat. Studiile arată că oamenii își amintesc doar foarte puține elemente din ceea ce vorbitorii prezintă.

- Nu încerca să mulțumești pe toată lumea, acesta este un punct de vedere nerealist.
- Nu încerca să imiți alți prezentatori, e un lucru foarte dificil. Încearcă să fii tu însuși, e mult mai ușor.
- Nu încerca să controlezi comportamentul auditorului. Dacă oamenii sunt agitați, nu încerca să le controlezi comportamentul. Dacă cineva vorbește cu vecinul, sau citește ziarul sau chiar a adormit în timpul prezentării tale, ignoră-l.
- Nu te scuza. Auditorul nu are de unde să știe că ai uitat să spui ceva. Vorbește cu încredere, fii convins de ceea ce spui.
- Organizează logic conținutul prezentării. Orice prezentare va avea o introducere, conținutul propriu-zis și o concluzie.
- Utilizează materiale audio-vizuale ajutătoare, dacă acestea sunt necesare. O prezentare Power-Point, bine realizată, poate fi de un real folos.
  - Asigură-te din timp că întregul echipament (calculator, video-proiector etc) funcționează.
  - Nu bombarda auditorul cu efecte sonore, cu animații exagerate, utilizează în mod echilibrat culorile.
  - Include în prezentarea electronică doar cele mai importante idei.
  - Nu încărca slideurile cu prea multe texte. În general, se admit 7-10 cuvinte pe linie și maximum 10 linii de text pe un slide.
  - Preferă un grafic în locul textelor, dar nu exagerați cu mai mult de două grafice pe un slide.
- Păstrează contactul vizual cu auditorul. Încearcă în acest fel să faci fiecare persoană din public să se simtă implicată.
- Fă scurte pauze. Nu alerga prin prezentare, dă-ți răgazul să respiri. Dă timp audienței să reflecte la ceea ce ai spus.
- Adaugă puțin umor prezentării tale, dacă acest lucru este posibil. Păstrează treaz interesul auditorului pe parcursul întregii prezentări.
- Nu ține mâinile în buzunar în timpul prezentării.
- Adoptă o ținută adecvată în ziua prezentării. Atrage atenția auditorului asupra a ceea ce spui, nu a mesajelor de pe tricou sau asupra bijuteriilor pe care le porți.



## Teme de proiect<sup>1</sup>

În acest moment v-ați însușit elementele de bază ale modelării bazelor de date. Ați văzut diverse situații ce pot apărea pe parcursul creării modelului conceptual și cum să rezolvați anumite situații problemă.

În acest capitol ați aflat cum se organizează un proiect, care sunt etapele de realizare a lui, cum să organizați munca într-o echipă. De asemenea știți acum cum să pregătiți și să faceți o prezentare în fața unui public.

Este momentul să aplicați toate aceste cunoștințe.

Formați echipe de 2-4 elevi, alegeți-vă o temă de proiect, fie din temele propuse în această secțiune, fie una propusă chiar de voi. Realizați modelul conceptual al afacerii modelate. Atenție! Repartizați sarcinile în mod echitabil în cadrul echipei de proiect.

Pregătiți o prezentare a proiectului realizat, materialele vizuale (postere, pliante, prezentarea PowerPoint), care să vă ajute la susținerea prezentării.

Puteți chiar organiza împreună cu cadrul didactic, un concurs la nivelul clasei, sau al școlii, la care să invitați și alți profesori din școală.

Nu uitați să documentați fiecare presupunere pe care ați făcut-o în realizarea proiectului.

**Tema 1: Campionatul Național de Fotbal.** O bază de date memorează informații despre jucătorii și cluburile din cele patru divizii din campionatul național de fotbal. Fiecare club de fotbal are un nume unic în întregul campionat. Un club de fotbal poate avea mai multe echipe în campionat. Pentru fiecare echipă se cunoaște căpitanul său, care este unul dintre jucători.

Jucătorii au atribuit un identificator unic, un nume, nu neapărat unic, și sunt angajați la diferitele echipe. În baza de date se păstrează și detalii privind nivelul abilităților (notă cuprinsă între 1 și 10) fiecărui jucător pentru fiecare dintre pozițiile de joc (portar, apărător, mijlocas, etc). De exemplu jucătorul Ionescu poate avea nivelul 10 pe postul de portar, 7 pentru poziția de apărător etc.

Este important ca în baza de date să se memoreze un istoric al tuturor jucătorilor, la ce echipe au jucat, în ce perioadă, etc.

Se va memora și un istoric al golurilor marcate de fiecare jucător de-a lungul carierei. Pentru fiecare gol se va ști data, meciul în care a fost marcat, minutul, etc.

---

<sup>1</sup> Temele proiectelor propuse aici sunt preluate și adaptate cu acordul domnului dr. Gordon Russell, <http://db.grussell.org>

**Tema 2: Firmă de închirieri de mașini.** Trebuie să proiectați baza de date a unei firme care oferă spre închiriere mașini de diferite tipuri. Trebuie să păstrați informații despre mașinile firmei, firmele cu care firma are contracte de colaborare (de exemplu garaje), mașinile închiriate, veniturile firmei și bineînțeles date despre clienții firmei.

Mașinile sunt descrise prin date precum: producător, model, anul de fabricație, mărimea motorului, tipul de combustibil, număr de pasageri, numărul de înmatriculare, prețul de cumpărare, data cumpărării, prețul de închiriere și detalii privind asigurarea mașinii. Toate reparațiile importante asupra mașinilor firmei sunt făcute de firme cu care colaborează. Unele firme solicită plata serviciilor de service imediat după ce reparația a fost făcută, altele acceptă plata în rate a serviciilor. Trebuie ținută evidența clară a fiecărei mașini, a celor închiriate, a celor aflate în reparații, etc.

Se păstrează de asemenea evidența tuturor veniturilor și cheltuielilor firmei: cumpărarea de noi mașini, închirierea de mașini, cheltuieli de reparații, vânzarea unor mașini mai vechi din parcul auto (firma preferă să nu păstreze în parcul auto o mașină mai mult de un an), taxe de asigurare a mașinilor, etc.

Firma deține un portofoliu de clienți destul de stabil. Pentru clienții privilegiați oferă reduceri la închirierea de mașini. Acești clienți au de asemenea posibilitatea de a rezerva o mașină din timp. O mașină poate fi închiriată pentru o perioadă de timp de la o zi la un an. Plata pentru închirierea unei mașini se poate face cash sau cu credit card. Se acceptă orice tip de card.

Datele importante despre clienți precum numele, adresa, numărul de telefon, seria permisului de conducere etc., vor fi și ele memorate în baza de date.

**Tema 3: Firmă IT.** Trebuie să proiectați baza de date a unei companii de dimensiune medie din domeniul IT. Firma livrează diferite produse clienților săi, de la simple aplicații create la cerere, până la instalări de echipamente hardware și software particularizat. Firma are ca angajați diverși experți, consultanți și personal auxiliar. Întregul personal este angajat pe termen nelimitat, nu există angajați temporari sau colaboratori.

Compania este împărțită în mai multe departamente, fiecare departament fiind condus de către un angajat din cadrul departamentului respectiv. Pentru un proiect care trebuie dezvoltat în cadrul firmei, se formează o echipă de persoane selectate din mai multe departamente. Managerul de proiect este pe deplin și exclusiv responsabil de conducerea proiectului, independent de ierarhia de conducere din cadrul firmei.

**Tema 4: Spital.** Un spital este format din mai multe secții, precum Pediatrie, Oncologie, Dermatologie etc. În fiecare secție sunt internați mai mulți pacienți, pe baza recomandării medicului de familie și a confirmării făcute de către un specialist al spitalului. La internare, sunt înregistrate datele personale ale pacienților. O fișă separată ține evidența investigațiilor făcute pacientului pe toată perioada internării, rezultatele acestor investigații, tratamentul aplicat pacientului și rezultatele obținute în urma tratamentelor efectuate. Un pacient este repartizat unui anumit medic care coordonează toate investigațiile și tratamentele aplicate pacientului, însă acesta poate solicita și altor colegi să examineze pacientul său.

Medicii sunt specialiști în diverse ramuri ale medicinei, și pot avea în supraveghere mai mulți pacienți, nu neapărat toți din aceeași secție.

**Tema 5: Editură.** O editură editează cărți științifice din diferite domenii. Cărțile sunt scrise de autori specializați într-un anumit domeniu. Firma are angajați mai mulți editori care nu sunt neapărat specialiști în diferitele domenii, fiecare editor fiind responsabil pentru mai multe publicații. O carte acoperă unul dintre domeniile în care este specialist autorul, fiecare autor lucrează cu un editor, dar poate avea spre publicare o altă carte de care este responsabil un alt editor.

**Tema 6: Firmă de înregistrări.** Notown Records a decis să memoreze informațiile despre muzicienii care cântă pe albumele înregistrate de firmă precum și alte date din interiorul firmei. Firma a decis să vă angajeze ca designeri ai bazei lor de date.

Despre fiecare muzician care înregistrează la Notown Records se cunosc cnp-ul, numele, adresa și numărul de telefon. Unii muzicieni aflați la început de carieră, având posibilități materiale scăzute, pot avea mai mulți o aceeași adresă (de exemplu mai mulți muzicieni închiriază împreună o locuință), și la nici o adresă nu există mai mult de un număr de telefon.

Fiecare instrument folosit la înregistrări, are un număr unic de identificare, un nume (de exemplu chitară, sintetizator, flaut) și o cheie muzicală.

Fiecare album înregistrat de Notown este etichetat cu un număr unic de identificare, un titlu, data copyright-ului, un format (CD, DVD, casetă audio sau video). Fiecare cântec înregistrat de Notown are un titlu și un autor.

Fiecare muzician poate cânta la mai multe instrumente, și pentru fiecare instrument pot exista mai mulți muzicieni care îl folosesc.

Fiecare album conține mai multe cântece, dar nici un cântec nu poate apărea pe mai mult de un album.

Bineînțeles că la înregistrarea unei melodii pot participa mai mulți muzicieni, și un muzician poate cânta mai multe melodii.

Fiecare album are un singur producător, care poate fi un muzician sau nu, iar un producător poate produce mai multe albume.

**Tema 7: Aeroport.** Clienții unui aeroport s-au plâns, în nenumărate rânduri, conducerii aeroportului despre proasta organizare a activității din aeroport. De aceea, conducerea acestuia a decis că toate informațiile legate de activitatea aeroportului trebuie să fie gestionate cu ajutorul unei baze de date și v-a angajat pe voi să proiectați această bază de date. Prima voastră sarcină este să organizați informațiile despre toate avioanele staționate sau deținute de către aeroport. În urma discuțiilor pe care le-ați purtat cu angajații aeroportului ați ajuns la concluzia că informațiile relevante sunt următoarele:

Fiecare avion are un număr de înregistrare, și un model. Aeroportul găzduiește un număr de modele de avioane, și fiecare model este identificat printr-un cod (de exemplu DC-10) și are o anumită capacitate și greutate.

La aeroport lucrează un număr de tehnicieni despre care trebuie să memorați numele, cnp-ul, adresa, numărul de telefon și salariul.

Fiecare tehnician este expert pentru un număr de modele de avioane.

Controlorii de trafic trebuie să aibă un control medical amănunțit. Pentru fiecare controlor de trafic trebuie să rețineți data celui mai recent control medical amănunțit.

Angajații firmei (inclusiv tehnicienii) pot fi membrii unuia dintre sindicatele existente. Pentru fiecare membru de sindicat trebuie să știți cărui sindicat îi aparține, data la care s-a înscris în sindicat, precum și un număr de legitimație.

Există o serie de teste care se aplică avioanelor pentru a se verifica starea lor tehnică și pentru a se emite autorizația de zbor pentru acel aparat. Fiecare test este identificat printr-un cod, un nume, și se cunoaște scorul maxim posibil și scorul minim necesar pentru ca autorizația de zbor să poată fi emisă.

Este nevoie să se memoreze data la care un anumit test a fost aplicat anumitei aeronave, scorul obținut, și trebuie să se știe care este tehnicianul care s-a ocupat de aplicarea testului respectiv și numărul de ore cât a durat testul.

**Tema 8: Farmacie.** Un mare lanț de farmacii v-a oferit un contract prin care se angajează să vă asigure orice medicament necesar dumneavoastră și familiei pe toată viața dacă le proiectați baza de date necesară. Date fiind costurile mari ale tratamentelor în caz de îmbolnăvire ați hotărât să acceptați contractul. Iată aici informațiile pe care le-ați obținut:

Pacienții care cumpără medicamente de la farmaciile firmei sunt identificați prin cnp și se memorează despre ei numele, vârsta și adresa.

Doctorii care emit rețete pe care farmaciile le eliberează sunt identificați de asemenea prin cnp, dar se memorează și numele, specialitatea, numărul anilor de experiență.

Fiecare farmacie a companiei are un nume propriu, o adresă și un număr de telefon.

Pentru fiecare medicament se cunoaște numele, care este unic și compoziția.

Fiecare farmacie vinde mai multe medicamente și are un preț pentru fiecare medicament. Un medicament poate fi vândut de mai multe farmacii și prețul poate varia de la o farmacie la alta.

Un medic poate prescrie mai multe medicamente unui pacient. Iar un pacient poate primi tratament de la mai mulți medici. Fiecare rețetă are înscrisă o dată la care a fost prescrisă și conține o listă a medicamentelor prescrise și cantitatea necesară.

Comaniile farmaceutice pot avea contracte cu farmaciile, iar o farmacie poate avea contract cu mai mult de o companie care livrează medicamentele. Pentru fiecare contract, trebuie să memorați data semnării contractului, data la care expiră contractul, și textul contractului.

Pentru fiecare companie farmaceutică se cunoaște numele, adresa, numărul de telefon și numele unei persoane de contact din cadrul firmei.



## **II. Programarea bazelor de date**





## Interogări simple. Sortarea datelor

II.1

### 1. Interogări simple. Sortarea datelor

2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea structurii tabelor.  
Constrângeri
7. Introducerea și actualizarea datelor din tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea drepturilor. Gestiunea tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ ce este SQL și care sunt categoriile de comenzi SQL existente
- ✓ cum să instalați, configurați și utilizați Oracle Database 10g Express Edition
- ✓ care sunt elementele de bază ale limbajului SQL
- ✓ care sunt operațiile ce se pot realiza cu comanda **SELECT**
- ✓ cum se scriu comenzile de interogare
- ✓ cum pot fi filtrate liniile care se vor afișa
- ✓ ce sunt și cum se folosesc alias-urile coloanelor
- ✓ cum se pot elimina liniile duplicate
- ✓ cum se pot sorta datele

## II.1.1. Noțiuni introductive

SQL (pronunțat fie ca un singur cuvânt “sequel” sau pe litere “S-Q-L”) se bazează pe studiile lui E.F. Codd, prima implementare a limbajului SQL fiind dezvoltată de către firma IBM la mijlocul anilor 1970. Mai târziu, compania Relational Software Inc. (cunoscută astăzi sub numele Oracle Corporation) a lansat prima versiune comercială de SQL. În prezent SQL este un limbaj complet standardizat, recunoscut de către Institutul Național American de Standarde (ANSI – American National Standards Institute). Puteți folosi SQL pentru a accesa baze de date Oracle, SQL Server, DB2, sau MySQL.

SQL utilizează o sintaxă simplă, ușor de învățat și utilizat. Comenzile SQL pot fi grupate în cinci categorii, după cum urmează:

- **Limbajul de interogare** permite regăsirea liniilor memorate în tabelele bazei de date. Vom scrie interogări folosind comanda **SELECT**.
- **Limbajul de manipulare a datelor (DML - Data Manipulation Language)** permite modificarea conținutului tabelelor. Există următoarele comenzi **DML**:
  - INSERT** - pentru adăugarea de noi linii într-o tabelă
  - UPDATE** - pentru modificarea valorilor memorate într-o tabelă
  - DELETE** - pentru ștergerea liniilor dintr-o tabelă.
- **Limbajul de definire a datelor (DDL - Data Definition Language)** vă permite să definiți structura tabelelor care compun baza de date. Comenzile din această grupă sunt:
  - CREATE** - vă permite să creați structurile bazei de date. De exemplu, **CREATE TABLE** este utilizată pentru crearea tabelelor, cu **CREATE USER**, puteți crea utilizatorii bazei de date, etc.
  - ALTER** - permite modificarea structurilor bazei de date. De exemplu, cu comanda **ALTER TABLE** puteți modifica structura unei tabele.
  - DROP** - puteți șterge structuri ale bazei de date. De exemplu, pentru a șterge o tabelă, folosiți comanda **DROP TABLE**.
  - RENAME** - puteți schimba numele unei tabele.
  - TRUNCATE** - vă permite să ștergeți întregul conținut al unei tabele.
- **Comenzi de control al tranzacțiilor (TC - Transaction Control):**
  - COMMIT** - vă permite să faceți ca modificările asupra bazei de date să devină permanente.
  - ROLLBACK** - permite renunțarea la ultimele modificări asupra bazei de date.

**SAVEPOINT** – vă permite să definiți un "punct de salvare" la care să puteți reveni, renunțând la modificările făcute după acel punct asupra bazei de date.

- **Limbaj de control al datelor (DCL - Data Control Language)** Permite definirea și modificarea drepturilor utilizatorilor asupra bazei de date. Există două comenzi în această categorie:

**GRANT** - vă permite să acordați drepturi altor utilizatori asupra structurilor bazei voastre de date.

**REVOKE** - puteți să anulați anumite drepturi utilizatorilor bazei de date.

Există multe metode prin care puteți rula comenzile SQL și vedea rezultatele rulării acestor comenzi. Pentru scopul acestui manual vă sfătuim să utilizați Oracle Database 10g Express Edition, o versiune simplificată a server-ului de Oracle, care este ideal pentru utilizarea pe calculatorul personal, fiind de dimensiuni mult reduse față de versiunea comercială a programului.

Puteți descărca gratuit această versiune a server-ului Oracle de pe site-ul Oracle de la adresa

<http://www.oracle.com/technology/software/products/database/xe/index.html>

însă veți fi solicitat să vă creați un cont pe acest site.

Vă prezentăm pe scurt pașii ce trebuie să îi urmați pentru a instala și configura Oracle Database 10g Express Edition.

**Pasul 1.** Porniți instalarea dând dublu click pe fișierul executabil descărcat de la adresa menționată anterior. Urmăriți pașii indicați de către programul de instalare. În unul dintre ecranele ce vor apărea vi se solicită introducerea unei parole. Aceasta va fi parola utilizatorului **SYSTEM** și veți avea nevoie de această parolă ulterior, deci notați-o pentru a nu o uita.



**Figura II.1.1** Introduceți parola utilizatorului **SYSTEM**

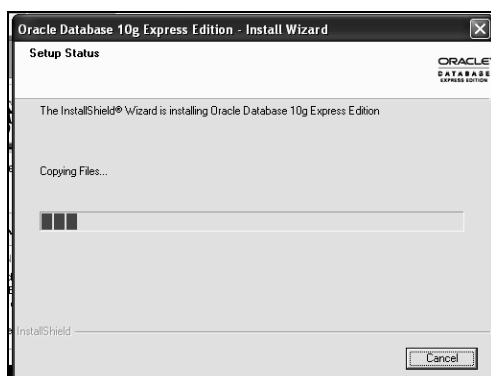


Figura II.1.2. Instalarea aplicației

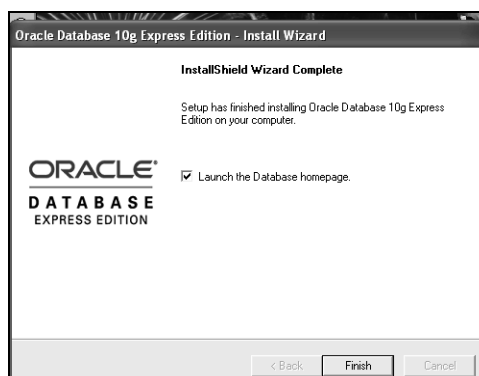


Figura II.1.3. Finalizarea instalării

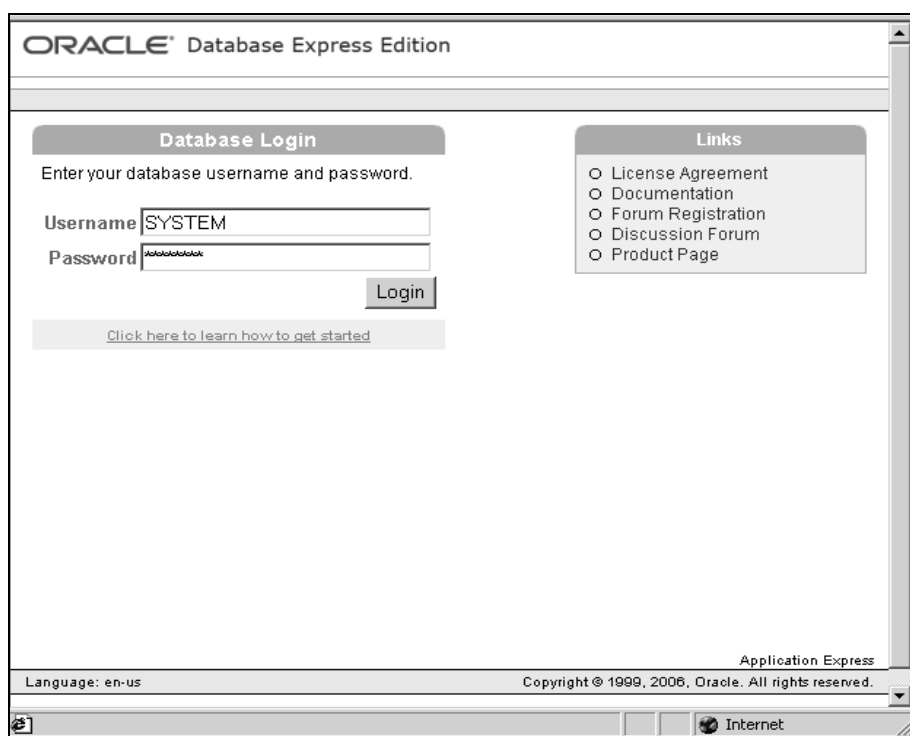


Figura II.1.4 Pagina principală a aplicației Oracle Database 10g Express Edition

**Pasul 2.** Logați-vă cu utilizatorul **SYSTEM** și parola dată la pasul 1.

**Pasul 3.** După logare alegeți opțiunea **Administration** și apoi **Database Users**. În noua fereastră deschisă (figura II.1.5) dați click pe iconul **HR**.

**HR** va fi numele de utilizator cu care vă veți putea loga pentru a rula comenzile SQL.

În fereastra Manage Database User (fig. II.1.6), faceți următoarele setări:

- introduceți parola pentru contul **HR**;
- în caseta **Account Status** selectați opțiunea **Unlocked**;
- în zona **Roles** asigurați-vă că sunt bificate opțiunile **CONNECT** și **RESOURCE**.

Apoi dați clic pe butonul **Alter User**.

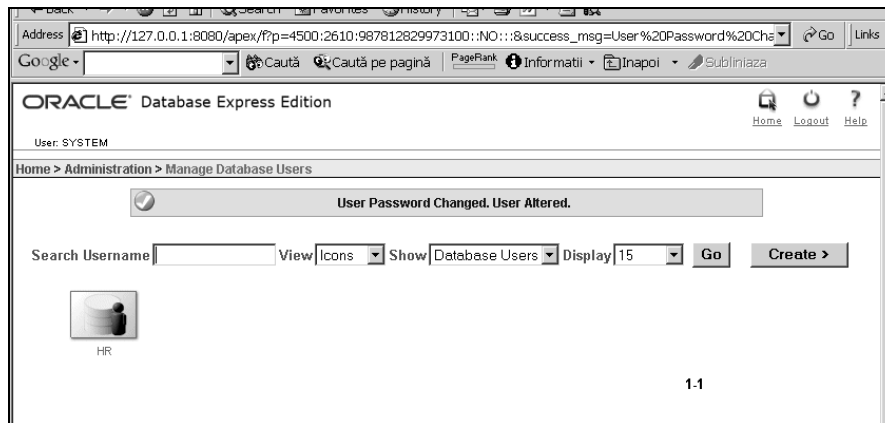


Figura II.1.5. Fereastra Database Users

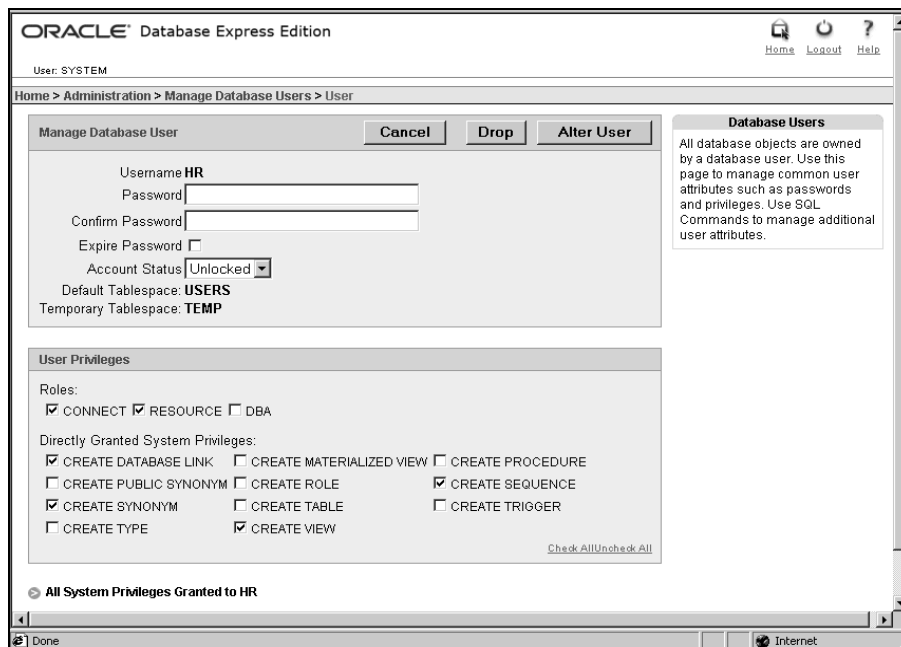
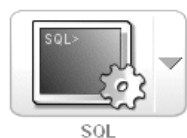


Figura II.1.6. Setarea drepturilor pentru utilizatorul HR

**Pasul 4.** Apăsați butonul **logout** din colțul dreapta sus al paginii și logați-vă cu noul cont creat.

**Pasul 5.** Pentru rularea comenzilor SQL veți da click pe butonul **"SQL"** (fig. II.1.7) iar apoi pe butonul **"SQL Commands"** (fig II.1.8).



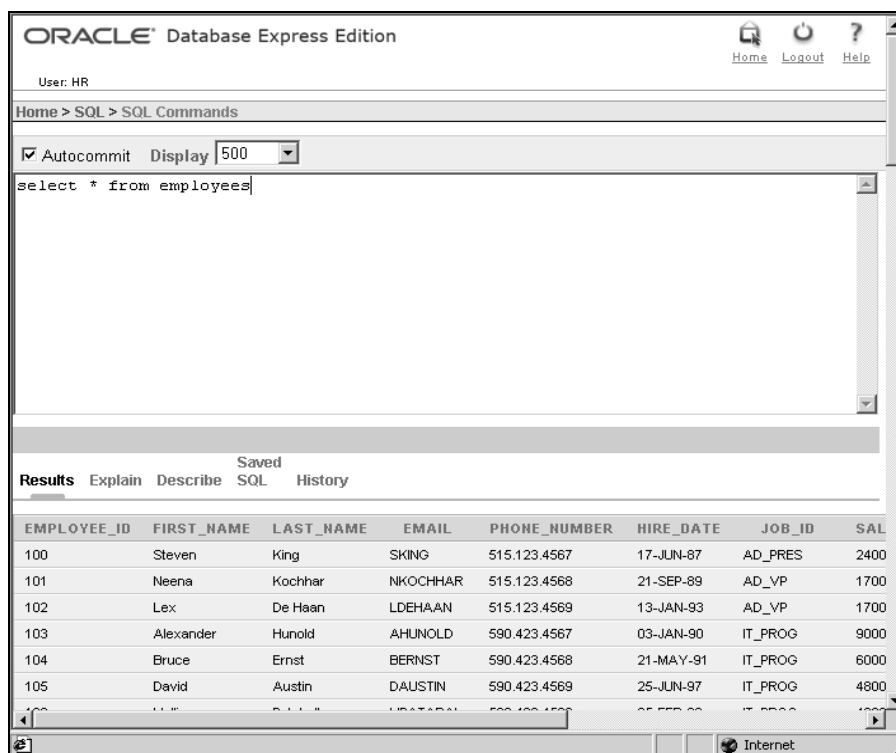
**Figura II.1.7.** Butonul **SQL**



**Figura II.1.8.** Butonul **SQL Commands**

În următoarea fereastră puteți rula comenzile SQL. Veți scrie comenzile în caseta text din această fereastră, apoi acționați butonul **Run** sau apăsați tastele **Ctrl+Enter**. Rezultatele rulării comenzii sau eventualele erori depistate vor fi afișate sub caseta text în care introduceți comenzile (fig. II.1.9.).

Dacă rezultatul comenzii va conține mai multe linii, pentru a le putea vedea pe toate, alegeți din caseta **Display** (afată deasupra casetei în care introduceți comenzile SQL) numărul dorit de linii afișate.



**Figura II.1.9.** Fereastra **SQL Commands**



Implicit baza de date conține câteva tabele populate cu date. Pentru a putea vedea care sunt aceste tabele, care este structura lor, ce date conțin, etc., din pagina principală a aplicației alegeți opțiunea **Object Browser**. În panoul din stânga dați click pe numele unei tabele și în panoul din dreapta aveți mai multe opțiuni pentru vizualizarea și modificarea structurii și conținutului tablei respective (fig II.1.10).

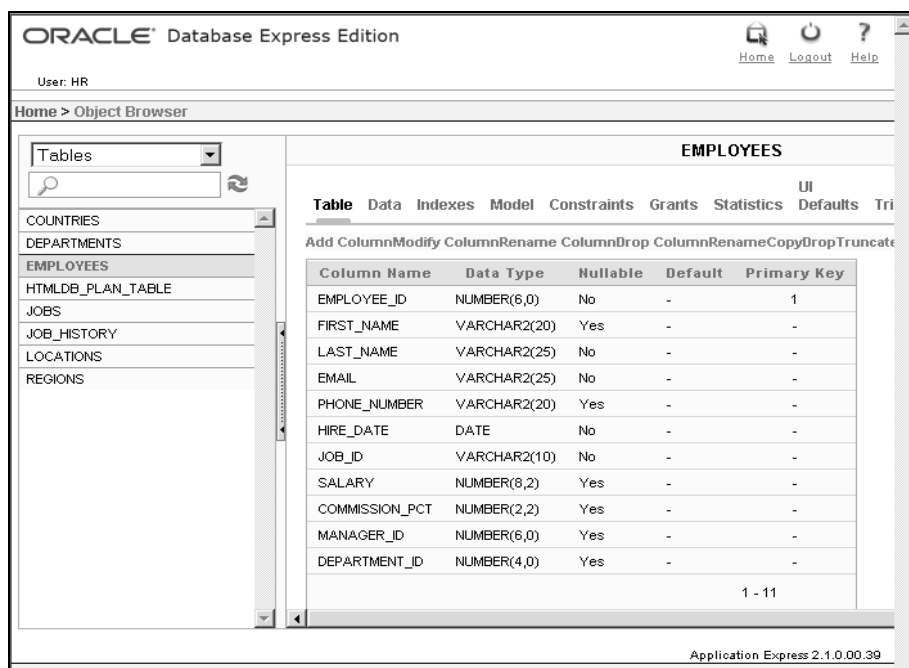


Figura II.1.10. Fereastra **Object Browser**

## II.1.2. Elemente de bază ale SQL

Vom prezenta foarte pe scurt principalele elemente ce intră în componența unei comenzi SQL.

### Nume

Toate obiectele dintr-o bază de date, tabele, coloane, vizualizări, indcsi, sinonime, etc, au un nume.

Numele poate fi orice șir de maximum 30 de litere, cifre și caracterele speciale: caracterul de subliniere (underscore \_), diez (#), și dolar (\$), primul caracter fiind obligatoriu o literă. Evident numele unui obiect din baza de date trebuie să fie unic.

## Cuvinte rezervate

Ca în orice limbaj, și în SQL există o listă de cuvinte rezervate. Acestea sunt cuvinte pe care nu le puteți folosi cu alt scop, ca de exemplu pentru denumirea tabelelor voastre.

## Constante

O constantă sau literal este o valoare fixă ce nu poate fi modificată. Există:

- constante numerice, de exemplu 2, 3.5, .9, etc. Se observă că dacă un număr real are partea întreagă egală cu zero, ea nu mai trebuie precizată.
- constante alfanumerice (sau șir de caractere). Constantele șir de caractere sunt scrise între apostrofuri și sunt case-sensitive. Exemple: 'abc', 'Numele'.

## Variabile

Variabilele sunt date care pot avea în timp valori diferite. O variabilă are întotdeauna un nume pentru a putea fi referită.

SQL suportă două tipuri de variabile:

- variabilele asociate numelor coloanelor din tabele
- variabile sistem.

## Expresii

O expresie este formată din variabile, constante, operatori și funcții. Funcțiile vor face obiectul a două dintre următoarele capitole ale manualului. În continuare ne vom ocupa de operatorii ce pot fi folosiți în expresii.

### Operatori aritmetici

Operatorii aritmetici permisi în SQL sunt cei patru operatori din matematică: adunare +, scădere -, înmulțire \*, împărțire /. Ordinea de efectuare a operațiilor aritmetice este cea din matematică (mai întâi înmulțirea și împărțirea și apoi adunarea și scăderea).

### Operatori alfanumerici

Există un singur operator alfanumeric și anume operatorul de concatenare a două șiruri || (două bare verticale fără spații între ele).

De exemplu, expresia 'abc' || 'xyz' are valoarea 'abcxyz'.

### Operatori de comparație

Pe lângă operatorii obișnuiți de comparație: <, >, <=, >=, <> sau != (pentru diferit), =, SQL mai implementează următorii operatori speciali:

- **LIKE** – despre care vom discuta puțin mai târziu în acest capitol;

- **BETWEEN** – testează dacă o valoare se găsește într-un interval definit de două valori. Astfel, expresia

**x BETWEEN a AND b**

este echivalentă cu expresia

**(x>=a) AND (x<=b)**

- **IN** – testează dacă o valoare aparține unei mulțimi de valori specificate. De exemplu, expresia

**x IN (a,b,c)**

este echivalentă cu

**(x=a) OR (x=b) OR (x=c)**

- **IS NULL** și **IS NOT NULL** – se folosesc pentru a testa dacă o expresie are valoarea **NULL** sau nu. Comparația cu **NULL** nu se poate face folosind operatorii obișnuiți = și respectiv <>.

### Operatori logici

În ordinea priorității lor, aceștia sunt:

- **NOT** – negația logică
- **AND** – și logic, expresia **a AND b** este adevărată dacă și numai dacă ambii operanzi **a** și **b** au valoarea adevărat.
- **OR** – sau logic, expresia **a OR b** este adevărată dacă și numai dacă cel puțin unul dintre operanzii **a** și **b** au valoarea adevărat.

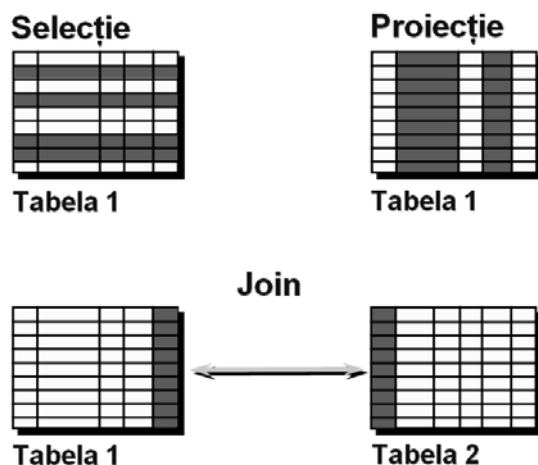
## II.1.3. Interogarea tabelelor. Comanda **SELECT**

Comanda **SELECT** este utilizată pentru a extrage date din baza de date. Setul de date returnate prin intermediul unei comenzi **SELECT** este compusă, ca și tabelele bazei de date, din linii și coloane, și vor putea fi simplu afișate, sau vom putea popula o tabelă cu datele returnate de către comanda **SELECT**, așa cum vom vedea într-un capitol următor.

Cu ajutorul comenzii **SELECT** putem realiza următoarele tipuri de operații:

- **selecția** – constă în filtrarea liniilor ce vor fi afișate. Vom folosi clauza **WHERE** pentru a defini criteriul sau criteriile pe care trebuie să le îndeplinească o linie pentru a fi returnată de către comanda **SELECT**.
- **proiecția** – constă în alegerea doar a anumitor coloane pentru a fi afișate.

- **join** – constă în preluarea datelor din două sau mai multe tabele, "legate" conform unor reguli precizate.



**Figura II.1.11.** Operațiile realizate cu ajutorul comenzii **SELECT**

Cea mai simplă formă a comenzii **SELECT** are sintaxa:

```
SELECT Lista_expresii
FROM tabela
```

În clauza **SELECT** se va preciza o listă de coloane sau expresii ce se vor afișa, separate prin câte un spațiu. În clauza **FROM** precizăm tabela din care se vor extrage coloanele ce vor fi afișate sau pe baza cărora vom realiza diverse calcule.

Vom exemplifica modul de folosire al comenzii **SELECT** pe tabela Persoane, având următoarea structură și conținut:

**Tabelul II.1.1.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	Ionescu	Gheorghe	Brasov	22	5	300
4	Georgescu	Maria	Iasi	30	6	890
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840
7	Bischin	Paraschiva	Brasov	22	-	500
8	Olaru	Angela	Ploiesti	22	2	1500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
3	Popescu	Ioan	Bucuresti	10	2	1200

Pentru a afișa toate datele (toate coloanele și toate liniile) din tabela **persoane** vom scrie simplu:

```
SELECT * FROM persoane
```

Observați că în locul listei de coloane am scris un singur asterisc, ceea ce înseamnă că dorim să afișăm toate coloanele tabelului.

Dacă însă dorim să afișăm doar informațiile din câteva coloane ale tabelului, de exemplu, dorim să afișăm numele, prenumele și localitatea fiecărei persoane, vom preciza numele coloanelor în clauza **SELECT**:

```
SELECT nume, prenume, localitate FROM persoane
```

rezultatul fiind cel din tabelul II.1.2.

**Tabelul II.1.2**

NUME	PRENUME	LOCALITATE
Ionescu	Gheorghe	Brasov
Georgescu	Maria	Iasi
Marinescu	Angela	Sibiu
Antonescu	Elena	Sibiu
Bischin	Paraschiva	Brasov
Olaru	Angela	Ploiesti
Vasilescu	Vasile	Cluj-Napoca
Popescu	Ioan	Bucuresti

După cum am precizat, putem realiza și calcule cu coloanele unei tabeli. De exemplu, pentru a afișa pentru fiecare persoană, salariul mărit cu 10%, folosim următoarea comandă:

```
SELECT nume, prenume, salariu, salariu * 1.10  
FROM persoane
```

și obținem:

**Tabelul II.1.3.**

NUME	PRENUME	SALARIU	SALARIU*1.10
Ionescu	Gheorghe	300	330
Georgescu	Maria	890	979
Marinescu	Angela	2100	2310
Antonescu	Elena	840	924
Bischin	Paraschiva	500	550
Olaru	Angela	1500	1650
Vasilescu	Vasile	950	1045
Popescu	Ioan	1200	1320

## Alias-ul unei coloane

Dacă priviți tabelul II.1.3, puteți observa că în capul de tabel afișat sunt trecute numele coloanelor cu majuscule sau expresia care a generat acea coloană, tot cu majuscule. Dacă dorim ca în capul de tabel să apară alt text, sau să nu se folosească doar majuscule va trebui să folosim un ALIAS pentru coloana respectivă. Alias-ul este introdus în clauza **SELECT**, imediat după numele coloanei respective astfel:

```
SELECT nume, prenume, salariu AS SalariuVechi,  
       salariu * 1.10 AS SalariuNou FROM persoane
```

În această comandă am stabilit două alias-uri **SalariuVechi** și respectiv **SalariuNou**. Trebuie subliniat că nu este obligatorie folosirea cuvântului **AS** pentru a defini un alias, însă este de preferat să îl utilizăm pentru o mai mare claritate. Comanda anterioară va afișa:

**Tabelul II.1.4.**

NUME	PRENUME	SALARIUVECHI	SALARIUNOU
Ionescu	Gheorghe	300	330
Georgescu	Maria	890	979
Marinescu	Angela	2100	2310
Antonescu	Elena	840	924
Bischin	Paraschiva	500	550
Olaru	Angela	1500	1650
Vasilescu	Vasile	950	1045
Popescu	Ioan	1200	1320

Puteți observa că deși în comanda **SELECT** am scris alias-urile folosind atât litere mici cât și litere mari, la afișare acestea sunt scrise tot cu majuscule. Pentru a evita acest lucru, trebuie să introducem alias-ul între ghilimele:

```
SELECT nume, prenume, salariu AS "SalariuVechi",  
       salariu * 1.10 AS "SalariuNou" FROM persoane
```

rezultatul obținut de această dată fiind cel din tabelul II.1.5.

De asemenea, dacă dorim ca alias-ul să conțină mai multe cuvinte de exemplu **Salariul Nou** respectiv **Salariul Vechi**, va trebui să folosim și de această dată ghilimele, în caz contrar generându-se o eroare. De exemplu comanda următoare va afișa tabelul II.1.6:

```
SELECT nume || ' ' || prenume "Numele si prenumele",  
       salariu AS "Salariu Vechi",  
       salariu * 1.10 AS "Salariu Nou" FROM persoane
```

Tabelul II.1.5.

NUME	PRENUME	SalariuVechi	SalariuNou
Ionescu	Gheorghe	300	330
Georgescu	Maria	890	979
Marinescu	Angela	2100	2310
Antonescu	Elena	840	924
Bischin	Paraschiva	500	550
Olaru	Angela	1500	1650
Vasilescu	Vasile	950	1045
Popescu	Ioan	1200	1320

Tabelul II.1.6.

Numele si prenumele	Salariu Vechi	Salariu Nou
Ionescu Gheorghe	300	330
Georgescu Maria	890	979
Marinescu Angela	2100	2310
Antonescu Elena	840	924
Bischin Paraschiva	500	550
Olaru Angela	1500	1650
Vasilescu Vasile	950	1045
Popescu Ioan	1200	1320

În cadrul clauzei **SELECT**, se pot folosi orice fel expresii în care se utilizează nume de coloane, constante, operatori, funcții, etc. Exemplificând, comanda următoare va afișa tabelul II.1.7.

```
SELECT nume||' '||prenume||' are salariul egal cu '||
salariu AS "Informatii persoane" FROM persoane
```

Tabelul II.1.7.

Informatii persoane
Ionescu Gheorghe are salariul egal cu 300
Georgescu Maria are salariul egal cu 890
Marinescu Angela are salariul egal cu 2100
Antonescu Elena are salariul egal cu 840
Bischin Paraschiva are salariul egal cu 500
Olaru Angela are salariul egal cu 1500
Vasilescu Vasile are salariul egal cu 950
Popescu Ioan are salariul egal cu 1200

## Eliminarea liniilor duplicate

Să analizăm rezultatul rulării următoarei comenzi:

```
SELECT localitate, firma  
FROM persoane
```

În tabelul II.1.8 se poate observa că în localitatea Brașov există două persoane care lucrează la aceeași firmă având codul 22.

**Tabelul II.1.8.**

LOCALITATE	FIRMA
Brasov	22
Iasi	30
Sibiu	-
Sibiu	10
Brasov	22
Ploiesti	22
Cluj-Napoca	15
Bucuresti	10

Dacă dorim să vedem la ce firme lucrează persoanele din fiecare localitate, însă o firmă să fie afișată o singură dată pentru o localitate anume, deci combinația valorilor localitate și firmă să fie unică, vom folosi clauza **DISTINCT** în cadrul clauzei **SELECT** astfel:

```
SELECT DISTINCT localitate, firma FROM persoane
```

combinația (Brașov, 22) fiind afișată acum o singură dată (tabelul II.1.9.).

**Tabelul II.1.9.**

LOCALITATE	FIRMA
Brasov	22
Bucuresti	10
Cluj-Napoca	15
Iasi	30
Ploiesti	22
Sibiu	10
Sibiu	-

Dar dacă dorim să afișăm doar localitățile ce apar în tabela Persoane, fiecare localitate să fie afișată o singură dată? Vom scrie:

```
SELECT DISTINCT localitate FROM persoane
```



rezultatul fiind acum:

**Tabelul II.1.10.**

LOCALITATE
Brasov
Bucuresti
Cluj-Napoca
Iasi
Ploiesti
Sibiu

## Filtrarea liniilor. Clauza WHERE

Imaginați-vă că tabela persoane conține date despre mii de persoane și că la un moment dat vă interesează doar informațiile despre persoanele dintr-o anumită localitate. Pentru a putea selecta doar acele linii care ne interesează, trebuie să adăugăm clauza **WHERE** la comanda **SELECT**. În această clauză vom preciza condițiile pe care trebuie să le îndeplinească o linie pentru a fi afișată. Așadar clauza **WHERE** permite realizarea operației de selecție (fig II.1.11).

De exemplu, pentru a afișa toate persoanele care provin din **București** sau **Brașov** vom scrie:

```
SELECT * FROM persoane
WHERE localitate='Brasov' OR localitate='Bucuresti'
```

care va afișa:

**Tabelul II.1.11.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500
3	Popescu	Ioan	Bucuresti	10	2	1200

Acum este timpul să vedem cum se folosește operatorul **LIKE**. Acesta este utilizat pentru a verifica dacă un șir de caractere respectă un anumit "model". Dacă valoarea se potrivește modelului, operatorul va returna valoarea **true** (adevărat) în caz contrar, va returna valoarea **False** (fals).

În model se pot utiliza următoarele caractere speciale:

- caracterul de subliniere (underscore **\_**) ține locul unui singur caracter, oricare ar fi acesta.

- caracterul procent (%) ține locul la zero sau mai multe caractere, oricare ar fi acestea.

De exemplu, dacă dorim să afișăm toate persoanele al căror prenume conține litera a pe orice poziție, vom scrie:

```
SELECT * FROM persoane
WHERE lower(prenume) LIKE '%a%'
```

Modelul '%a%' precizează că în fața caracterului a, în prenume, se pot găsi oricâte caractere, inclusiv zero caractere, iar după caracterul a se găsesc de asemenea oricâte caractere, inclusiv zero. Am folosit funcția **LOWER** pentru a transforma toate caracterele în litere mici. Altfel, numele care încep cu litera A, întrucât acesta e scris cu majuscule, nu ar fi fost afișate.

Rezultatul rulării acestei comenzi arată astfel:

**Tabelul II.1.12.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
4	Georgescu	Maria	Iasi	30	6	890
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840
7	Bischin	Paraschiva	Brasov	22	-	500
8	Olaru	Angela	Ploiesti	22	2	1500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
3	Popescu	Ioan	Bucuresti	10	2	1200

Dacă însă dorim să afișăm persoanele al căror prenume conține litera a pe a doua poziție vom folosi caracterul underscore în model:

```
SELECT * FROM persoane
WHERE prenume LIKE '_a%'
```

**Tabelul II.1.13.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
4	Georgescu	Maria	Iasi	30	6	890
7	Bischin	Paraschiva	Brasov	22	-	500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950

În cazul în care trebuie să verificăm dacă un șir conține unul dintre caracterele speciale underscore (\_), backslash (\), procent (%) vom scrie în model caracterul respectiv precedat de orice caracter special (de exemplu \ sau &), iar după model, vom preciza cu ajutorul clauzei **ESCAPE** care este caracterul special care introduce secvența corespunzătoare caracterelor \, \_, %.

Pentru a afișa persoanele din tabela **employees** al căror **job\_id** conține caracterul underscore (**\_**), pe a treia poziție de la sfârșit folosim comanda:

```
SELECT first_name, job_id FROM employees
WHERE job_id LIKE '%&__' ESCAPE '&'
```

sau

```
SELECT first_name, job_id FROM employees
WHERE job_id LIKE '%\__' ESCAPE '\'
```

iar dacă dorim să afișăm persoanele al căror **job\_id** conține un caracter underscore oriunde în șir vom utiliza comanda:

```
SELECT first_name, job_id FROM employees
WHERE job_id LIKE '%&_%' ESCAPE '&'
```

sau

```
SELECT first_name, job_id FROM employees
WHERE job_id LIKE '%\_%' ESCAPE '\'
```

Rezultatele afișate sunt cele din tabelul II.1.14, respectiv II.1.15.

**Tabelul II.1.14.**

FIRST_NAME	JOB_ID
Neena	AD_VP
Lex	AD_VP

**Tabelul II.1.15.**

FIRST_NAME	JOB_ID
Steven	AD_PRES
Neena	AD_VP
Lex	AD_VP
Alexander	IT_PROG
Bruce	IT_PROG
...	...

## II.1.4. Sortarea datelor. Clauza **ORDER BY**

Ați fost probabil destul de des în situația de a trebui să ordonați anumite date pe baza unor criterii oarecare. Imaginați-vă cam ce ar însemna să căutați numărul de telefon al unei persoane într-o carte de telefoane în care persoanele sunt trecute într-o ordine aleatoare, nu ordonate alfabetic așa cum suntem noi obișnuiți.

Pentru a preciza criteriile după care se ordonează datele folosim clauza **ORDER BY**. În această clauză se vor preciza coloanele sau expresiile după care se vor ordona liniile unei tabele înainte de a fi afișate.

De exemplu, afișarea datelor din tabela **persoane** în ordine alfabetică (crescătoare) a localității se face folosind comanda:

```
SELECT * FROM persoane
ORDER BY localitate
```

**Tabelul II.1.16.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
4	Georgescu	Maria	Iasi	30	6	890
8	Olaru	Angela	Ploiesti	22	2	1500
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840

Se observă că există mai multe persoane din aceeași localitate. Dacă vrem ca persoanele din aceeași localitate să fie ordonate descrescător după salariu scriem:

```
SELECT * FROM persoane
ORDER BY localitate, salariu DESC
```

opțiunea **DESC** precizează că sortarea se face descrescător. Pentru a sorta crescător se poate preciza acest lucru cu opțiunea **ASC**, dar aceasta este opțională deoarece implicit datele sunt sortate crescător.

Rezultatul rulării comenzii anterioare este cel din tabelul II.1.17.

**Tabelul II.1.17.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
7	Bischin	Paraschiva	Brasov	22	-	500
1	Ionescu	Gheorghe	Brasov	22	5	300
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
4	Georgescu	Maria	Iasi	30	6	890
8	Olaru	Angela	Ploiesti	22	2	1500
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840

Haideți să sortăm acum tabela **persoane** după codul firmei. Vom scrie:

```
SELECT * FROM persoane ORDER BY firma
```

Rularea acestei comenzi duce la afișarea tabelului II.1.18. Să observăm că Marinescu Angela, deoarece nu are completat codul firmei (valoarea codului firmei este null) a fost afișată ultima. Așadar la ordonarea crescătoare (implicită) valorile nule se trec la sfârșit, în timp ce la sortarea descrescătoare valorile nule apar la început.

**Tabelul II.1.18.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
6	Antonescu	Elena	Sibiu	10	1	840
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500
8	Olaru	Angela	Ploiesti	22	2	1500
4	Georgescu	Maria	Iasi	30	6	890
5	Marinescu	Angela	Sibiu	-	3	2100

Comanda

```
SELECT * FROM persoane
ORDER BY firma DESC
```

va face ca Marinescu Angela să fie afișată prima (tabelul II.1.19).

**Tabelul II.1.19.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
5	Marinescu	Angela	Sibiu	-	3	2100
4	Georgescu	Maria	Iasi	30	6	890
1	Ionescu	Gheorghe	Brasov	22	5	300
8	Olaru	Angela	Ploiesti	22	2	1500
7	Bischin	Paraschiva	Brasov	22	-	500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
6	Antonescu	Elena	Sibiu	10	1	840
3	Popescu	Ioan	Bucuresti	10	2	1200

În criteriile de ordonare pot să apară și expresii nu doar coloane din tabela interogată. Astfel, putem scrie:

```
SELECT * FROM persoane ORDER BY prenume || nume
```

rezultatul fiind cel din tabelul II.1.20.

De asemenea, putem preciza ca sortarea să se facă după o expresie care apare în clauza **SELECT** prin indicarea poziției expresiei respective în lista de expresii din clauza **SELECT**. Astfel comanda

```
SELECT nume, prenume, salariu
FROM persoane ORDER BY 3 DESC
```

va sorta descrescător liniile după salariu, deoarece în clauza **SELECT**, **salariu** este a treia expresie (**atenție**: în tabela persoane salariul este coloana a 7-a):

**Tabelul II.1.20.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
6	Antonescu	Elena	Sibiu	10	1	840
7	Bischin	Paraschiva	Brasov	22	-	500
4	Georgescu	Maria	Iasi	30	6	890
1	Ionescu	Gheorghe	Brasov	22	5	300
5	Marinescu	Angela	Sibiu	-	3	2100
8	Olaru	Angela	Ploiesti	22	2	1500
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950

**Tabelul II.1.21.**

NUME	PRENUME	SALARIU
Marinescu	Angela	2100
Olaru	Angela	1500
Popescu	Ioan	1200
Vasilescu	Vasile	950
Georgescu	Maria	890
Antonescu	Elena	840
Ionescu	Gheorghe	300
Bischin	Paraschiva	500

Mai mult, în clauza **ORDER BY** putem folosi alias-ul unei coloane ca în exemplul următor:

```
SELECT nume || ' ' || prenume AS "Nume si prenume", salariu
FROM persoane
ORDER BY "Nume si prenume"
```

rezultatul fiind cel din tabelul II.1.22.

Desigur clauzele **WHERE** și **ORDER BY** pot apărea împreună în aceeași comandă, ordinea în care acestea apar fiind **WHERE** și apoi **ORDER BY**, aceasta fiind și ordinea în care sunt executate: mai întâi sunt selectate liniile care trebuie să fie afișate și abia apoi sunt sortate conform criteriului stabilit prin clauza **ORDER BY**. De exemplu, pentru a afișa în ordine descrescătoare a salariilor doar persoanele din Brașov și Sibiu scriem:

```
SELECT * FROM persoane
WHERE localitate IN ('Sibiu', 'Brasov')
ORDER BY salariu DESC
```

rezultatul rulării acestei comenzi fiind cel din tabelul II.1.23.

**Tabelul II.1.22.**

Nume si prenume	SALARIU
Antonescu Elena	840
Bischin Paraschiva	500
Georgescu Maria	890
Ionescu Gheorghe	300
Marinescu Angela	2100
Olaru Angela	1500
Popescu Ioan	1200
Vasilescu Vasile	950

**Tabelul II.1.23.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500

## II.1.5. Afișarea primelor *n* linii

La sfârșitul anului școlar, dirigintele clasei vă roagă să-l ajutați să afle care sunt primii trei elevi din clasă, în ordinea descrescătoare a mediei generale, pentru a ști cui să dea premiile. Așadar se pune problema ca la afișarea datelor dintr-o tabelă să afișați doar primele *n* linii.

Pentru aceasta veți avea nevoie de pseudocoloana **ROWNUM** care returnează numărul de ordine al unei linii într-o tabelă. De exemplu comanda următoare va afișa codul, numele și prenumele persoanelor împreună cu numărul de ordine al acestora în tabela **persoane**:

```
SELECT cod, nume, prenume, rownum
FROM persoane
```

rezultatul este cel din tabelul următor:

**Tabelul II.1.24.**

COD	NUME	PRENUME	ROWNUM
1	Ionescu	Gheorghe	1
4	Georgescu	Maria	2
5	Marinescu	Angela	3
6	Antonescu	Elena	4
7	Bischin	Paraschiva	5
8	Olaru	Angela	6
2	Vasilescu	Vasile	7
3	Popescu	Ioan	8

Deși ne-am aștepta ca o comandă **SELECT** care folosește clauza **ORDER BY**, **ROWNUM** să ne afișeze numărul de ordine al înregistrărilor în ordinea dată de **ORDER BY**, acest lucru nu se întâmplă, numărul de ordine fiind cel din tabela inițială. Observați în acest sens tabelul II.1.25 afișat la rularea comenzii următoare

```
select rownum, cod, nume, prenume,
       localitate, firma, job, salariu
from persoane
order by salariu desc
```

**Tabelul II.1.25.**

ROWNUM	COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
3	5	Marinescu	Angela	Sibiu	-	3	2100
6	8	Olaru	Angela	Ploiesti	22	2	1500
8	3	Popescu	Ioan	Bucuresti	10	2	1200
7	2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
2	4	Georgescu	Maria	Iasi	30	6	890
4	6	Antonescu	Elena	Sibiu	10	1	840
5	7	Bischin	Paraschiva	Brasov	22	-	500
1	1	Ionescu	Gheorghe	Brasov	22	5	300

Așadar, dacă dorim să afișăm primele 3 înregistrări din tabela inițială vom putea scrie simplu:

```
SELECT cod, nume, prenume, rownum
FROM persoane
WHERE ROWNUM<=3
```



afișându-se rezultatul dorit (tabelul II.1.26.)

**Tabelul II.1.26.**

COD	NUME	PRENUME	ROWNUM
1	Ionescu	Gheorghe	1
4	Georgescu	Maria	2
5	Marinescu	Angela	3

Însă, pentru a afișa persoanele cu cele mai mici trei salarii, comanda următoare nu afișează ceea ce am dori, deoarece Oracle, prima dată, va returna primele trei înregistrări din tabela persoane și abia apoi le va sorta:

```
select rownum, cod, nume, prenume,
       localitate, firma, job, salariu
from persoane
where rownum<=3
order by salariu desc
```

comanda aceasta afișând:

**Tabelul II.1.27.**

ROWNUM	COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
3	5	Marinescu	Angela	Sibiu	-	3	2100
2	4	Georgescu	Maria	Iasi	30	6	890
1	1	Ionescu	Gheorghe	Brasov	22	5	300

Pentru a obține rezultatul dorit de noi vom folosi o subinterogare astfel:

```
select *
from (select * from persoane
      order by salariu)
where rownum<=3
```

În acest fel am forțat Oracle să sorteze mai întâi liniile și apoi să afișeze primele trei linii din tabela obținută.

**Tabelul II.1.28.**

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500
6	Antonescu	Elena	Sibiu	10	1	840



## Aplicații

Întrebările 1-8 se vor referi la următoarele tabele despre mesajele postate pe un forum:

### Users

- #UserId (number)
- UserName (varchar2)
- Cost (numeric)

### Groups

- #GroupId (number)
- Title (varchar2)
- Category (varchar2)
- NumberOfPosts (number)
- GroupSize (number)
- Owner (number)

### Posts

- #PostId (number)
- UserId (number)
- GroupId (number)
- ThreadId (number)
- PostText (varchar2)
- DateCreated (date)

1. Afișați numele grupurilor 2 și 3.
2. Afișați textul și id-urile tuturor mesajelor postate de utilizatorul 4 înainte de 1 martie 2007.
3. Afișați id-urile tuturor persoanelor care au postat un mesaj în thread-ul 2.
4. Afișați titlurile și categoriile grupurilor 2, 3 și 6. Ordonăți rezultatele crescător după categorie.
5. Pentru toate mesajele postate după 2 ianuarie 2007, afișați id-ul utilizatorului, id-ul grupului, și data postării. Ordonăți rezultatele crescător după id-urile utilizatorilor, iar pentru un utilizator, cel mai recent mesaj postat de el va fi afișat primul.
6. Afișați toate grupurile al căror titlu începe cu litera s sau S.
7. Tabela **users** memorează costurile în dolari pentru fiecare utilizator. Presupunând că un dolar este egal cu 3 RON, afișați pentru fiecare utilizator id-ul, numele și costul în RON.
8. Afișați id-ul, textul și data postării tuturor mesajelor postate în 2006 al căror text conține cuvântul "bike".

Următoarele întrebări se referă la tabela **employees**, pre-existentă în contul HR din Oracle Database 10g Express Edition. Pentru a vedea care sunt câmpurile acestei tabele și care este tipul fiecărei coloane rulați comanda

**DESCRIBE employees**

9. Afișați numele, prenumele și salariul angajaților al căror salariu este mai mare de 3000.

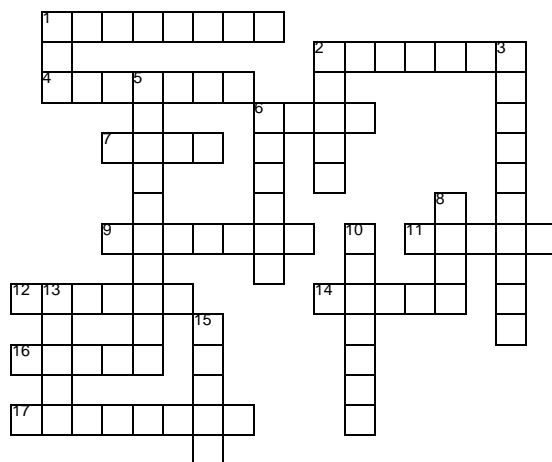
10. Afișați numele, prenumele și id-ul departamentului angajatului cu codul 109.
11. Afișați numele, prenumele și salariul angajaților a căror departament are id-ul în afara intervalului [ 40 , 70 ].
12. Afișați persoanele angajate între 1 mai 2006 și 1 martie 2007. Ordonăți rezultatele crescător după data angajării.
13. Afișați toate informațiile despre angajații din departamentele 20 și 50.
14. Afișați numele, prenumele și salariul angajaților din departamentele 20 și 50 care au salariul mai mare de 3000. Etichetați cele trei coloane cu **Numele**, **Prenumele** și respectiv **Salariul Lunar**.
15. Afișați numele și job\_id-ul angajaților care nu au manager.
16. Afișați toți angajații pentru care a treia literă din prenume este a.



## Joc

Vă propunem să vă testați cunoștințele înscrise în acest capitol rezolvând următorul rebus. Cuvintele care se completează în careu sunt date de răspunsurile la întrebările de mai jos. Acestea se referă la tabela **cuvinte** având următorul conținut:

Cod	Cuvant	Valoare
5	Gifts	120
7	Red	157
1	White	854
3	Pink	69
94	Heart	2541
8	Cupid	124
12	Love	33
75	Chocolates	22
55	Romantic	54
64	Couples	36
10	Flowers	147
14	Roses	69
27	Poetry	154
48	Dancing	124
65	Mood	563
93	Sweatheart	33
44	Sweets	21
16	Bouquet	54
80	February	99
6	Candy	698



#### Orizontal

1. Al doilea cuvânt afișat de comanda

```
select cuvânt from cuvinte
where valoare between 50 and 70
order by cuvânt desc
```

2. Primul cuvânt afișat de comanda

```
select cod, cuvânt from cuvinte
where cuvânt LIKE '%o%u%'
order by 2 desc
```

4. Cuvântul afișat de comanda

```
select cuvânt from cuvinte
where cuvânt LIKE 'D%'
```

6. Cuvântul a cărui valoare este dată de comanda:

```
select distinct valoare
from cuvinte
where valoare between 60 and 70
```

#### Vertical

1. Al doilea cuvânt afișat de comanda

```
select cuvânt from cuvinte
where valoare>150
order by valoare, cod desc
```

2. Cuvântul afișat de comanda

```
select cuvânt from cuvinte
where cuvânt like '%d%' and
valoare>600
```

3. Cuvântul afișat de comanda

```
select cuvânt from cuvinte
where valoare*3-6 = cod
```

5. Cuvântul afișat de comanda

```
select * from cuvinte
where cuvânt like '_h%o%'
```

6. Cuvântul afișat de comanda

```
select * from cuvinte
where cuvânt like '%y%' and
not cuvânt like '%a%'
```

**7. Cuvântul afișat de comanda**

```
select cuvant from cuvinte
where cod in (12,27) and
      lower(cuvant) like '%l%'
```

**9. Cuvântul afișat de comanda**

```
select cuvant from cuvinte
where cuvant like '___w%'
```

**11. Cuvântul afișat de comanda**

```
select cuvant from cuvinte
where lower(cuvant) like '%s%s'
and cod<20
```

**12. Cuvântul afișat de comanda**

```
select cuvant from cuvinte
where cod in (10,44) and
      (cuvant like '%e%e%' or
       valoare=99)
```

**14. Al treilea cuvânt afișat de comanda**

```
select cuvant from cuvinte
where cod in (8,44) and
      cuvant like '%u%' or
      valoare between 8 and 44
order by cuvant
```

**16. Cuvântul afișat de comanda**

```
select cuvant from cuvinte
where valoare>100 and valoare<150
and cod<10
and cuvant like '%t%'
```

**17. Cuvântul afișat de comanda**

```
select * from cuvinte
where cuvant like '%r_a__'
```

**8. Primul cuvânt afișat de comanda**

```
select cuvant, valoare
from cuvinte
where cuvant like '%o%'
order by 2 desc
```

**10. Primul cuvânt afișat de comanda**

```
select cuvant, valoare
from cuvinte
where cuvant like '%ou%'
order by 2 desc
```

**13. Cuvântul afișat de comanda**

```
select cuvant from cuvinte
where cuvant like '_h%'
and valoare>100
```

**15. Cuvântul afișat de comanda**

```
select cuvant from cuvinte
where cuvant like 'H%t'
```

(vezi rezolvarea la pagina 312)

1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ ce este și cum se folosește  
tabela DUAL
- ✓ ce sunt funcțiile singulare
- ✓ care sunt categoriile de funcții  
singulare predefinite în  
Oracle
- ✓ cum se folosesc funcțiile  
singulare în interogări
- ✓ care sunt și cum se folosesc  
funcțiile numerice
- ✓ care sunt și cum funcționează  
funcțiile care operează  
asupra caracterelor
- ✓ care sunt și cum funcționează  
funcțiile care operează  
asupra datelor calendaristice
- ✓ care sunt și cum se folosesc  
funcțiile de conversie

## II.2.1. Tipuri de funcții

Funcțiile Oracle sunt împărțite astfel:

- **Funcții singulare** – acestea operează la un moment dat asupra unei singure înregistrări. Aceste funcții vor fi discutate în acest capitol.
- **Funcțiile de grup** – operează asupra unui grup de înregistrări și returnează o singură valoare pentru întregul grup.

Funcțiile singulare pot fi folosite în:

- clauza **SELECT**, pentru a modifica modul de afișare a datelor, pentru a realiza diferite calcule, etc.;
- clauza **WHERE**, pentru a preciza mai exact care sunt înregistrările ce se afișează;
- clauza **ORDER BY**.

Funcțiile singulare (*single-row functions*) pot fi la rândul lor împărțite în:

- Funcții care operează asupra șirurilor de caractere;
- Funcții numerice;
- Funcții pentru manipularea datelor calendaristice;
- Funcții de conversie – care convertesc datele dintr-un tip în altul;
- Funcții de uz general.

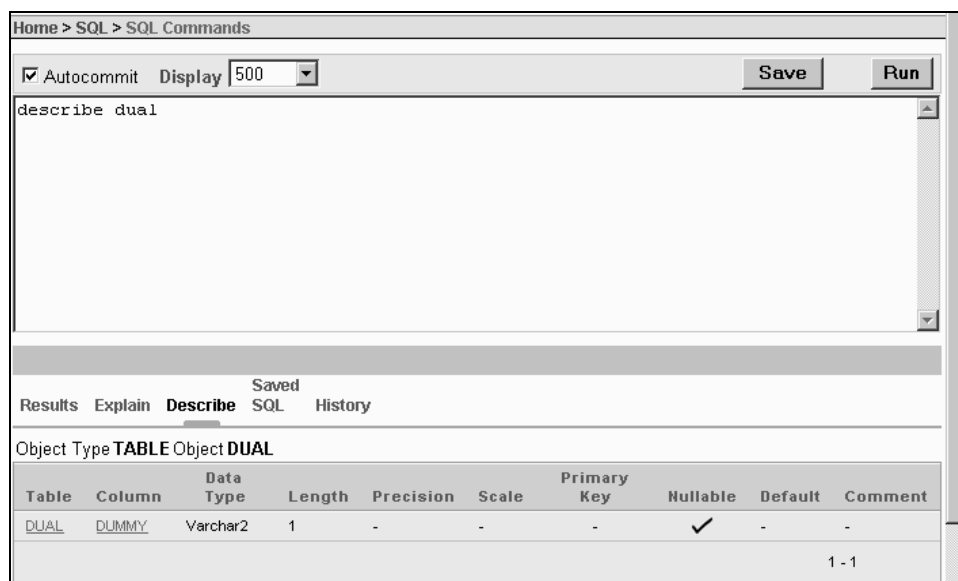
Unele funcții, precum **TRUNC** și **ROUND**, pot acționa asupra mai multor tipuri de date, dar cu semnificații diferite.

## II.2.2. Tabela DUAL

În cele ce urmează vom folosi tabela **DUAL** pentru a testa modul de operare a funcțiilor singulare.

Această tabelă este una specială, care conține o singură coloană numită **"DUMMY"** și o singură linie (vezi figura II.2.1).

Tabela **DUAL** se folosește atunci când realizăm calcule sau evaluăm expresii care nu derivă din nicio tabelă anume.



**Figura II.2.1.** Structura tabelii **DUAL**

Fie de exemplu comanda:

```
SELECT (5*7-3)/2 FROM DUAL;
```

Expresia evaluată în această comandă nu are în componență nici o coloană a vreunei tabeli, motiv pentru care este nevoie să apelăm la tabela **DUAL**.

Putem privi tabela **DUAL** ca pe o variabilă în care memorăm rezultatele calculelor noastre.

Tabela **DUAL** este o facilitate specifică Oracle. Este echivalentul tabelii **SYSDUMMY1** din **DB2**, tabelă aflată în shema sistem **SYSIBM**. În **Microsoft SQL server 2000** este permisă scrierea de interogări fără clauza **FROM**.

## II.2.3. Funcții asupra șirurilor de caractere

Șirurile de caractere pot conține orice combinație de litere, numere, spații, și alte simboluri, precum semne de punctuație, sau caractere speciale. În Oracle există două tipuri de date pentru memorarea șirurilor de caractere:

- **CHAR** – pentru memorarea șirurilor de caractere de lungime fixă
- **VARCHAR2** – pentru memorarea șirurilor de caractere având lungime variabilă.
- **LOWER(sir)** – convertește caracterele alfanumerice din șir în litere mari.



- **UPPER(sir)** – convertește caracterele alfanumerice din șir în litere mici.
- **INITCAP(sir)** – convertește la majusculă prima literă din fiecare cuvânt al șirului. Cuvintele sunt șiruri de litere separate prin orice caracter diferit de literă. Literele din interiorul cuvântului care erau scrise cu majuscule vor fi transformate în litere mici.

Exemplu	Rezultatul afișat
<code>SELECT LOWER(first_name) FROM employees;</code>	afișează prenumele persoanelor din tabela <b>employees</b> scrise cu litere mici
<code>SELECT LOWER('abc123ABC') FROM DUAL;</code>	abc123abc
<code>SELECT UPPER('abc123ABC') FROM DUAL;</code>	ABC123ABC
<code>SELECT INITCAP('aBc def*ghi') FROM dual;</code>	Abc Def*Ghi <i>Explicație:</i> șirul conține 3 cuvinte aBc def și ghi

- **CONCAT(sir1, sir2)** – concatenează două șiruri de caractere

Exemplu	Rezultatul afișat
<code>SELECT CONCAT('abc','def') FROM dual;</code>	abcdef <i>Explicație:</i> comanda poate fi transcrisă folosind operatorul de concatenare: <code>SELECT 'abc'    'def' FROM dual;</code>

- **SUBSTR(sir,poz,nr)** – extrage din **sir** cel mult **nr** caractere începând din poziția **poz**.

#### Observații

- dacă din poziția **poz** până la sfârșitul șirului sunt mai puțin de **nr** caractere, se vor extrage toate caracterele de la poziția **poz** până la sfârșitul șirului.
- parametrul **poz** poate fi și o valoare negativă, ceea ce înseamnă că poziția de unde se va începe extragerea caracterelor din șir se va determina numărând caracterele din șir de la dreapta spre stânga (vezi ultimele 3 exemple de mai jos).
- dacă **nr** nu este specificat, se va returna subșirul începând cu caracterul de pe poziția **poz** din șir până la sfârșitul șirului.

Exemplu	Rezultatul afișat
<code>select substr('abc<u>d</u>ef',3,2) from dual</code>	cd

Exemplu	Rezultatul afișat
<code>select substr('abcdef',3,7) from dual</code>	<code>cdef</code> <b>Explicație.</b> Chiar dacă din poziția 3 până la sfârșitul șirului nu mai sunt 7 caractere se returnează caracterele rămase
<code>select substr('abcdef',3) from dual</code>	<code>cdef</code> <b>Explicație.</b> Același rezultat ca mai sus dacă nu se specifică numărul de caractere ce se extrag
<code>select substr('abcdef',7,3) from dual</code>	nu se va afișa nimic deoarece nu există poziția 7 în șir, acesta având doar 5 caractere.
<code>select substr('abcdef',-4,2) from dual</code>	<code>cd</code> <b>Explicație.</b> Se extrag două caractere începând cu al patrulea caracter din dreapta.
<code>select substr('abcdef',-4,7) from dual</code>	<code>cdef</code>
<code>select substr('abcdef',-10,5) from dual</code>	nu se va afișa nimic deoarece șirul conține mai puțin de 10 caractere

- **INSTR(sir, subsir, poz, k)** – returnează poziția de început a celei de a k-a apariții a subșirului **subsir** în șirul **sir**, căutarea făcându-se începând cu poziția **poz**. Dacă parametrii **poz** și **k** lipsesc, atunci se va returna poziția primei apariții a subșirului **subsir** în întregul șir **sir**.

Poziția de unde începe căutarea poate fi precizată și relativ la sfârșitul șirului, ca și în cazul funcției **substr**, dacă parametrul **poz** are o valoare negativă.

Exemplu	Rezultatul afișat
<code>select instr('abcdabcdabc','cd') from dual</code>	3
<code>select instr('abcd','ef') from dual</code>	0
<code>select instr('abcd','bce') from dual</code>	0
<code>select instr('ababababababab','ab',4,2) from dual</code>	7 <b>Explicație.</b> Se începe căutarea din poziția a patra, adică în zona subliniată cu o linie, și se afișează poziția de start a celei de a doua apariții, (subșirul subliniat cu linie dublă)

Exemplu	Rezultatul afișat
<pre>select instr('abababababab','ab',-4,1) from dual</pre>	9

- **LENGTH(sir)** – returnează numărul de caractere din șirul **sir**.

Exemplu	Rezultatul afișat
<pre>select length('abcd') from dual</pre>	4

- **LPAD(sir1,nr,sir2)** – completează șirul **sir1** la stânga cu caracterele din șirul **sir2** până ce șirul obținut va avea lungimea **nr**.

Dacă lungimea șirului **sir1** este mai mare decât **nr**, atunci funcția va realiza trunchierea șirului **sir1**, ștergându-se caracterele de la sfârșitul șirului.

Exemplu	Rezultatul afișat
<pre>select lpad('abcd',3,'*') from dual</pre>	abc
<pre>select lpad('abcd',10,'*.*') from dual</pre>	*.*.*.abcd
<pre>select lpad('abc',10,'*.*') from dual</pre>	*.*.*.*abc
<pre>select lpad('abc',5,'xyzw') from dual</pre>	xyabc

- **RPAD(sir,nr,subsir)** – similară cu funcția **LPAD**, completarea făcându-se la dreapta.

Exemplu	Rezultatul afișat
<pre>select rpad('abcd',3,'*') from dual</pre>	abc
<pre>select rpad('abcd',10,'*.*') from dual</pre>	abcd*.*.*.
<pre>select rpad('abc',10,'*.*') from dual</pre>	abc*.*.*.*
<pre>select rpad('abc',5,'xyzw') from dual</pre>	abcxy

- **TRIM(LEADING ch FROM sir)**      **TRIM(sir)**  
**TRIM(TRAILING ch FROM sir)**      **TRIM(ch FROM sir)**  
**TRIM(BOTH ch FROM sir)**
  - funcția **TRIM** șterge caracterele **ch** de la începutul, sfârșitul sau din ambele părți ale șirului **sir**.
  - în ultimele două formate ale funcției este subînțeleasă opțiunea **BOTH**.

- dacă **ch** nu este specificat se vor elimina spațiile inutile de la începutul, sfârșitul sau din ambele părți ale șirului **sir**.

Exemplu	Rezultatul afișat
<code>select trim(leading 'a' from 'aaxaxaa') from dual</code>	<code>xaxaa</code>
<code>select trim(trailing 'a' from 'aaxaxaa') from dual</code>	<code>aaxax</code>
<code>select trim(both 'a' from 'aaxaxaa') from dual</code>	<code>xax</code>
<code>select trim('a' from 'aaxaxaa') from dual</code>	<code>xax</code>
<code>select '*'    trim(' abc ')    '*' from dual</code>	<code>*abc*</code>

- **REPLACE(sir, subsir, sirnou)** - înlocuiește toate aparițiile subșirului **subsir** din șirul **sir** cu șirul **sirnou**. Dacă nu este specificat noul șir, toate aparițiile subșirului **subsir** se vor elimina.

Exemplu	Rezultatul afișat
<code>select replace('abracadabra', 'ab', 'xy') from dual</code>	<code>xyracadxyra</code>
<code>select replace('abracadabra', 'ab', 'xyz') from dual</code>	<code>xyzracadxyzra</code>
<code>select replace('abracadabra', 'a') from dual</code>	<code>brcdbr</code>

## Combinarea funcțiilor asupra șirurilor de caractere

Evident într-o expresie pot fi folosite două sau mai multe astfel de funcții, imbricate ca în următorul exemplu.

```
SELECT substr('abcabcabc',1,instr('abcabcabc','bc')-1) ||  
       'xyz' ||  
       substr('abcabcabc',instr('abcabcabc','bc')+length('bc'))  
FROM dual
```

Să analizăm pe această comandă

```
instr('abcabcabc','bc')
```

care returnează poziția primei apariții a șirului **'bc'** în șirul **'abcabcabc'**, adică **2**. Primul apel al funcției **substr** este deci echivalent cu apelul

`substr('abcabcabc',1,1)`

adică extrage doar prima literă 'a'. Al doilea apel al funcției substr este echivalent cu

`substr('abcabcabc',4)`

adică extrage toate caracterele de la poziția 4 până la sfârșitul șirului, deci 'abcabc'. Așadar, cele două apeluri extrag subșirul de dinaintea primei apariții a lui 'bc' în șirul 'abcabcabc', și respectiv de după această apariție. Cele două secvențe se concatenează apoi între ele incluzându-se șirul 'xyz'. În concluzie comanda înlocuiește prima apariție a șirului 'bc' din șirul 'abcabcabc' cu șirul 'xyz'.

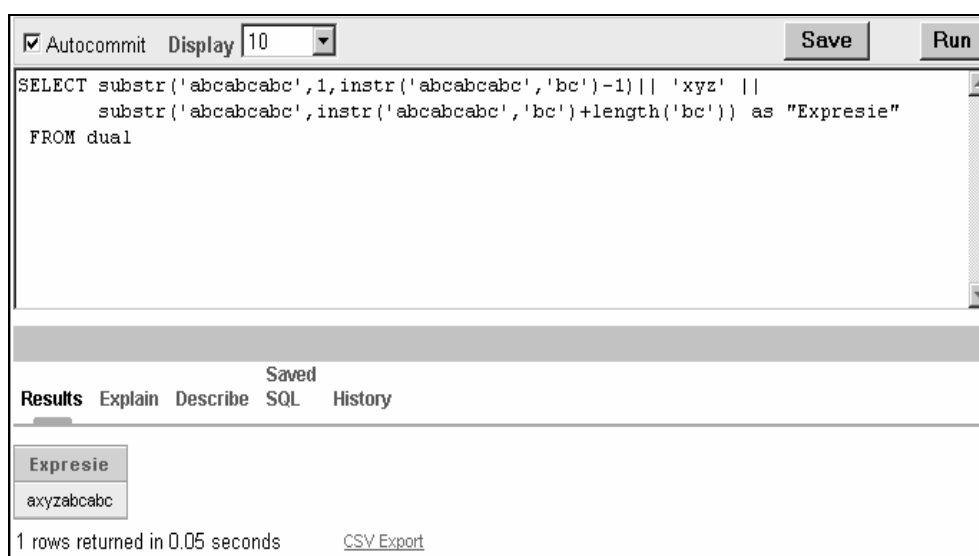


Figura II.2.2 Combinarea funcțiilor caracter

## II.2.4. Funcții numerice

Aceste funcții operează asupra valorilor numerice și returnează un rezultat numeric. Funcțiile numerice oferite de Oracle sunt destul de puternice.

- **ABS(n)** – returnează valoarea absolută a argumentului.

Exemplu	Rezultatul afișat
<code>select abs(-5.23) from dual</code>	5.23
<code>select abs(5) from dual</code>	5

- **ACOS(n), ASIN(n), ATAN(n)** – sunt funcțiile trigonometrice inverse, cu semnificația din matematică. Valoarea returnată de aceste funcții este exprimată în radiani.

- **SIN(n), COS(n), TAN(n)** – sunt funcțiile trigonometrice cu aceeași semnificație ca și la matematică. Argumentul acestor funcții trebuie precizat în radiani.

Exemplu	Rezultatul afișat
<code>select sin(3.1415/2) from dual</code>	.9999999989269140374952 06086034346145374
<code>select cos(3.1415/2) from dual</code>	.0000463267948800483535 5670590049419594

- **POWER(m,n)** – calculează valoarea  $m^n$ .

Exemplu	Rezultatul afișat
<code>select power(2,5) from dual</code>	32
<code>select power(2,0.5) from dual</code>	1.414213562373095048801 68872420969807855
<code>select power(2,-1) from dual</code>	.5
<code>select power(2,-0.75) from dual</code>	.5946035575013605333587 49985280237957651

- **SQRT(x)** – calculează rădăcina pătrată a argumentului. Apelul **SQRT(x)** returnează aceeași valoare ca și **POWER(x,0.5)**.

Exemplu	Rezultatul afișat
<code>select sqrt(3) from dual</code>	1.732050807568877293527 44634150587236694

- **REMAINDER(x,y)** – funcția determină mai întâi acel multiplu a lui **y** care este cel mai apropiat de **x** și returnează apoi diferența dintre **x** și acel multiplu.

Exemplu	Rezultatul afișat
<code>select remainder(10,3) from dual</code>	1 <b>Explicație.</b> Cel mai apropiat de 10 multiplu a lui 3 este 9. $10-9=1$ .
<code>select remainder(5,3) from dual</code>	-1 <b>Explicație.</b> Cel mai apropiat de 5 multiplu a lui 3 este 6, iar $5-6=-1$ .
<code>select remainder(10,3.5) from dual</code>	-0.5 <b>Explicație.</b> Cel mai apropiat de 10 multiplu a lui 3.5 este 10.5, iar $10-10.5=-0.5$ .
<code>select remainder(-10,3.5) from dual</code>	0.5 <b>Explicație.</b> Cel mai apropiat de -10 multiplu a lui 3.5 este -10.5, iar $-10-(-10.5)=0.5$ .

- **MOD(x,y)** – funcția returnează restul împărțirii lui **x** la **y**. Teorema împărțirii cu rest este extinsă de această funcție și pentru numerele reale. Adică se ține cont de relația:

$$x=y * \text{cât} + \text{rest}$$

unde restul trebuie să fie în modul strict mai mic decât **y**.

Exemplu	Rezultatul afișat
<code>select mod(10,3) from dual</code>	1 <i>Explicație. 10=3*3+1.</i>
<code>select mod(5,3) from dual</code>	2 <i>Explicație. 5=3*1+2</i>
<code>select mod(10,3.5) from dual</code>	3 <i>Explicație. 10=3.5*2+3.</i>
<code>select mod(-10,3.5) from dual</code>	-3 <i>Explicație. -10=3.5*(-2)-3.</i>
<code>select mod(-10,-3.5) from dual</code>	-3 <i>Explicație. -10=-3.5*2-3.</i>
<code>select mod(10,-3.5) from dual</code>	3 <i>Explicație. 10=-3.5*(-2)+3.</i>

Se observă din exemplele anterioare că restul are întotdeauna același semn cu primul parametru.

- **SIGN(x)** – returnează semnul lui **x**, adică 1 dacă **x** este număr pozitiv, respectiv -1 dacă **x** este număr negativ.
- **CEIL(x)** – returnează cel mai mic număr întreg care este mai mare sau egal decât parametrul transmis.
- **FLOOR(x)** – returnează cel mai mare număr întreg care este mai mic sau egal decât parametrul transmis.

Exemplu	Rezultatul afișat
<code>select ceil(3) from dual</code>	3
<code>select ceil(-3) from dual</code>	-3
<code>select ceil(-3.7) from dual</code>	-3
<code>select ceil(3.7) from dual</code>	4
<code>select floor(3) from dual</code>	3
<code>select floor(-3) from dual</code>	-3
<code>select floor(-3.7) from dual</code>	-4
<code>select floor(3.7) from dual</code>	3

- **ROUND(x,y)** – rotunjește valoarea lui **x** la un număr de cifre precizat prin parametrul **y**.

Dacă al doilea parametru este un număr pozitiv, atunci se vor păstra din **x** primele **y** zecimale, ultima dintre aceste cifre fiind rotunjită, în funcție de următoarea zecimală.

Al doilea argument poate fi o valoare negativă, rotunjirea făcându-se la stânga punctului zecimal. Cifra a  $|y|+1$  din fața punctului zecimal (numărând de la punctul zecimal spre stânga, începând cu 1) va fi rotunjită în funcție cifra aflată imediat la dreapta ei. Primele  $|y|$  cifre din stânga punctului zecimal vor deveni 0.

Cel de al doilea argument este opțional, în cazul în care nu se precizează, este considerată implicit valoarea 0.

Exemplu	Rezultatul afișat
<code>select round(745.123,2) from dual</code>	745.12
<code>select round(745.126,2) from dual</code>	745.13
<code>select round(745.126,-1) from dual</code>	750
<code>select round(745.126,-2) from dual</code>	700
<code>select round(745.126,-3) from dual</code>	1000
<code>select round(745.126,-4) from dual</code>	0
<code>select round(745.126,0) from dual</code>	745
<code>select round(745.826,0) from dual</code>	746
<code>select round(745.826) from dual</code>	746

- **TRUNC(x)** – este asemănătoare cu funcția **ROUND**, fără a rotunji ultima cifră.

Exemplu	Rezultatul afișat
<code>select trunc(745.123,2) from dual</code>	745.12
<code>select trunc(745.126,2) from dual</code>	745.12
<code>select trunc(745.126,-1) from dual</code>	740
<code>select trunc(745.126,-2) from dual</code>	700
<code>select trunc(745.126,-3) from dual</code>	0
<code>select trunc(745.126,-4) from dual</code>	0
<code>select trunc(745.126,0) from dual</code>	745
<code>select trunc(745.826,0) from dual</code>	745
<code>select trunc(745.826) from dual</code>	745



## II.2.5. Funcții asupra datelor calendaristice

Una dintre caracteristicile importante ale Oracle este abilitatea de a memora și opera cu date calendaristice. Tipurile de date calendaristice recunoscute de Oracle sunt:

- **DATE** - valorile având acest tip sunt memorate într-un format intern specific, care include pe lângă ziua, luna și anul, de asemenea ora, minutul, și secunda.
- **TIMESTAMP** – valorile având acest tip memorează data calendaristică, ora, minutul și secunda dar și fracțiunea de secundă.
- **TIMESTAMP WITH [LOCAL] TIME ZONE** – este similar cu **TIMESTAMP**, însă se va memora și diferența de fus orar față de ora universală, a orei de pe server-ul bazei de date, sau a aplicației client, în cazul în care se include opțiunea **LOCAL**.
- **INTERVAL YEAR TO MONTH** – memorează o perioadă de timp în ani și luni.
- **INTERVAL DAY TO SECOND** – memorează un interval de timp în zile, ore, minute și secunde.

Să exemplificăm aceste tipuri de date creând o tabelă de test cu comanda:

```
create table test3
  (data1 DATE, data2 TIMESTAMP(5),
   data3 TIMESTAMP(5) WITH TIME ZONE,
   data4 TIMESTAMP(5) WITH LOCAL TIME ZONE)
```

Vom insera acum o linie nouă în această tabelă:

```
insert into test3
  values(sysdate,systimestamp,systimestamp,systimestamp)
```

și la afișarea tabelii

```
select * from test3
```

vom obține rezultatul din figura II.2.3.

DATA1	DATA2	DATA3	DATA4
27-FEB-07	27-FEB-07 05.49.35.02886 AM	27-FEB-07 05.49.35.02886 AM -06:00	27-FEB-07 11.49.35.02886 AM

**Figura II.2.3.** Rezultat

## Aritmetica datelor calendaristice

Oracle știe să realizeze operații aritmetice asupra datelor calendaristice, astfel adăugarea valorii 1 la o dată calendaristică, va duce la obținerea următoarei date calendaristice:

```
SELECT sysdate, sysdate+5, sysdate-70 from dual
```

SYSDATE	SYSDATE+5	SYSDATE-70
21-APR-07	26-APR-07	10-FEB-07

**Figura II.2.4.** Adunarea unui număr întreg la o dată calendaristică

De asemenea, se poate face diferența dintre două date calendaristice, obținându-se numărul de zile dintre cele două date:

```
SELECT first_name, last_name,
       hire_date, sysdate-hire_date
FROM employees
```

FIRST_NAME	LAST_NAME	HIRE_DATE	SYSDATE-HIRE_DATE
Steven	King	17-JUN-87	7248.1856597222222222222222222222...
Neena	Kochhar	21-SEP-89	6421.1856597222222222222222222222...
Lex	De Haan	13-JAN-93	5211.1856597222222222222222222222...
Alexander	Hunold	03-JAN-90	6317.1856597222222222222222222222...
...	...	...	...

**Figura II.2.5.** Diferența dintre două date calendaristice

Deși implicit o dată calendaristică de tip **DATE** nu este afișată în format complet (nu se afișează ora, minutul, secunda), în tabelă se memorează complet. De aceea poate fi uneori derutant rezultatul unor operații aritmetice cu date calendaristice, după cum se vede în figura II.2.6, în care diferența dintre ziua de astăzi și cea de ieri este de 1.187997....

```
SELECT sysdate-TO_DATE('20-APR-07','dd-MON-yy') FROM dual
```

SYSDATE-TO_DATE('20-APR-07','DD-MON-YY')
1.18799768518518518518518518518519

**Figura II.2.6.**

De ce se obține acest lucru? Simplu, data de 20 aprilie a fost precizată fără oră, așadar a fost considerată implicit ora 00:00. Iar **sysdate** ne-a furnizat data curentă incluzând și ora. Așadar de ieri de la ora 00:00 până astăzi la ora 12:32 a trecut mai mult de o zi.

## Funcții cu date calendaristice

Oracle oferă un număr foarte mare de funcții care operează asupra datelor calendaristice, dar în cele ce urmează ne vom opri asupra celor mai importante dintre acestea.

- **SYSDATE** – returnează data și ora curentă a server-ului bazei de date.
- **CURRENT\_DATE** – returnează data și ora curentă a aplicației client. Aceasta poate să difere de data bazei de date.
- **SYSTIMESTAMP** – returnează data în formatul **TIMESTAMP**.

```
select CURRENT_DATE, sysdate, systimestamp
from dual
```

CURRENT_DATE	SYSDATE	SYSTIMESTAMP
21-APR-07	21-APR-07	21-APR-07 04.33.32.445081 AM -05:00

**Figura II.2.7.** Funcțiile **SYSDATE**, **CURRENT\_DATE** și **SYSTIMESTAMP**

- **ADD\_MONTHS(data,nrluni)** – adaugă un număr de luni la data curentă. Dacă al doilea parametru este un număr negativ, se realizează de fapt scăderea unui număr de luni din data precizată.

Exemplu	Rezultatul afișat
select sysdate, ADD_MONTHS(sysdate,2) from dual	27-FEB-07 27-APR-07
select sysdate, ADD_MONTHS(sysdate,-2) from dual	27-FEB-07 27-DEC-07

- **MONTHS\_BETWEEN(data1,data2)** – determină numărul de luni dintre două date calendaristice precizate. Rezultatul returnat poate fi un număr real (vezi figura II.2.8). Dacă prima dată este mai mică (o dată mai veche) atunci rezultatul va fi un număr negativ.

```
select sysdate, hire_date,  
MONTHS_BETWEEN(sysdate, hire_date),  
MONTHS_BETWEEN(hire_date, sysdate)  
from employees
```

SYSDATE	HIRE_DATE	MONTHS_BETWEEN( SYSDATE, HIRE_DATE)	MONTHS_BETWEEN( HIRE_DATE, SYSDATE)
21-APR-07	17-JUN-87	238.13 ...	-238.13 ...
21-APR-07	21-SEP-89	211	-211
21-APR-07	13-JAN-93	171.26 ...	-171.26 ...
21-APR-07	03-JAN-90	207.58 ...	-207.58 ...
21-APR-07	21-MAY-91	191	-191
...	...	...	...

**Figura II.2.8.** Funcția **MONTHS\_BETWEEN**

- **LEAST(data1,data2,...)** – determină cea mai veche (cea mai mică) dată dintre cele transmise ca parametru.
- **GREATEST(data1,data2,...)** – determină cea mai recentă (cea mai mare) dată dintre cele transmise ca parametru.

```
select hire_date,sysdate,
       least(hire_date,sysdate),greatest(hire_date,sysdate)
from employees
```

HIRE_DATE	SYSDATE	LEAST(HIRE_DATE, SYSDATE)	GREATEST( HIRE_DATE,SYSDATE)
17-JUN-87	21-APR-07	17-JUN-87	21-APR-07
21-SEP-89	21-APR-07	21-SEP-89	21-APR-07
13-JAN-93	21-APR-07	13-JAN-93	21-APR-07
03-JAN-90	21-APR-07	03-JAN-90	21-APR-07
21-MAY-91	21-APR-07	21-MAY-91	21-APR-07
...	...	...	...

**Figura II.2.9. Funcțiile LEAST și GEATEST**

- **NEXT\_DAY(data, 'ziua')** – returnează următoarea dată de 'ziua' de după data transmisă ca parametru, unde 'ziua' poate fi 'Monday', 'Tuesday' etc. În exemplele care urmează, data curentă este considerată ziua de marți, 27 februarie 2007.
- **LAST\_DAY(data)** – returnează ultima zi din luna din care face parte data transmisă ca parametru.

Exemplu	Rezultatul afișat
<code>select next_day(sysdate,'Friday') from dual</code>	02-MAR-07
<code>select next_day(sysdate,'TUESDAY') from dual</code>	06-MAR-07 <b>Explicație.</b> Chiar dacă ziua curentă este o zi de marți, funcția va returna următoarea zi de marți.
<code>select last_day(sysdate) from dual</code>	28-FEB-07
<code>select last_day(sysdate+20) from dual</code>	31-MAR-07
<code>select last_day(ADD_MONTHS(sysdate,12)) from dual</code>	29-FEB-07 <b>Explicație.</b> Ziua returnată de <code>sysdate</code> este 27-FEB-07, la care adăugăm 12 luni, deci obținem data de 27-FEB-08, iar anul 2008 este un an bisect de aceea ultima zi din lună este 29-FEB-08.

- **ROUND(data,'format')** – dacă nu se precizează formatul, funcția rotunjește data transmisă ca parametru la cea mai apropiată oră **12 AM**, adică dacă ora memorată în **data** este înainte de miezul zilei atunci se va returna ora **12 AM** a datei transmise. Dacă ora memorată în **data** este după miezul zilei se va returna ora **12 AM** a zilei următoare.

```
select to_char(sysdate,'dd-MON-YY hh:mi AM'),
       round(sysdate) from dual
```

TO_CHAR(SYSDATE,'DD-MON-YYHH:MIAM')	ROUND(SYSDATE)
21-APR-07 04:41 AM	21-APR-07

**Figura II.2.10. Funcția ROUND**

În cazul în care este specificat formatul, data va fi rotunjită conform formatului indicat. Câteva dintre formatele cele mai uzuale sunt:

- **y, yy, yyyy, year** – se rotunjește data la cea mai apropiată dată de 1 ianuarie. Dacă data este înainte de 1 iulie, se va returna data de 1 ianuarie a aceluiași an. Dacă data este după data de 1 iulie se va returna data de 1 ianuarie a anului următor.
- **mm, month** – rotunjește data la cel mai apropiat început de lună. Orice dată calendaristică aflată după data de 16 inclusiv, este rotunjită la prima zi a lunii următoare.
- **ww, week** – se rotunjește data la cel mai apropiat început de săptămână. Prima zi a săptămânii este considerată luni. Pentru datele aflate după ziua de joi inclusiv, se va returna ziua de luni a săptămânii următoare.

Exemplu	Rezultatul afișat
select sysdate, round(sysdate,'year'), round(ADD_MONTHS(sysdate,5),'year') from dual	27-FEB-07 01-JAN-07 01-JAN-08
select sysdate, round(sysdate,'mm'), round(sysdate+16,'mm'), round(sysdate+17,'mm') from dual	27-FEB-07 01-MAR-07 01-MAR-07 01-APR-07
select sysdate, round(sysdate,'ww'), round(sysdate+1,'ww'), round(sysdate+2,'ww') from dual	27-FEB-07 26-FEB-07 26-FEB-07 05-FEB-07

- **TRUNC(data, 'format')** – trunchiază data specificată conform formatului specificat. Se pot folosi aceleași formate ca și în cazul funcției **ROUND**.

Exemplu	Rezultatul afișat
<pre>select sysdate,        trunc(sysdate, 'year'),        trunc(ADD_MONTHS(sysdate, 5), 'year') from dual</pre>	<pre>27-FEB-07 01-JAN-07 01-JAN-07</pre>
<pre>select sysdate,        trunc(sysdate, 'month'),        trunc(sysdate+16, 'month'),        trunc(sysdate+17, 'month') from dual</pre>	<pre>27-FEB-07 01-FEB-07 01-MAR-07 01-MAR-07</pre>
<pre>select sysdate,        trunc(sysdate, 'ww'),        trunc(sysdate+1, 'ww'),        trunc(sysdate+2, 'ww') from dual</pre>	<pre>27-FEB-07 26-FEB-07 26-FEB-07 26-FEB-07</pre>

## II.2.6. Funcții de conversie

Oracle oferă un set bogat de funcții care vă permit să transformați o valoare dintr-un tip de dată în altul.

### Transformarea din dată calendaristică în șir de caractere

Transformarea unei date calendaristice în șir de caractere se poate realiza cu ajutorul funcției **TO\_CHAR**. Această operație se poate dovedi utilă atunci când dorim obținerea unor rapoarte cu un format precis. Sintaxa acestei funcții este:

**TO\_CHAR (dt, format)**

**dt** poate avea unul dintre tipurile pentru date calendaristice (**DATE**, **TIMESTAMP**, **TIMESTAMP WITH TIME ZONE**, **TIMESTAMP WITH LOCAL TIME ZONE**, **INTERVAL MONTH TO YEAR** sau **INTERVAL DAY TO SECOND**). Formatul poate conține mai mulți parametri care pot afecta modul în care va arăta șirul returnat. Câțiva dintre acești parametri sunt prezentați în continuare.

Aspect	Parametru	Descriere	Exemplu
Secolul	<b>CC</b>	Secolul cu două cifre	21
Trimestrul	<b>Q</b>	Trimestrul din an în care se găsește data	3

Aspect	Parametru	Descriere	Exemplu
Anul	<b>YYYY, RRRR</b>	Anul cu patru cifre.	<b>2006</b>
	<b>YY, RR</b>	Ultimele două cifre din an.	<b>06</b>
	<b>Y</b>	Ultima cifră din an	<b>6</b>
	<b>YEAR, Year</b>	Numele anului	<b>TWO THOUSAND-SIX, Two Thousand-Six</b>
Luna	<b>MM</b>	Luna cu două cifre	<b>02</b>
	<b>MONTH, Month</b>	Numele complet al lunii.	<b>JANUARY, January</b>
	<b>MON, Mon</b>	Primele trei litere ale denumirii lunii.	<b>JAN, Jan</b>
	<b>RM</b>	Luna scrisă cu cifre romane.	<b>IV</b>
Săptămâna	<b>WW</b>	Numărul săptămânii din an.	<b>35</b>
	<b>W</b>	Ultima cifră a numărului săptămânii din an.	<b>2</b>
Ziua	<b>DDD</b>	Numărul zilei din cadrul anului.	<b>103</b>
	<b>DD</b>	Numărul zilei din cadrul lunii	<b>31</b>
	<b>D</b>	Numărul zilei din cadrul săptămânii.	<b>5</b>
	<b>DAY, Day</b>	Numele complet al zilei din săptămână	<b>SATURDAY, Saturday</b>
	<b>DY, Dy</b>	Prescurtarea denumirii zilei din săptămână.	<b>SAT, Sat</b>
Ora	<b>HH24</b>	Ora în formatul cu 24 de ore.	<b>23</b>
	<b>HH</b>	Ora în formatul cu 12 ore.	<b>11</b>
Minutele	<b>MI</b>	Minutele cu două cifre	<b>57</b>
Secunde	<b>SS</b>	Secundele cu două cifre	<b>45</b>
Sufixe	<b>AM sau PM</b>	<b>AM</b> sau <b>PM</b> după cum e cazul.	<b>AM</b>
	<b>A.M. sau P.M.</b>	<b>A.M.</b> sau <b>P.M.</b> după cum e cazul.	<b>P.M.</b>
	<b>TH</b>	Sufix pentru numerale ( <b>th</b> sau <b>nd</b> sau <b>st</b> )	
	<b>SP</b>	Numerele sunt scrise în cuvinte.	

În cadrul formatului se pot folosi oricare dintre următorii separatori:

- / , . ; :

Dacă în șirul returnat dorim să includem și anumite texte, acestea se vor include între ghilimele.

Iată în continuare și câteva exemple de folosire a acestei funcții.

Exemplu	Rezultatul afișat
<pre>select sysdate,   to_char(sysdate,'MONTH DD, YYYY')   to_char(sysdate,'Month DD, YYYY')   to_char(sysdate,'Mon DD, YYYY') from dual</pre>	<pre>28-FEB-07 FEBRUARY 28, 2007 February 28, 2007 Feb 28, 2007</pre>
<pre>select   to_char(sysdate,'"Trimestrul "Q "al   anului " Year') from dual</pre>	<pre>Trimestrul 1 al anului Two Thousand Seven</pre>
<pre>select   to_char(sysdate,'"Secolul "CC') from dual</pre>	<pre>Secolul 21</pre>
<pre>select   to_char(sysdate,'Day, dd.RM.YYYY') from dual</pre>	<pre>Wednesday, 28.II.2007</pre>
<pre>select   to_char(sysdate,'Dy, D, DD, DDD') from dual</pre>	<pre>Wed, 4, 28, 059</pre>
<pre>select   to_char(sysdate,'HH24:MI/HH:MI AM') from dual</pre>	<pre>21:53/09:53 PM</pre>
<pre>select to_char(sysdate+1,'ddth') from dual</pre>	<pre>01st</pre>
<pre>select to_char(sysdate+1,'ddspth') from dual</pre>	<pre>first</pre>
<pre>select to_char(sysdate+2,'Ddspth') from dual</pre>	<pre>Second</pre>
<pre>select to_char(sysdate+10,'DDspth') from dual</pre>	<pre>TENTH</pre>
<pre>select to_char(sysdate,'mmsp') from dual</pre>	<pre>two</pre>



## Transformarea din șir de caractere în dată calendaristică

Folosind funcția `TO_DATE` se poate transforma un șir de caractere precum `'May 26, 2006'` într-o dată calendaristică. Sintaxa funcției este:

`TO_DATE(sir, format)`

Formatul nu este obligatoriu, însă dacă nu este precizat, șirul trebuie să respecte formatul implicit al datei calendaristice `DD-MON-YYYY` sau `DD-MON-YY`. Formatul poate folosi aceiași parametri de format ca și funcția `TO_CHAR`.

Exemplu	Rezultatul afișat
<pre>select to_date('7.4.07', 'MM.DD.YY') from dual;</pre>	04-JUL-07
<pre>select to_date('010101','ddmmyy') from dual</pre>	01-JAN-01

## Formatul RR și formatul YY

Așa cum s-a precizat anterior, în formatarea unei date calendaristice se pot folosi pentru an atât `YY` (respectiv `YYYY`) cât și `RR` (respectiv `RRR`). Diferența dintre aceste două formate este modul în care ele interpretează anii aparținând de secole diferite. Oracle memorează toate cele patru cifre ale unui an, dar dacă sunt transmise doar două dintre aceste cifre, Oracle va interpreta secolul diferit în cazul celor două formate.

Vom începe printr-un exemplu:

```
select to_char(to_date('05-FEB-95','DD-MON-YY'),
              'DD-MON-YYYY') as "YY Format",
       to_char(to_date('05-FEB-95','DD-MON-RR'),
              'DD-MON-RRRR') as "RR Format"
from dual
```

YY Format	RR Format
05-FEB-2095	05-FEB-1995

**Figura II.2.11.** Formatele `YY` și `RR`

Se observă modul diferit de interpretare a anului.

Dacă utilizați formatul `YY` și anul este specificat doar prin două cifre, se presupune că anul respectiv face parte din același secol cu anul curent. De

exemplu, dacă anul transmis este 15 iar anul curent este 2007, atunci anul transmis este interpretat cu 2015. De asemenea 75 este interpretat ca 2075.

```
select to_char(to_date('15','YY'),'YYYY'),
       to_char(to_date('75','YY'),'YYYY')
from dual
```

TO_CHAR(TO_DATE('15','YY'),'YYYY')	TO_CHAR(TO_DATE('75','YY'),'YYYY')
2015	2075

**Figura II.2.12. Formatul YY**

Dacă folosiți formatul **RR** și anul transmis este de două cifre, primele două cifre ale anului transmis este determinat în funcție de cele două cifre transmise și de ultimele două cifre ale anului curent. Regulile după care se determină secolul datei transmise sunt următoarele:

**Regula 1:** Dacă anul transmis este între 00 și 49, și ultimele două cifre ale anului curent sunt între 00 și 49 atunci secolul este același cu secolul anului curent. De exemplu, dacă anul transmis este 15, iar anul curent este 2007, anul transmis este interpretat ca fiind 2015.

**Regula 2:** Dacă anul transmis este între 50 și 99, iar anul curent este între 00 și 49 atunci secolul este secolul prezent minus 1. De exemplu, dacă transmiteți 75 iar anul curent este 2007, anul transmis este interpretat ca fiind 1975.

**Regula 3:** Dacă anul transmis este între 00 and 49 iar anul prezent este între 50 și 99, secolul este considerat secolul prezent plus 1. De exemplu dacă ați transmis anul 15, iar anul curent este 1987, anul transmis este considerat ca fiind anul 2015.

**Regula 4:** Dacă anul transmis este între 50 și 99, iar anul curent este între 50 și 99, secolul este același cu al anului curent. De exemplu, dacă transmiteți anul 55 iar anul prezent ar fi 1987, atunci anul transmis este considerat ca fiind anul 1955.

```
select to_char(to_date('04-JUL-15','DD-MON-RR'),
              'DD-MON-YYYY') as dt1,
       to_char(to_date('04-JUL-75','DD-MON-RR'),
              'DD-MON-YYYY') as dt2
from dual
```

DT1	DT2
04-JUL-2015	04-JUL-1975

**Figura II.2.13. Formatul RR**

## Transformarea din număr în șir de caractere

Pentru a transforma un număr într-un șir de caractere, se folosește funcția `TO_CHAR`, cu următoarea sintaxă:

`TO_CHAR(numar, format)`

Formatul poate conține unul sau mai mulți parametri de formatare dintre cei prezentați în tabelul următor.

Parametru	Exemplu de format	Descriere
<b>9</b>	999	returnează cifrele numărului din pozițiile specificate, precedat de semnul minus, dacă numărul este negativ
<b>0</b>	0999	completează cifrele numărului cu zerouri în față
<b>.</b>	999.99	specifică poziția punctului zecimal
<b>,</b>	9,999	specifică poziția separatorului virgulă
<b>\$</b>	\$999	afișează semnul dolar
<b>EEEE</b>	9.99EEEE	returnează scrierea științifică a numărului
<b>L</b>	L999	afișează simbolul monetar
<b>MI</b>	999MI	afișează semnul minus după număr dacă acesta este negativ
<b>PR</b>	999PR	numerele negative sunt închise între paranteze unghiulare
<b>RN</b> <b>rn</b>	RN rn	afișează numărul în cifre romane
<b>V</b>	99V99	afișează numărul înmulțit cu 10 la puterea <b>x</b> , și rotunjit la ultima cifră, unde <b>x</b> este numărul de cifre 9 de după <b>v</b>
<b>X</b>	XXXX	afișează numărul în baza 16

Vom exemplifica în continuare câteva dintre aceste formate.

Exemplu	Rezultatul afișat
<code>select to_char(123.45,'9999.99') from dual</code>	123.45
<code>select to_char(123.45,'0000.000') from dual</code>	0123.450
<code>select to_char(123.45,'9.99EEEE') from dual</code>	1.23E+02
<code>select to_char(-123.45,'999.999PR') from dual</code>	<123.450>

Exemplu	Rezultatul afișat
<code>select to_char(1.2373,'99999V99') from dual</code>	124
<code>select to_char(1.2373,'L0000.000') from dual</code>	\$0001.237
<code>select to_char(4987,'XXXXXX') from dual</code>	137B
<code>select to_char(498,'RN') from dual</code>	CDXCVIII

## Transformarea șir de caractere în număr

Transformarea inversă din șir de caractere într-o valoare numerică se realizează cu ajutorul funcției **TO\_NUMBER**:

**TO\_NUMBER(sir,format)**

Parametrii de formatare ce se pot folosi sunt aceiași ca în cazul funcției **TO\_CHAR**. Iată câteva exemple.

Exemplu	Rezultatul afișat
<code>select to_number('970.13') + 25.5 FROM dual</code>	995.63
<code>select to_number('- \$12,345.67',' \$99,999.99') from dual;</code>	-12345.67

## II.2.7. Funcții de uz general

Pe lângă funcțiile care controlează modul de formatare sau conversie al datelor, Oracle oferă câteva funcții de uz general, care specifică modul în care sunt tratate valorile **NULL**.

- **NVL(val1,val2)** – funcția returnează valoarea **val1**, dacă aceasta este nenulă, iar dacă **val1** este **NULL** atunci va returna valoarea **val2**. Funcția **NVL** poate lucra cu date de tip caracter, numeric sau dată calendaristică, însă este obligatoriu ca cele două valori să aibă același tip.

```
select first_name, commission_pct, NVL(commission_pct,0.8)
from employees
where employee_id between 140 and 150
```

Rezultatul returnat de această comandă este cel din figura II.2.14.

FIRST_NAME	COMMISSION_PCT	NVL(COMMISSION_PCT,0.8)
Trenna	-	.8
Curtis	-	.8
Randall	-	.8
Peter	-	.8
Eleni	.2	.2

**Figura II.2.14. Funcția NVL**

- **NVL2(val1,val2,val3)** – dacă valoarea **val1** nu este nulă, atunci funcția va returna valoarea **val2**, iar dacă **val1** are valoarea **NULL**, atunci funcția va returna valoarea **val3** (vezi figura II.2.15.).

```
select first_name, commission_pct,
       NVL2(commission_pct,'ARE','NU ARE')
from employees where employee_id between 140 and 150
```

FIRST_NAME	COMMISSION_PCT	NVL2(COMMISSION_PCT,'ARE','NUARE')
Trenna	-	NU ARE
Curtis	-	NU ARE
Randall	-	NU ARE
Peter	-	NU ARE
Eleni	.2	ARE

**Figura II.2.15 Funcția NVL2**

- **NULLIF(expr1,expr2)** – dacă cele două expresii sunt egale, funcția returnează **NULL**. Dacă valorile celor două expresii sunt diferite atunci funcția va returna valoarea primei expresii (vezi figura II.2.16.).

```
select employee_id, first_name, last_name,
       NULLIF(length(first_name),length(last_name))
from employees where employee_id between 103 and 142
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	NULLIF(LENGTH(FIRST_NAME), LENGTH(LAST_NAME))
103	Alexander	Hunold	9
104	Bruce	Ernst	-
107	Diana	Lorentz	5
124	Kevin	Mourgos	5
141	Trenna	Rajs	6
142	Curtis	Davies	-
...	...	...	...

**Figura II.2.16 Funcția NULLIF**

- **COALESCE(expr1, expr2, ..., exprn)** – funcția returnează valoarea primei expresii nenule (vezi figura II.2.17).

```
select coalesce(null, null, '33', 'test') from dual
```

COALESCE(NULL,NULL,'33','TEST')
33

Figura II.2.17 Funcția COALESCE

## II.2.8 Funcții și expresii condiționale

Oracle SQL oferă posibilitatea de a construi expresii alternative asemănătoare structurilor **IF-THEN-ELSE** prezente în alte limbaje.

- **DECODE(expresie, val11, val12, val21, val22, ..., valn1, valn2, val)** – aceasta compară valoarea expresiei cu valorile **val11, val21, ..., valn1**. Dacă valoarea expresiei este egală cu valoarea **vali1**, atunci funcția va returna valoarea **vali2**. Dacă funcția nu este egală cu nici una din valorile **vali1**, atunci funcția va returna valoarea **val**.

```
select DECODE('Maria' , 'Dana', 'Ea este Ana' ,
                  'Maria', 'Ea este Maria' ,
                  'Nu e nici Ana nici Maria')
from dual
```

Această comandă va afișa mesajul “Ea este Maria” însă următoarea comandă va afișa “Nu e nici Ana nici Maria”.

```
select DECODE('Valeria' , 'Dana', 'Ea este Ana' ,
                  'Maria', 'Ea este Maria' ,
                  'Nu e nici Ana nici Maria')
from dual
```

- În locul funcției **DECODE** se poate folosi expresia condițională **CASE**. Funcția **CASE** utilizează cuvintele cheie **when, then, else** și **end** pentru a indica ramura selectată. În general, orice apel al funcției **DECODE** poate fi transcris folosind funcția **CASE**. Chiar dacă o expresie folosind **CASE** este mai lungă decât expresia echivalentă care folosește funcția **DECODE**, varianta cu **CASE** este mult mai ușor de citit și greșelile sunt depistate mai ușor. În plus, varianta **CASE** este compatibilă **ANSI-SQL**.

Cele două comenzi de mai sus pot fi transcrise cu ajutorul funcției CASE astfel:

```
select CASE 'Maria'
        WHEN 'Dana' THEN 'Ea este Ana'
        WHEN 'Maria' THEN 'Ea este Maria'
        ELSE 'Nu e nici Ana nici Maria'
      END
from dual

select CASE 'Valeria'
        WHEN 'Dana' THEN 'Ea este Ana'
        WHEN 'Maria' THEN 'Ea este Maria'
        ELSE 'Nu e nici Ana nici Maria'
      END
from dual
```



## Aplicații

1. Ce vor afișa pe ecran următoarele comenzi?

- a) 

```
select SUBSTR('curious_george', -1)
from dual
```
- b) 

```
select SUBSTR('curious_george', 1, 7)
from dual
```
- c) 

```
select SUBSTR('curious_george', 9, 6)
from dual
```
- d) 

```
select SUBSTR ('curious_george', -8, 2)
from dual
```
- e) 

```
select SUBSTR('curious_george',
              INSTR ('curious_george', -1, ' ') + 1)
from dual
```
- f) 

```
select SUBSTR('curious_george',
              INSTR ('curious_george', ' ', -1, 3) + 1,
              LENGTH ('cute')) from dual
```
- g) 

```
select SUBSTR ('curious_george',
              -1 * LENGTH ('curious_george'))
from dual
```

2. În ce dată veți avea exact 10000 de zile (vârsta exprimată în zile să fie 10000)? În ce zi a săptămânii se va întâmpla acest lucru?

3. Afișați data nașterii tuturor angajaților din tabela **employees** în formatul **April 2nd, 1967**.

Întrebările 4-7 se vor referi la următoarele tabele despre mesajele postate pe un forum:

<b>Users</b>	<b>Posts</b>
- #UserId (number)	- #PostId (number)
- UserName (varchar2)	- UserId (number)
- Cost (numeric)	- GroupId (number)
	- ThreadId (number)
<b>Groups</b>	- PostText (varchar2)
- #GroupId (number)	- DateCreated (date)
- Title (varchar2)	
- Category (varchar2)	
- NumberOfPosts (number)	
- GroupSize (number)	
- Owner (number)	

4. Afișați id-urile grupurilor pentru care numele categoriei are lungimea cuprinsă între 6 și 9 caractere (inclusiv).

5. Pentru toate mesajele postate, afișați **postId**, și numărul de zile ce au trecut de când mesajul a fost postat. Denumiți cele două coloane afișate **postNumber** și respectiv **daysPast**.

6. Pentru toate mesajele afișate în data de 13 Aprilie a oricărui an afișați **groupId**-ul și un text astfel: dacă textul nu conține nici un caracter, veți afișa mesajul "text vid", dacă textul are 10 sau mai puține caractere (dar mai mult de 0 caractere) veți afișa mesajul "text scurt", în orice alt caz veți afișa mesajul "text lung".

7. Pentru fiecare grup care nu face parte din categoria "Sport" afișați câte o linie de forma:

**Grupul xxx face parte din categoria yyy.**

unde **xxx** este numele grupului, iar **yyy** este categoria căreia aparține grupul. Eliminați înainte de afișarea numelui grupului și a categoriei toate spațiile inutile de la începutul sau sfârșitul acestora.



1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ cum se pot prelua informații  
din mai multe tabele simultan
- ✓ care sunt tipurile de join  
existente
- ✓ cum se realizează fiecare tip  
de join folosind sintaxa  
Oracle
- ✓ cum se realizează fiecare tip  
de join folosind sintaxa ANSI
- ✓ cum se folosesc operatorii  
**UNION**, **INTERSECT**, **MINUS**

În capitolele anterioare am aflat cum putem afișa informații din baza de date, însă la fiecare rulare a unei comenzi **SELECT** am afișat date dintr-o singură tabelă.

Unul dintre rezultatele procesului de normalizare este acela că datele sunt memorate, de cele mai multe ori, în tabele diferite. De aceea, la afișarea diferitelor rapoarte va trebui să puteți prelua date din mai multe tabele printr-o singură comandă SQL.

Din fericire SQL oferă facilități pentru combinarea datelor din mai multe tabele și afișarea lor într-un singur raport. O astfel de operație se numește **join**, sau **interogare multiplă**.

Pe parcursul acestui capitol vom folosi ca exemple tabela **Persoane** a cărei cheie primară este atributul **IdPersoana**, tabela **Firme** a cărei cheie primară este atributul **IdFirm**, și tabela **Joburi** cu cheia primară **IdJob**. Presupunem că aceste tabele conțin următoarele înregistrări:

**Tabelul II.3.1. Tabela Persoane**

IDPERSONA	NUME	PRENUME	LOCALITATE	IDFIRM	IDJOB
1	Ionescu	Gheorghe	Brasov	22	5
2	Vasilescu	Vasile	Cluj-Napoca	15	1
3	Popescu	Ioan	Bucuresti	10	2
4	Georgescu	Maria	Iasi	30	6
5	Marinescu	Angela	Sibiu	-	3
6	Antonescu	Elena	Sibiu	10	1
7	Bischin	Paraschin	Brasov	15	-
8	Olaru	Angela	Ploiesti	22	2

**Tabelul II.3.2. Tabela Firme**

IdFirm	Nume	Localitate
10	SC Crisib SA	Sibiu
15	SC SoftCom	Alba Iulia
20	SC TimTip	Timisoara
22	Brasoveanca	Brasov

**Tabelul II.3.3. Tabela Joburi**

IdJob	Nume
1	Reprezentant Vanzari
2	Manager
6	Operator IT
3	Programator
4	Administrator
5	Administrator retea

În Oracle există două moduri diferite de a scrie join-urile:

- Prima metodă folosește sintaxa specifică Oracle. În acest caz condițiile de join sunt incluse în clauza **WHERE**. Această metodă este mai ușor de înțeles, însă are dezavantajul că în aceeași clauză

**WHERE** se includ atât condițiile de filtrare a înregistrărilor afișate cât și condițiile de join.

- A doua variantă folosește sintaxa ANSI/ISO, care este puțin mai greoaie, însă comenzile scrise folosind această sintaxă sunt portabile și în alte SGBD-uri care folosesc limbajul SQL.

Indiferent de sintaxa folosită există mai multe moduri de legare a tabelelor și anume:

- **Produsul cartezian** – leagă fiecare înregistrare dintr-o tabelă cu toate înregistrările din cealaltă tabelă.
- **Equijoin** – sunt legate două tabele cu ajutorul unei condiții de egalitate.
- **NonEquijoin** - în acest caz condiția de join folosește alt operator decât operatorul de egalitate.
- **SelfJoin** – este legată o tabelă cu ea însăși, e folosită de obicei în conjuncție cu relațiile recursive.
- **OuterJoin** – sunt o extensie a equijoin-ului, când pentru unele înregistrări dintr-o tabelă nu există corespondent în cealaltă tabelă, și dorim ca aceste înregistrări fără corespondent să fie totuși afișate.

## II.3.1. Produsul cartezian

### a) Sintaxa Oracle

După cum am precizat, acest tip de legătură între două tabele, va lega fiecare rând din prima tabelă cu fiecare rând din cea de a doua tabelă. De exemplu comanda:

```
SELECT p.num, p.prenume, f.num  
FROM persoane p, firme f
```

Va afișa următoarele informații:

**Tabelul II.3.4.** Produsul cartezian între tabelele **Persoane** și **Firme**

Nume	Prenume	Nume
Ionescu	Gheorghe	SC Crisib SA
Vasilescu	Vasile	SC Crisib SA
Popescu	Ioan	SC Crisib SA
Georgescu	Maria	SC Crisib SA
Marinescu	Angela	SC Crisib SA

Nume	Prenume	Nume
Antonescu	Elena	SC Crisib SA
Bischin	Paraschin	SC Crisib SA
Olaru	Angela	SC Crisib SA
Ionescu	Gheorghe	SC SoftCom
Vasilescu	Vasile	SC SoftCom
Popescu	Ioan	SC SoftCom
Georgescu	Maria	SC SoftCom
Marinescu	Angela	SC SoftCom
Antonescu	Elena	SC SoftCom
Bischin	Paraschin	SC SoftCom
Olaru	Angela	SC SoftCom
Ionescu	Gheorghe	SC TimTip
Vasilescu	Vasile	SC TimTip
Popescu	Ioan	SC TimTip
Georgescu	Maria	SC TimTip
Marinescu	Angela	SC TimTip
Antonescu	Elena	SC TimTip
Bischin	Paraschin	SC TimTip
Olaru	Angela	SC TimTip
Ionescu	Gheorghe	Brasoveanca
Vasilescu	Vasile	Brasoveanca
Popescu	Ioan	Brasoveanca
Georgescu	Maria	Brasoveanca
Marinescu	Angela	Brasoveanca
Antonescu	Elena	Brasoveanca
Bischin	Paraschin	Brasoveanca
Olaru	Angela	Brasoveanca

adică se obțin  $8 \times 4 = 32$  înregistrări (tabela persoane conține 8 înregistrări, tabela firme 4 înregistrări)

Este de remarcat notația **p.nume**, **p.prenume**, **f.nume**, precum și literele **p** și **f** care urmează după numele tabelelor din clauza **FROM**. Spunem că am definit un alias al fiecărei tabele. Am fost nevoiți să folosim acest alias, deoarece în ambele tabele există o coloană cu numele **nume** și dacă nu prefațăm numele acestei coloane cu alias-ul tabelii se va genera o ambiguitate pe care server-ul bazei de date nu va ști să o rezolve. Alias-ul tabelii este obligatoriu să-l folosim când două tabele conțin coloane cu același nume. În exemplul anterior coloana prenume nu este obligatoriu să o prefațăm cu alias-ul coloanei, așadar comanda anterioară poate fi scrisă și astfel:

```
SELECT p.num, prenum, f.num
FROM persoane p, firme f
```

Deci, produsul cartezian apare atunci când nu este precizată nici o condiție privind modul de legare al celor două tabele.

#### b) Sintaxa ANSI

Pentru a obține produsul cartezian, în sintaxa ANSI vom folosi clauza **CROSS JOIN** în cadrul clauzei **FROM** ca în exemplul următor.

```
SELECT p.num, p.prenum, f.num
FROM persoane p CROSS JOIN firme f
```

Rezultatul obținut va coincide cu cel obținut anterior.

## II.3.2. Equijoin

Oare cum procedăm dacă dorim să afișăm pentru fiecare persoană, numele firmei la care lucrează? Să vedem de exemplu cum aflăm numele firmei la care lucrează Ionescu Gheorghe. Ne uităm în tabela **persoane**, la valoarea din coloana **IdFirm**. Această valoare este 22. Apoi, în tabela **firm** căutăm firma având codul 22, și preluăm numele acestei firme din coloana **num**. Acest nume este Brasoveanca. Afirmăm că Ionescu Gheorghe lucrează la firma Brasoveanca. Deci a trebuit ca valoarea din coloana **IdFirm** din tabela **Persoane** să coincidă cu valoarea coloanei **IdFirm** din tabela **Firme**.

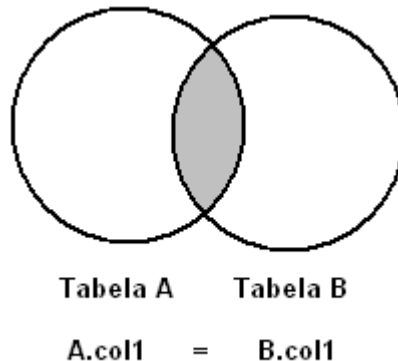
#### a) Sintaxa Oracle

Cum realizăm acest lucru folosind SQL? Simplu. Vom preciza condiția de egalitate dintre coloanele **IdFirm** din cele două tabele în clauza **WHERE** ca mai jos:

```
SELECT p.num, prenum, f.num
FROM persoane p, firme f
WHERE p.idfirm = f.idfirm
```

**Tabelul II.3.5.** Equijoin între tabelele **Persoane** și **Firme**

Nume	Prenume	Nume
Ionescu	Gheorghe	Brasoveanca
Vasilescu	Vasile	SC SoftCom
Popescu	Ioan	SC Crisib SA
Antonescu	Elena	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Olaru	Angela	Brasoveanca



**Figura II.3.1.** Equijoin

Bineînțeles că în condiția de equijoin pot fi precizate mai multe condiții. Dacă tabelele **elevi** și **note** ar conține următoarele coloane:

**Elevi** (#nume, #prenume, \*adresa)

**Note**(#nume, #prenume, #disciplina, #data, \*nota)

atunci pentru a afișa toate notele unui elev vom folosi comanda:

```
SELECT a.nume, a.prenume,
       b.disciplina, b.data, b.nota
FROM elevi a, firme b
WHERE a.nume=b.nume AND a.prenume=b.prenume
```

#### **b) Sintaxa ANSI**

În cazul sintaxei ANSI lucrurile se complică ușor. În principal equijoin-ul se realizează folosind opțiunea **NATURAL JOIN** în cadrul clauzei **from** astfel:

```
SELECT nume, prenume, nume
FROM persoane NATURAL JOIN firme
```

Însă dacă rulăm această comandă vom fi surprinși că ea nu afișează nici o linie. De ce? Pentru că **NATURAL JOIN**-ul leagă cele două tabele pe toate coloanele cu nume comun din cele două tabele. Adică, comanda anterioară este echivalentă cu următoarea comandă scrisă folosind sintaxa Oracle:

```
SELECT p.nume, prenume, f.nume
FROM persoane p, firme f
WHERE p.idfirm = f.idfirm AND p.nume=f.nume
```

ori nu are nici un sens să punem condiția ca numele firmei (**f.nume**) să coincidă cu numele persoanei (**p.nume**).

Reguli de folosire a opțiunii **NATURAL JOIN**:

- tabelele sunt legate pe toate coloanele cu nume comun
- coloanele cu nume comun trebuie să aibă același tip
- în clauza **SELECT** coloanele comune celor două tabele NU vor fi prefațate de alias-ul tabelului.

Pentru a lega două tabele folosind sintaxa ANSI dar condiția de egalitate să fie pusă doar pe anumite coloane (nu pe toate coloanele cu nume comun, ci doar pe o parte din acestea) se va folosi în loc de **NATURAL JOIN** clauza **JOIN**, iar coloanele pe care se face join-ul se precizează în opțiunea **USING**. Astfel comanda pentru afișarea firmelor la care lucrează fiecare angajat se scrie astfel:

```
SELECT p.num, prenume, f.num  
FROM personae p JOIN firme f  
USING (IdFirm)
```

Restricții la folosirea clauzei **JOIN** cu clauza **USING**:

- în clauza **USING** se trec în paranteză, separate prin virgulă, numele coloanelor pe care se va face join-ul;
- coloanele din clauza **USING** trebuie să aibă același tip în cele două tabele

Dacă în cele două tabele nu există coloane cu același nume sau coloanele cu nume comun au tipuri diferite în cele două tabele, se va folosi clauza **JOIN** în conjuncție cu **ON**. În clauza **ON** se poate trece orice condiție de join între cele două tabele.

```
SELECT p.num, prenume,  
       f.num  
FROM persoane p JOIN firme f  
ON (p.IdFirm=f.IdFirm)
```

Rezultatul obținut este același cu cel din tabelul II.3.5.

## II.3.3. Nonequijoin

### a) Sintaxa Oracle

Să presupunem că în tabela **Note** avem trecute mai multe note ale elevilor unei școli. Structura tabelului este

```
Note(#nume, #prenume, #disciplina, #data, *nota)
```

Dorim să înlocuim notele cu calificative și știm că notele de 9 și 10 sunt transformate în calificativul **FOARTE BINE**, notele de 7 și 8 în **BINE**, etc. Aceste echivalențe sunt memorate în tabela **CALIFICATIVE** cu structura următoare

```
CALIFICATIVE(#id, *nota1, *nota2, *calificativ)
```

cu semnificația că notele cuprinse între notele **nota1** și **nota2** inclusiv, se vor transforma în **calificativ**.

Pentru a scrie calificativele corespunzătoare fiecărei note din tabela **note**, vom scrie următoarea comandă:

```
SELECT nume, prenume, disciplina, data, calificativ  
FROM note, calificative  
WHERE nota BETWEEN nota1 AND nota2
```

#### **b) Sintaxa ANSI**

Echivalent vom scrie:

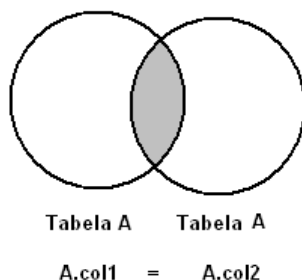
```
SELECT nume, prenume, disciplina, data, calificativ  
FROM note JOIN calificative  
ON (nota BETWEEN nota1 AND nota2)
```

### **II.3.4. Self Join**

Ținând cont de faptul că SelfJoin-ul este de fapt un equijoin dintre o tabelă și ea însăși, lucrurile sunt mult mai simple. Considerăm de exemplu, tabela **angajați** cu următoarea structură:

```
Angajați (#id, *nume, *prenume, *id_manager)
```

în câmpul **id\_manager** memorându-se codul șefului fiecărui angajat.



**Figura II.3.2. SelfJoin**



Dorim să afișăm numele fiecărui angajat și numele șefului acestuia. Vom folosi următoarele comenzi:

#### a) Sintaxa Oracle

```
SELECT a.num |'| ' |'| a.prenume AS "Angajat",  
       b.num |'| ' |'| b.prenume AS "Sef"  
FROM angajat a, angajat b  
WHERE a.id_manager = b.id
```

adică vom privi tabela **angajați** o dată ca tabelă de angajați (a) și apoi ca tabelă de manageri.

#### b) Sintaxa ANSI

```
SELECT a.num |'| ' |'| a.prenume AS "Angajat",  
       b.num |'| ' |'| b.prenume AS "Sef"  
FROM angajat a JOIN angajat b  
ON (a.id_manager = b.id)
```

## II.3.5. OuterJoin

Să privim pentru început la tabelul II.3.5, rezultatul rulării unei comenzi de equijoin. Se poate observa că lipsesc din acest tabel două persoane: **Georgescu** și **Marinescu**. De ce oare? Din tabelele II.3.1 și II.3.2 rezultă că **Georgescu** nu lucrează încă la nicio firmă, iar **Marinescu** este atribuit unui firme care nu există (poate încă nu există sau a fost desființată). Deci pentru acești doi angajați nu se poate găsi nici o înregistrare în tabela **Firme** pentru care condiția de equijoin să fie îndeplinită, și de aceea nu sunt afișați.

Dacă dorim totuși să afișăm toți angajații din tabela persoane, indiferent dacă lucrează sau nu la o firmă, va trebui să putem suplini cumva această lipsă de informații.

Pentru a indica lipsa de informații dintr-o tabelă, vom folosi secvența (+) imediat după numele coloanei din tabela respectivă din condiția de join din clauza **WHERE**.

De exemplu, următoarea comandă va afișa toate persoanele cu sau fără firmă corespunzătoare (în sintaxa Oracle):

```
SELECT a.num, a.prenume, b.num  
FROM persoane a, firme b  
WHERE a.IdFirm = b.IdFirm (+)
```

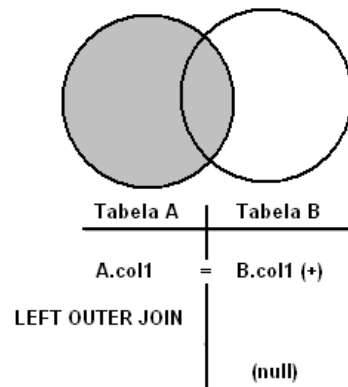
Rezultatul r  rii acestei comenzi este cel din tabelul II.3.6.

**Tabelul II.3.6.** Outer Join

Nume	Prenume	NumeFirma
Antonescu	Elena	SC Crisib SA
Popescu	Ioan	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Vasilescu	Vasile	SC SoftCom
Olaru	Angela	Brasoveanca
Ionescu	Gheorghe	Brasoveanca
Marinescu	Angela	-
Georgescu	Maria	-

Se observ   c   semnul (+) se g  se  te dup   coloana **IdFirm** din tabela **firme** (b). Aceast   tabel   fiind a doua tabel   din clauza **FROM**, vom spune c   este vorba de un **LEFT OUTER JOIN**, adic   sunt afi  ate toate   nregistr  rile din tabela din st  nga din clauza **FROM** cu sau f  r     nregistr  ri corespunz  toare   n tabela a doua. Sintaxa ANSI folose  te clauza **LEFT OUTER JOIN**   mpreun   cu **ON**. Comanda anterioar   este echivalent   cu urm  toarea comand     n sintaxa ANSI:

```
SELECT a.num  , a.prenume, b.num  
FROM persoane a LEFT OUTER JOIN firme b
ON (a.IdFirm = b.IdFirm)
```



**Figura II.3.3.** Left Outer Join

Dac   vom pune semnul (+)   n dreptul celeilalte tabele, adic   vom scrie:

```
SELECT a.num  , a.prenume, b.num  
FROM persoane a, firme b
WHERE a.IdFirm (+) = b.IdFirm
```

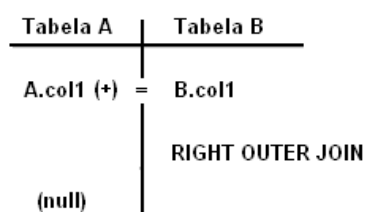
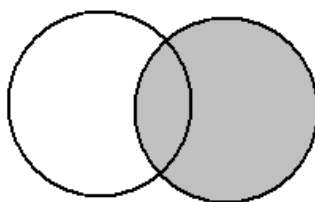
se vor afișa toate firmele, cu sau fără angajați, adică toate înregistrările din tabela aflată în dreapta în clauza **FROM** (firme), cu sau fără înregistrări corespunzătoare în cealaltă tabelă, adică cu sau fără angajați. Este așadar vorba despre un **RIGHT OUTER JOIN**. Astfel, în sintaxa ANSI vom scrie:

```
SELECT a.num, a.prenume, b.num
FROM persoane a RIGHT OUTER JOIN firme b
ON (a.IdFirm = b.IdFirm)
```

Rezultatul obținut va fi cel din tabelul II.3.7.

**Tabelul II.3.7.** Right Outer Join

Nume	Prenume	NumeFirma
Ionescu	Gheorghe	Brasoveanca
Vasilescu	Vasile	SC SoftCom
Popescu	Ioan	SC Crisib SA
Antonescu	Elena	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Olaru	Angela	Brasoveanca
-	-	SC TimTip



**Figura II.3.4.** Right Outer Join

**ATENȚIE:** este importantă ordinea tabelor în clauza **FROM**, nu ordinea în care sunt scrise cele două părți ale egalității din clauza **WHERE**, respectiv **ON**. Astfel, comenzile:

```
SELECT a.num, a.prenume, b.num
FROM persoane a, firme b
WHERE a.IdFirm = b.IdFirm (+)
```

și

```
SELECT a.nume, a.prenume, b.nume  
FROM persoane a, firme b  
WHERE b.IdFirm (+) = a.IdFirm
```

sunt echivalente și reprezintă un **LEFT OUTER JOIN**, chiar dacă semnul (+) apare o dată în stânga semnului de egalitate și o dată în dreapta semnului de egalitate.

De asemenea, deși următoarele două comenzi sunt echivalente:

```
SELECT a.nume, a.prenume, b.nume  
FROM persoane a, firme b  
WHERE a.IdFirm = b.IdFirm (+)
```

și

```
SELECT a.nume, a.prenume, b.nume  
FROM firme b, persoane a  
WHERE a.IdFirm = b.IdFirm (+)
```

prima este un **LEFT OUTER JOIN**, iar a doua este un **RIGHT OUTER JOIN**, pentru că se afișează toate înregistrările din tabela a (cea care nu are + în dreptul ei), tabelă care în prima comandă se găsește în stânga în clauza **FROM**, iar în a doua comandă se găsește în dreapta în clauza **FROM**.

V-ați putea întreba cum am putea să afișăm toate înregistrările din ambele tabele, indiferent dacă ele au sau nu corespondent în cealaltă tabelă. Am dori deci să obținem tabelul următor:

**Tabelul II.3.8. Full Outer Join**

Nume	Prenume	NumeFirma
Antonescu	Elena	SC Crisib SA
Popescu	Ioan	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Vasilescu	Vasile	SC SoftCom
Olaru	Angela	Brasoveanca
Ionescu	Gheorghe	Brasoveanca
Marinescu	Angela	-
Georgescu	Maria	-
-	-	SC TimTip

Apar atât persoanele care nu sunt încă angajate, sau a căror firmă nu mai există în baza de date, dar și firmele pentru care nu avem nici un angajat memorat în baza de date.

Am fi tentați să scriem:

```
SELECT a.num, a.prenume, b.num
FROM firme b, persoane a
WHERE a.IdFirm (+) = b.IdFirm (+)
```

adică să punem (+) în ambele părți ale semnului de egalitate pentru că avem de suplini lipsa de informații din ambele tabele. Însă **sintaxa Oracle nu permite acest lucru! Singura modalitate de a obține un FULL OUTER JOIN este de a folosi sintaxa ANSI:**

```
SELECT a.num, a.prenume, b.num
FROM persoane a FULL OUTER JOIN firme b
ON (a.IdFirm = b.IdFirm)
```

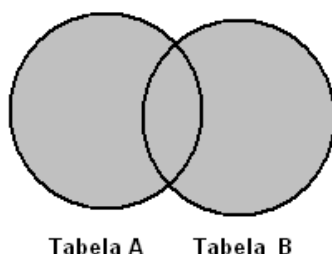


Figura II.3.5. Full Outer Join

Tabelul următor face o sinteză a comenzilor JOIN din acest capitol, punând față în față comenzile echivalente folosind cele două sintaxe.

Tabelul II.3.9. Comparație între sintaxa Oracle și sintaxa ANSI

Sintaxa Oracle	Sintaxa ANSI/ISO
<b>Produsul Cartezian</b>	
SELECT p.num, p.prenume, f.num FROM persoane p, firme f	SELECT p.num, p.prenume, f.num FROM persoane p <b>CROSS JOIN</b> firme f
<b>Equijoin</b>	
SELECT p.num, prenume, f.num FROM persoane p, firme f WHERE p.idfirm = f.idfirm	SELECT p.num, prenume, f.num FROM personae p <b>JOIN</b> firme f <b>USING</b> (IdFirm)
SELECT p.num, prenume, f.num FROM persoane p, firme f WHERE p.idfirm = f.idfirm AND p.num=f.num	SELECT num, prenume, FROM persoane p <b>NATURAL JOIN</b> firme f  NU AFIȘEAZĂ NIMIC !!!

Sintaxa Oracle	Sintaxa ANSI/ISO
SELECT a.num, a.prenume, b.disciplina, b.data, b.nota FROM elevi a, firme b WHERE a.num=b.num AND a.prenume=b.prenume	SELECT num, prenume, disciplina, data, nota FROM elevi <b>NATURAL JOIN</b> note
SELECT p.num, prenume, f.num FROM persoane p, firme f WHERE p.IdFirm=f.IdFirm	SELECT p.num, prenume, f.num FROM persoane p <b>JOIN</b> firme f <b>USING</b> (IdFirm)
<b>Nonequijoin</b>	
SELECT num, prenume, disciplina, data, calificativ FROM note, calificative WHERE nota BETWEEN nota1 AND nota2	SELECT num, prenume, disciplina, data, calificativ FROM note <b>JOIN</b> calificative <b>ON</b> (nota BETWEEN nota1 AND nota2)
<b>Selfjoin</b>	
SELECT a.num    ' '    a.prenume AS "Angajat", b.num    ' '    b.prenume AS "Sef" FROM angajat a, angajat b WHERE a.id_manager = b.id	SELECT a.num    ' '    a.prenume AS "Angajat", b.num    ' '    b.prenume AS "Sef" FROM angajat a <b>JOIN</b> angajat b <b>ON</b> (a.id_manager = b.id)
<b>Outer Join</b>	
SELECT a.num, a.prenume, b.num FROM persoane a, firme b WHERE a.IdFirm = b.IdFirm (+)	SELECT a.num, a.prenume, b.num FROM persoane a <b>LEFT OUTER JOIN</b> firme b <b>ON</b> (a.IdFirm = b.IdFirm)
SELECT a.num, a.prenume, b.num FROM persoane a, firme b WHERE a.IdFirm (+) = b.IdFirm	SELECT a.num, a.prenume, b.num FROM persoane a <b>RIGHT OUTER JOIN</b> firme b <b>ON</b> (a.IdFirm = b.IdFirm)
<b>NU EXISTA ECHIVALENT !</b>	SELECT a.num, a.prenume, b.num FROM persoane a <b>FULL OUTER</b> <b>JOIN</b> firme b <b>ON</b> (a.IdFirm = b.IdFirm)

## II.3.6. Operatorii UNION, INTERSECT, MINUS

Un caz mai special de interogare a mai multor tabele este acela în care combinăm rezultatele a două sau mai multe interogări independente una de cealaltă.

Operatorii folosiți în acest scop sunt:

**UNION ALL** – returnează toate liniile returnate de interogările pe care le leagă, inclusiv duplicatele (dacă cele două subinterogări returnează amândouă o aceeași linie, acest operator le va include pe ambele în rezultat).

**UNION** – asemănător cu operatorul anterior, însă sunt eliminate duplicatele;

**INTERSECT** – afișează liniile returnate de ambele interogări;

**MINUS** – returnează liniile care sunt returnate de prima interogare, dar nu sunt returnate și de a doua interogare.

**Atenție!** Numărul de coloane și tipul coloanelor returnate de cele două interogări trebuie să fie același, chiar dacă au alt nume.

Sintaxa folosirii acestor operatori este

**interogare operator interogare**

Vom exemplifica utilizarea lor pe două tabele formale

**Tabelul II.3.10.**  
Tabela A

ColA	ColB
A	10
A	15
B	7
C	20
C	30
D	40

**Tabelul II.3.11.**  
Tabela B

ColC	ColD
A	8
B	6
B	7
C	15
C	30
C	60
D	8

**Tabelul II.3.12.**  
Tabela C

ColE	ColF
A	10
B	6
C	20
D	8
E	10

Interogarea

**SELECT ColA, ColB FROM A**

**UNION ALL**

**SELECT ColC, ColD FROM B**

va afișa tabela II.3.13. Comanda următoare va elimina duplicatele, rezultatul fiind cel din tabela II.3.14.

```
SELECT ColA, ColB FROM A
UNION
SELECT ColC, ColD FROM B
```

**Tabelul II.3.13.** Utilizarea operatorului **UNION ALL**

COLA	COLB
A	10
A	15
B	7
C	20
C	30
D	40
A	8
B	6
B	7
C	30
C	15
C	60
D	8

**Tabelul II.3.14.** Utilizarea operatorului **UNION**

COLA	COLB
A	8
A	10
A	15
B	6
B	7
C	15
C	20
C	30
C	60
D	8
D	40

Similar, comenzile următoare vor afișa tabelul II.3.15 și respectiv II.3.16:

```
SELECT ColA, ColB FROM A
INTERSECT
SELECT ColC, ColD FROM B
```

și

```
SELECT ColA, ColB FROM A
MINUS
SELECT ColC, ColD FROM B
```

**Tabelul II.3.15.** Utilizarea operatorului **INTERSECT**

COLA	COLB
B	7
C	30

**Tabelul II.3.16.** Utilizarea operatorului **MINUS**

COLA	COLB
A	10
A	15
C	20
D	40

Un exemplu practic de folosire a acestor operatori poate fi dat dacă ne imaginăm că pentru cercul de informatică de la liceul vostru, profesorul coordonator de cerc a întocmit un tabel **info** conținând **numele**, **prenumele** și



`clasa` elevilor înscriși la acest cerc. Similar, profesorul de la cercul de matematică a realizat un tabel `mate` cu aceleași coloane, memorând elevii de la cercul de matematică.

Directorul școlii dorește o listă cu elevii înscriși la ambele cercuri. Nu aveți altceva de făcut decât să scrieți următoarea comandă:

```
SELECT nume, prenume, clasa FROM info
INTERSECT
SELECT nume, prenume, clasa FROM mate
```

Desigur puteți combina mai mult de două interogări folosind operatorii `UNION`, `INTERSECT` și `MINUS`. Implicit operatorii sunt evaluați de jos în sus, însă puteți indica ordinea de efectuare a acestor operații prin folosirea parantezelor. De exemplu, comanda:

```
SELECT colA, colB FROM A
UNION
SELECT colC, colD FROM B
INTERSECT
SELECT colE, colF FROM C
```

va returna tabelul II.3.17, în timp ce comanda

```
SELECT colA, colB FROM A
UNION
(SELECT colC, colD FROM B
INTERSECT
SELECT colE, colF FROM C)
```

va returna tabelul II.3.18.

**Tabelul II.3.17.**

COLA	COLB
A	10
B	6
C	20
D	8

**Tabelul II.3.18.**

COLA	COLB
A	10
A	15
B	6
B	7
C	20
C	30
D	8
D	40



## Test de evaluare

1. Următoarea comandă nu este scrisă corect. Care este principala eroare din această comandă?

```
SELECT clientNr, Nume
FROM comenzi, client
WHERE comenzi.clientNr = comenzi.comandaNr and
      data_livrare = '01-JAN-2001'
```

- a) Condiția de join este greșită
- b) Ordinea tabelurilor este greșită
- c) Cuvintele cheie nu sunt scrise toate cu majuscule
- d) Data nu respectă formatul corect.

2. Următoarea bază de date memorează datele meteorologice măsurate într-o perioadă de timp în stațiile meteorologice din România. Fiecare stație meteo se găsește într-o anumită regiune și fiecare stație înregistrează cantitatea de ploi căzute (în centimetri cubi) și numărul orelor de soare.

Tabela Regiuni

CodRegiune	Nume
1	Transilvania
2	Banat
3	Crișana

Tabela stații

IdStatie	Regiune	Ploaie	Soare
1	10	2	1
2	11	4	1
3	55	0	3
4	23	1	3
5	17	6	2
6	11	4	2
7	41	3	2

Următoarea comandă se dorește a afișa detaliile privind vremea înregistrată doar în Banat:

```
SELECT * FROM regiuni, stații
WHERE nume LIKE 'Banat'
```

Care dintre următoarele afirmații este corectă?

- a) Rezultatul va afișa stații pe care nu dorim să le afișăm
- b) Nu se va afișa nici o linie
- c) Se vor afișa linii nedorite
- d) Comanda va afișa rezultatul dorit.

3. Se dau următoarele tabele:

Tabela R

ColA	ColB
A	1
C	4
D	6
E	6

Tabela S

ColC	ColD
C	2
D	1
G	6
J	7

Câte linii va afișa următoarea comandă?

**SELECT \* FROM R FULL OUTER JOIN S  
ON (R.ColA = S.ColC)**

- a) 6                      b) 8                      c) 0                      d) 2

4. Fiind date tabelele

Tabela A

Col1	Col2
A	1
B	3
C	4

Tabela B

Col3	Col4
A	1
C	4
D	5
E	3

Care este rezultatul rulării următoarei comenzi?

**SELECT \* FROM A LEFT OUTER JOIN B  
ON (A.Col1 = B.Col3)**

a)

Col1	Col2	Col3	Col4
A	1	A	1
C	4	C	4
-	-	D	5
-	-	E	3

b)

Col1	Col2	Col3	Col4
A	1	A	1
B	3	-	-
C	4	C	4

c)

Col1	Col2	Col3	Col4
A	1	A	1
C	4	C	4

d)

Col1	Col2	Col3	Col4
A	1	A	1
B	3	-	-
C	4	C	4
-	-	D	5
-	-	E	3

5. Fiind date tabelele

Tabela A

Col1	Col2
A	1
B	3
C	4

Tabela B

Col3	Col4
A	1
C	4
D	5
E	3

Care este rezultatul rulării următoarei comenzi?

```
SELECT *
FROM A JOIN B ON (A.Col1 = B.Col3)
```

a)

Col1	Col2	Col3	Col4
A	1	A	1
B	3	-	-
C	4	C	4

b)

Col1	Col2	Col3	Col4
A	1	A	1
B	3	-	-
C	4	C	4
-	-	D	5
-	-	E	3

c)

Col1	Col2	Col3	Col4
A	1	A	1
C	4	C	4
-	-	D	5
-	-	E	3

d)

Col1	Col2	Col3	Col4
A	1	A	1
C	4	C	4

6. Se dau tabelele

Tabela Departments

Depno	Numedep
1	Computing
2	Electrical
3	Geografy
4	History
5	Business

Tabela Employees

Empno	Empname
1	Gordon
1	Ken
1	Brian
1	Colin
1	George

Tabela WorkFor

EmpNo	Depno
1	1
3	2
4	1
3	3
1	2
2	5

Care dintre următoarele tabele afișează toate departamentele cu persoanele angajate în aceste departamente?

a) `SELECT depname, empname  
FROM departments, workfor  
WHERE departments.depno =  
workfor.empno`

b) `SELECT depname, empname  
FROM departments,  
employees  
WHERE departments.depno =  
employees.empno`

c) `SELECT depno, empno  
FROM workfor`

a) `SELECT depname, empname  
FROM departments, workfor  
employees  
WHERE departments.depno=  
workfor.depno AND  
workfor.empno=employees.empno`

7. Dacă tabela c are 10 înregistrări, iar tabela d are 10 înregistrări, câte linii va afișa următoarea comandă?

```
SELECT a, b FROM c, d
```

a) 0

b) 10

c) 100

d) 1000

8. Se dau tabelele

Tabela P

ColW	ColX
A	4
B	5
C	6

Tabela Q

ColY	ColZ
B	7
D	4
C	6
E	9

Câte linii va afișa următoarea comandă?

```
SELECT * FROM P, Q
WHERE P.ColX (+) = Q.ColZ
```

- a) 7      b) 4      c) 3      d) 2

9. Considerăm baza de date conținând următoarele tabele:

```
FILME (id, titlu, an)
ACTORI (id, nume)
DISTRIBUȚII (filmId, actorId)
```

Care dintre comenzile următoare afișează titlurile tuturor filmelor în care a jucat Marilyn Monroe în anul 1959?

- a) 

```
SELECT titlu FROM filme, distribuții, actori
WHERE filmId = filme.id
AND actori.id = actorId
AND nume = 'Marilyn Monroe' AND an = 1959
```
- b) 

```
SELECT titlu FROM filme, actori
WHERE nume = 'Marilyn Monroe' AND an = 1959
```
- c) 

```
SELECT titlu FROM filme, distribuții, actori
WHERE filmId = filme.id
AND nume = 'Marilyn Monroe' AND an = 1959
```
- d) 

```
SELECT titlu FROM filme, distribuții, actori
WHERE filmId = filme.id
AND actori.id = actorId
AND filme.an = distribuții.an
AND nume = 'Marilyn Monroe' AND an = 1959
```

(vezi baremul de corectare și răspunsurile la pagina 312)



## Aplicații

1. Se consideră o bază de date cu următoarele coloane:

**Angajați** (**id**, **nume**, **salariu**)

**Angajări** (**idAngajat**, **IdDepartament**)

**Departamente** (**idDepartament**, **Nume**, **idManager**, **etaj**)

Scrieți câte o comandă pentru fiecare dintre următoarele cerințe:

a) Să se afișeze numele tuturor angajaților care lucrează la etajul 10 și câștigă mai puțin de 850.

b) Angajații din departamentul de jucării primesc o mărire de salariu de 10%. Afișați numele fiecărui angajat din departamentul de jucării și valoarea noului salariu.

c) Afișați numele angajaților care au salariul mai mic decât 100 sau mai mare decât 1000.

d) Afișați numele angajaților care câștigă mai mult de 200 și lucrează fie în departamentul video știe în departamentul de jucării.

e) Afișați numele tuturor angajaților care câștigă mai mult decât managerul departamentului în care lucrează.

2. Se consideră o bază de date cu următoarele coloane:

**Furnizori** (**fId**, **nume**, **oraș**)

**Componente** (**cId**, **nume**, **culoare**)

**Comenzi** (**fId**, **cId**, **cantitate**)

Scrieți câte o comandă pentru fiecare din următoarele cerințe:

a) Să se afișeze numele tuturor componentelor de culoare roșie care au fost comandate de la furnizori din Sibiu, Iași sau Brașov.

b) Afișați numele și orașul furnizorilor care au o comandă de mai mult de 150 de piese de culoare roșie sau verde.

c) Afișați numele furnizorilor din Craiova.

d) Afișați toate informațiile despre furnizorii care oferă componente de culoare verde.

e) Pentru fiecare comandă, pentru o componentă de culoare roșie, afișați cantitatea componentei și numele componentei.

3. Se consideră tabelele având următoarea structură:

```
Clase (codcls, nume, sala, etaj, profil, dirig)
Profesori (cod, nume, prenume, specializarea)
Incadrari (codprof, codcls, nr_ore)
Elevi (id, codcls, nume, prenume, adresa, telefon)
```

- a) Afișați profesorii clasei "IX B".
- b) Afișați numele tuturor colegilor de clasă ai elevei "Enescu Maria".
- c) Afișați clasele la care predă "Marinescu Ioan".
- d) Afișați numele tuturor profesorilor și clasele la care aceștia sunt diriginți. Se vor afișa toți profesorii, și cei care nu au dirigenție.
- e) Afișați numele profesorilor pentru a căror specializare nu mai există alt profesor în școală.
- f) Afișați clasele a căror sală se găsește la un etaj la care mai există cel puțin încă o clasă cu același profil. Pentru fiecare clasă se vor afișa numele, sala, etajul, profilul și numele dirigintelui.
- g) Afișați numele profesorilor care predau atât la clasa "IX A" cât și la clasa "IX B".
- h) Există vreun profesor în școală care predă la toate clasele?
- i) Afișați numele și prenumele tuturor profesorilor care predau mai mult de 2 ore la clasa în care învață "Enescu Maria".
- j) Afișați numele tuturor profesorilor de matematică (specializarea este "Matematica") și clasele la care aceștia predau.
- k) Afișați numele tuturor diriginților de la profilul "Matematică-Informatică".
- l) Afișați profesorii care predau matematică și informatică la clasa la care este diriginte "Marinescu Ioan".
- m) Afișați lista tuturor elevilor de la profilul "Matematică-Informatică", ordonați alfabetic după nume.
- n) Afișați clasele la care predau atât "Marinescu Ioan" cât și "Marinescu Anca".
- o) Afișați profesorii care nu au ore la clasele la care sunt diriginți.
- p) Afișați profesorii care predau cel puțin 3 ore la clasa la care sunt diriginți.
- r) Afișați numele profesorilor de matematică sau informatică care sunt diriginți la clase de "Matematică-Informatică".
- s) Afișați numele profesorilor de matematică sau informatică care predau la clase de "Matematică-Informatică".

4. Se consideră tabelele având următoarele coloane<sup>1</sup>:

course: courseno (pk), cname, cdate  
department: depno (pk), dname, location, head  
employee: empno (pk), surname, forenames,  
dob, address, depno (fk)  
jobhistory: empno (fk), position, startdate, enddate,  
salary  
empcourse: empno (fk), courseno (fk)

a) Afișați numele complet a tuturor angajaților care au urmat vreun curs de **Accounting** (numele cursului conține cuvântul **Accounting**).

b) Afișați numele tuturor angajaților care au urmat cel puțin un curs în anul 1988.

c) Afișați în ordine alfabetică, după numele departamentului, apoi după nume (**surname**) și apoi după prenume (**forenames**), poziția curentă a tuturor angajaților.

d) Afișați numele și prenumele tuturor angajaților care lucrează în același departament cu Matthew Brownlie.

e) Afișați codul și salariul angajaților care câștigă mai mult decât angajatul cu codul 16. În fiecare linie veți afișa atât salariul angajatului respective cât și salariul angajatului cu codul 16.

f) Afișați numele complet și poziția actuală a tuturor angajaților care au urmat un curs pe care l-a urmat și Robert Roberts.

g) Afișați numele complet al oricărui angajat care a început un job nou în aceeași zi cu Allan Robinson.

h) Afișați numele complet și poziția actuală a tuturor angajaților care lucrează în același departament cu Brian Murphy și sunt mai vechi decât acesta în departamentul respectiv.

i) Afișați numele complet al angajaților care în acest moment au același salariu cu Claire MacCallan. Ordonăți lista alfabetic după nume (**surname**).

---

<sup>1</sup> Întrebările de la această problemă sunt preluate de la adresa <http://db.grussell.org/sql/index.cgi> cu acordul domnului dr. Gordon Russell.



1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ care sunt funcțiile de grup și cum sunt ele folosite
- ✓ cum se pot grupa datele dintr-o tabelă
- ✓ care sunt regulile de folosire a clauzei **GROUP BY**
- ✓ cum se pot filtra grupurile folosind clauza **HAVING**
- ✓ care este diferența dintre clauzele **WHERE** și **HAVING**
- ✓ în ce ordine sunt executate clauzele **WHERE**, **GROUP BY** și **HAVING**



## II.4.1. Studiu de caz

1. Să ne imaginăm că tocmai a fost lansat în școala voastră un concurs între clase. Se va acorda un premiu acelei clase care va acumula cel mai mic număr de absențe nemotivate în decursul semestrului 2. Ce avem de făcut? Modelul conceptual al bazei de date necesare pentru rezolvarea acestei situații poate fi următorul:

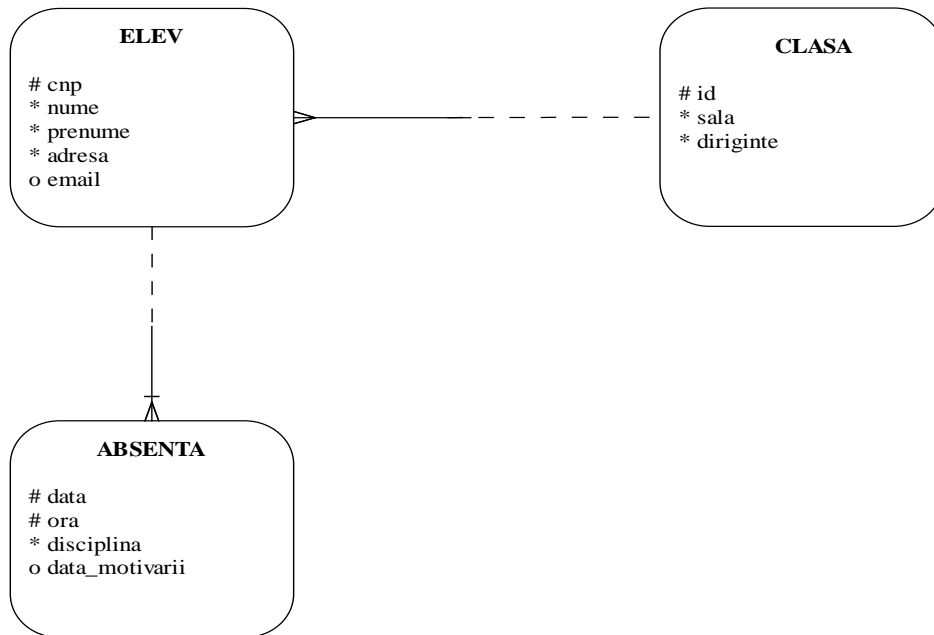


Figura II.4.1. Modelul conceptual al bazei de date

Având această structură de bază de date, va trebui să putem număra câte absențe nemotivate a acumulat fiecare elev în parte într-un anumit interval de timp, adică să numărăm câte înregistrări din tabela **ABSENTE**, pentru care câmpul **data\_motivării** a rămas necompletat corespund fiecărui elev în parte. Apoi pentru fiecare clasă în parte să facem o sumă a numărului de absențe astfel obținut. Întrebarea este cum numărăm câte înregistrări dintr-o tabelă respectă anumite criterii.

2. Tocmai au avut loc alegerile prezidențiale. S-au desfășurat buletinele de vot. Trebuie acum să numărăm în fiecare circumscripție electorală, câte voturi a primit fiecare candidat în parte. Va trebui să aflăm apoi câte voturi a primit în total fiecare candidat. A obținut vreun candidat majoritatea sau va avea loc un al doilea tur de scrutin? În ce circumscripții/localități/zone geografice a primit un candidat cele mai multe voturi, dar cele mai puține voturi? Aceste informații sunt foarte importante pentru candidați, pentru a putea să-și concentreze atenția în viitoarele

campanii electorale (eventual pentru al doilea tur de scrutin) în acele zone unde a obținut cele mai puține voturi.

Vom afla pe parcursul acestui capitol cum putem răspunde la astfel de întrebări.

## II.4.2. Funcții de grup

Într-un capitol anterior am discutat despre funcțiile singulare, adică despre funcțiile care operează la un moment dat asupra unei singure înregistrări.

Este acum momentul să discutăm despre funcțiile de grup, care returnează o singură valoare pentru un **grup sau set** de linii dintr-un tabel. Puteți calcula cea mai mare valoare dintr-un set de valori, puteți determina numărul de înregistrări ce respectă o anumită condiție etc.

Pentru exemplificarea acestor funcții vom folosi tabela **VOTURI** și tabela **JUDEȚE** care conțin următoarele date<sup>1</sup>.

**Tabelul II.4.1. Tabela VOTURI**

Judet	Candidat	Număr_voturi
B	1	347016
B	2	1552
B	3	1374
IS	1	196508
IS	2	1038
IS	3	1267
SB	1	65084
SB	2	561
SB	3	533
B	4	96744
B	5	25656
B	6	13361
IS	4	35784
IS	5	5558
IS	6	4094
SB	4	19937
SB	5	4323
SB	6	2366

---

<sup>1</sup> Valorile din acest tabel sunt reale (cu excepția ultimelor trei linii), au fost preluate de pe site-ul Institutului Național de Statistică (<http://www.INSSE.ro/cms/rw/pages/index.ro.do>) și se referă la alegerile prezidențiale din 2004 din județele Sibiu, Iași, București.

Judet	Candidat	Număr_voturi
B	7	25937
B	8	4619
B	9	4323
IS	7	3682
IS	8	1291
IS	9	327
SB	7	4225
SB	8	765
SB	9	3797
B	10	2037
B	11	22687
B	12	514366
IS	10	1312
IS	11	3781
IS	12	12184
SB	10	660
SB	11	3768
SB	12	105993
SB	13	100
B	13	(null)
IS	13	(null)

**Tabelul II.4.2. Tabela JUDETE**

Cod_județ	Județ	Număr_alegători
B	București	1750192
IS	Iași	650029
SB	Sibiu	363380

Vom prezenta în continuare principalele funcții de grup.

- **COUNT(x)** – determină numărul de valori ale lui **x**. Funcția, ca de altfel toate funcțiile de grup ignoră câmpurile completate cu **NULL**, adică va număra doar valorile nenule ale lui **x**.

De exemplu, comanda

```
SELECT COUNT(JUDET), COUNT(numar_voturi)
FROM voturi
```

va afișa numărul total de înregistrări din tabelă, 39 (câmpul **JUDET** nu are nici o valoare **NULL**) precum și numărul de linii pentru care câmpul **numar\_voturi** este nenul, adică 37, ultimele două linii din tabel având valoare **null** în câmpul **numar\_voturi**.

Tabelul II.4.3.

COUNT(JUDET)	COUNT(NUMAR_VOTURI)
39	37

Funcția **COUNT** poate fi folosită în combinație cu clauza **DISTINCT**, pentru a număra doar valorile distincte dintr-un domeniu. De exemplu dacă dorim să știm pentru câte județe avem rezultatele votării în tabela noastră, vom folosi comanda:

```
SELECT count(distinct judet)
FROM voturi
```

Se va obține valoarea 3, întrucât avem doar 3 județe înregistrate (București, Iași, Sibiu).

Tabelul II.4.4.

COUNT(DISTINCTJUDET)
3

Să vedem încă un exemplu:

```
SELECT count(distinct candidat), count(candidat)
FROM voturi
```

Evident primul apel de funcție afișează valoarea 13, deoarece există 13 candidați pentru care au fost exprimate voturi, iar a doua comandă afișează valoarea 39, adică exact numărul de linii din tabel deoarece toate liniile au completat câmpul **candidat**.

Tabelul II.4.5.

COUNT(DISTINCTCANDIDAT)	COUNT(CANDIDAT)
13	39

- **MAX(x)** – determină valoarea maximă a valorilor expresiei **x**.

Să vedem cum putem afla care este cel mai mare număr de voturi exprimate pentru un candidat într-un județ.

```
SELECT MAX(numar_voturi)
FROM voturi
```

Tabelul II.4.6.

MAX(NUMAR_VOTURI)
514366

Se poate observa pe tabelul cu datele din tabela voturi că acest maxim a fost obținut în București de către candidatul având codul 12.

Totuși această informație nu este foarte relevantă pentru că și populația din București este mult mai mare decât în celelalte județe. Ar trebui să putem determina numărul de voturi primite de către un candidat raportat la numărul de alegători (persoane cu drept de vot). SQL ne permite să aplicăm funcțiile de grup nu doar pe câmpuri din baza de date ci și pe expresii, ca în exemplul următor:

```
SELECT max(100*numar_voturi/numar_alegatori)
FROM voturi v, judete j
WHERE v.judet=j.cod_judet
```

**Tabelul II.4.7.**

MAX(100*NUMAR_VOTURI/NUMAR_ALEGATORI)
30.2306512478674028389502622190702260976

Prin această comandă am obținut cel mai mare procent de voturi obținut de către un candidat într-un județ. Acest procent a fost obținut raportat la totalul persoanelor cu drept de vot și a fost obținut de către candidatul cu codul 1 în județul Iași:

```
SELECT 100*numar_voturi/numar_alegatori,
        j.judet, v.candidat
FROM voturi v, judete j
WHERE v.judet=j.cod_judet
```

**Tabelul II.4.8.**

100*NUMAR_VOTURI/NUMAR_ALEGATORI	JUDET	CANDIDAT
19.8273103750902758097397314123250477662	Bucuresti	1
.088675985263331108815489957673215281523	Bucuresti	2
.078505672520500607933301032115333631967	Bucuresti	3
30.2306512478674028389502622190702260976	Iasi	1
...	...	...

În acest moment nu știm încă să scriem o comandă pentru a afișa județul și candidatul pentru care s-a obținut valoarea maximă, dar vom afla cum realizăm acest lucru în capitolul următor.

- **MIN(x)** – determină valoarea minimă a valorilor expresiei **x**.
- **SUM(x)** – determină suma valorilor expresiei **x**.

Cum aflăm oare numărul total de voturi valabil exprimate în județul Sibiu?  
Foarte simplu:

```
SELECT sum(numar_voturi)
FROM voturi
WHERE judet='SB'
```

Tabelul II.4.9.

SUM(NUMAR_VOTURI)
212112

- **AVG(x)** – determină media valorilor expresiei **x**. De exemplu, putem afla procentul mediu obținut de un candidat în toate județele:

```
SELECT avg(100*numar_voturi/numar_alegatori)
FROM voturi v, judete j
WHERE (candidat=12) and
      (v.judet=j.cod_judet)
```

Comanda afișează media procentelor obținute în fiecare județ de către candidatul cu codul 12:

Tabelul II.4.10.

AVG(100*NUMAR_VOTURI/NUMAR_ALEGATORI)
20.1440450845973468926087992135771906663

Am dori să afișăm un tabel cu procentele obținute de toți candidații, însă vom vedea cum realizăm acest lucru într-un paragraf următor.

După cum am precizat la funcția **COUNT**, funcțiile de grup, deci și **AVG** ignoră valorile **NULL**. Așadar dacă vom rula comanda:

```
SELECT avg(numar_voturi)
FROM voturi
WHERE candidat=13
```

vom obține valoarea 100, deși în baza de date există 3 linii pentru candidatul 13, și doar o linie are completat câmpul **numar\_voturi** cu valoarea 100. Dacă dorim să obținem valoarea 33.333, adică 100/3, vom scrie:

```
SELECT AVG(NVL(numar_voturi,0))
FROM voturi
WHERE candidat=13
```

adică înlocuim valorile **null** cu valoarea 0, pentru ca acestea să intre în calculul mediei.

- **STDEV(x)** – funcție statistică definită ca fiind abaterea pătratică a expresiei date. Cu cât valoarea funcției este mai mică cu atât valorile expresiei **x** sunt mai apropiate de medie.
- **VARIANCE(x)** – este o funcție statistică care calculează dispersia expresiei **x**. Se definește ca pătratul abaterii medii pătratice.

**Observație.** Funcțiile **COUNT**, **MIN**, **MAX** pot fi aplicate și datelor de tip șir de caractere sau date calendaristice, celelalte funcții fiind aplicabile doar valorilor numerice.

Comanda următoare va afișa data celei mai vechi angajări, data celei mai recente angajări, numărul de date de angajare și numărul de date distincte de angajare din tabela **employees**:

```
select min(hire_date), max(hire_date),
       count(distinct hire_date), count(hire_date)
from employees
```

**Tabelul II.4.11.**

MIN(HIRE_DATE)	MAX(HIRE_DATE)	COUNT(DISTINCT HIRE_DATE)	COUNT(HIRE_DATE)
17-JUN-87	29-JAN-00	19	20

## II.4.3. Gruparea datelor. Clauza GROUP BY

Uneori am putea dori să grupăm liniile dintr-o tabelă și să obținem anumite informații despre grupurile respective.

De exemplu am dori să calculăm numărul total de voturi obținut de fiecare candidat în toată țara. Cu ceea ce am învățat până acum, am putea rula o comandă de forma celei de mai jos pentru fiecare candidat în parte:

```
SELECT sum(numar_voturi)
FROM voturi
WHERE candidat=1
```

**Tabelul II.4.12.**

SUM(NUMAR_VOTURI)
608608

Însă această metodă nu este convenabilă, întrucât am dori să obținem un tabel cu toate aceste date, ca în tabelul II.4.13.

O astfel de grupare a datelor se poate face folosind clauza **GROUP BY**. Comanda care a fost rulată pentru a obține rezultatul din tabelul II.4.13, este:



```
SELECT candidat, sum(numar_voturi) AS "TOTAL VOTURI"
FROM voturi
GROUP BY candidat
```

**Tabelul II.4.13.**

CANDIDAT	TOTAL VOTURI
1	608608
2	3151
3	3174
4	152465
5	35537
6	19821
7	33844
8	6675
9	8447
10	4009
11	30236
12	632543
13	100

Se observă că pentru fiecare grup de înregistrări s-a obținut câte o singură valoare, adică pentru fiecare candidat am obținut o sumă a tuturor voturilor primite. De exemplu, candidatul cu codul 1 a obținut în București 347016 voturi, la Iași 196508 voturi, iar la Sibiu 65084 voturi, în total 608608 voturi adică exact valoarea din tabelele II.4.12 și II.4.13.

Clauza **GROUP BY** poate fi folosită și fără funcții de grup, doar pentru a afișa liniile grupate după un anumit criteriu, ca în exemplul următor:

```
SELECT candidat,numar_voturi FROM voturi
GROUP BY candidat, numar_voturi
```

**Tabelul II.4.14.**

CANDIDAT	NUMAR_VOTURI
1	65084
1	196508
1	347016
2	561
2	1038
2	1552
3	533
3	1267
3	1374
...	...

Să vedem cum aflăm procentul mediu obținut de către fiecare candidat.

```
SELECT candidat,AVG(100*numar_voturi/numar_alegatori)
FROM voturi v, judete j WHERE v.judet=j.cod_judet
GROUP BY candidat
```

**Tabelul II.4.15.**

CANDIDAT	AVG(100*NUMAR_VOTURI/NUMAR_ALEGATORI)
1	22.6562295618455989756920853154424336476
2	.13424833638246597421520567348011821838
3	.14003282051378316208662701165544554467
4	5.50638342911377223943294320779193667412
5	1.17019960040031154685700409684022462069
6	.68144301477468349708451524754117789898
7	1.07036088696521333741348008106908880327
8	.224347948245650587054654794284450480961
9	.447406194157174323863379832964865398693
10	.166617475745310934099468805148008754076
11	.97161839160269742583080767121400220691
12	20.1440450845973468926087992135771906663
13	.027519401177830370411139853596785733942

## Reguli de folosire a clauzei GROUP BY

- În clauza **GROUP BY** nu se acceptă aliasele coloanelor, comanda următoare va genera o eroare

```
SELECT department_id As Departament,
       job_id, MAX(salary)
FROM employees GROUP BY Departament, job_id
```

- toate câmpurile care apar în select în afara funcțiilor de grup trebuie să apară în clauza **GROUP BY** ca în exemplele de mai jos:

```
SELECT department_id, job_id, MAX(salary)
FROM employees GROUP BY department_id, job_id
```

sau

```
SELECT department_id, department_name, max(salary)
FROM employees NATURAL JOIN departments
GROUP BY department_id, department_name
```

sau

```
SELECT upper(last_name), sum(salary)
FROM employees GROUP BY last_name
```

Observați în acest ultim exemplu că deși în clauza **SELECT** câmpului **last\_name** îi este aplicată o funcție (simplă nu de grup!), în clauza **GROUP BY**, **last\_name** poate să apară fără funcția respectivă. Aveți grijă să nu confundați funcțiile singulare cu cele de grup!

- Nu se pot folosi funcții de grup în clauza **WHERE**. De aceea, următoarea comandă nu va putea fi rulată, ea generând o eroare:

```
SELECT * FROM voturi
WHERE numar_voturi=max(numar_voturi)
```

Pentru a putea afla ce candidat/candidați au obținut cele mai multe voturi vom folosi o subinterogare (asupra acestui subiect vom reveni în capitolul următor) astfel:

```
SELECT * FROM voturi
WHERE numar_voturi =
      (SELECT max(numar_voturi) from voturi)
```

- în clauza **GROUP BY** pot să apară și alte coloane care nu apar în **SELECT**

```
SELECT MAX(salary) FROM employees
GROUP BY departments
```

- funcțiile de grup pot fi imbricate ca în exemplul următor, în care am determinat cel mai mare număr total de voturi obținut de către un candidat.

```
SELECT max(sum(numar_voturi)) FROM voturi
GROUP BY candidat
```

**Tabelul II.4.16.**

MAX(SUM(NUMAR_VOTURI))
632543

## II.4.4. Selectarea grupurilor. Clauza **HAVING**

De multe ori nu ne interesează să afișăm toate grupurile de obținute prin folosirea clauzei **GROUP BY**. Pentru a filtra grupurile folosim clauza **HAVING**. Așa cum am văzut în exemplele anterioare, putem folosi clauza **GROUP BY** fără clauza **HAVING**, însă clauza **HAVING** poate fi folosită doar atunci când este prezentă clauza **GROUP BY**.

Să analizăm un exemplu. Să presupunem că dorim să afișăm toți candidații care au obținut un procent în alegeri mai mare de 5% din numărul total de persoane cu drept de vot. Pentru aceasta procedăm astfel:

- folosim clauza **GROUP BY** pentru a grupa liniile după candidați și calculăm pentru fiecare candidat procentul obținut:

```
SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON v.judet=j.cod_judet
GROUP BY candidat
```

**Tabelul II.4.17.**

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.0222817982769582150245277809640393096
2	.114017906347551618341432066351112190219
3	.114850153839139586358522811360974322994
4	5.51689625238954537938001904037522059082
5	1.28589474385050519231973067023785271463
6	.717216414381091915945898123499014510416
7	1.22463409153492128567039887451191398469
8	.24153269592824723974264012786216244675
9	.305651937454068080015892308621975458831
10	.145064356251137555674643336719012621576
11	1.09407978937625221585894635296484550411
12	22.8883619596316544971578748162270892216
13	.003618467354730295726481500042878838154

- Folosim clauza **HAVING** pentru a filtra grupurile care se vor afișa

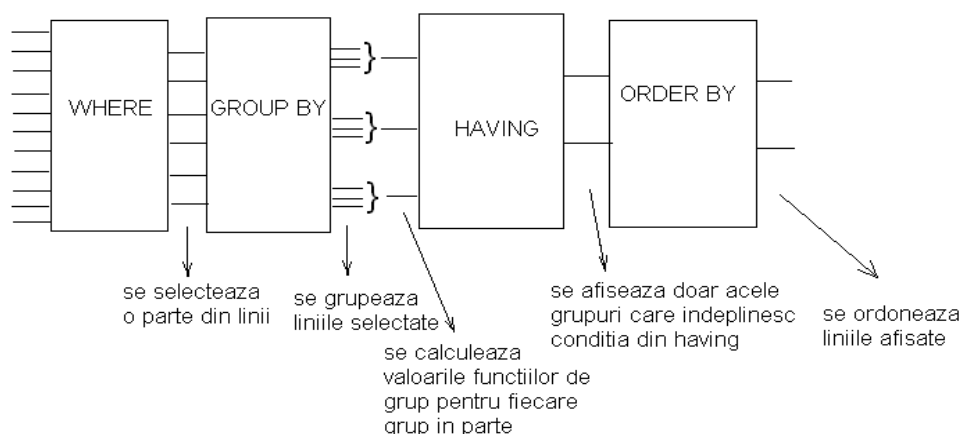
```
SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
GROUP BY candidat
HAVING 100*sum(numar_voturi)/sum(numar_alegatori)>5
```

**Tabelul II.4.18.**

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.0222817982769582150245277809640393096
4	5.51689625238954537938001904037522059082
12	22.8883619596316544971578748162270892216

Bineînțeles că putem folosi clauzele **WHERE**, **GROUP BY** și **HAVING** împreună. În acest caz, clauza **WHERE** va filtra mai întâi liniile din tabelă, liniile rămase vor fi grupate apoi conform criteriului dat de clauza **GROUP BY** și în final sunt afișate doar acele grupuri care respectă condiția dată de clauza **HAVING**. (figura II.4.2.)

Atenție! Trebuie făcută distincția clară dintre clauzele **WHERE** și **HAVING**. Clauza **WHERE** acționează asupra liniilor în timp ce **HAVING** acționează la nivel de grup.



**Figura II.4.2.** Ordinea de executare a clauzelor comenzii **SELECT**

Să vedem de exemplu cum se evaluează comanda următoare

```
SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
WHERE numar_voturi>15000
GROUP BY candidat
HAVING 100*sum(numar_voturi)/sum(numar_alegatori)>5
```

**Tabelul II.4.19.**

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.0222817982769582150245277809640393096
4	5.51689625238954537938001904037522059082
12	29.3512120713181287412967242185267405132

Observați însă mai întâi că prin adăugarea clauzei **WHERE**, rezultatele obținute diferă puțin de cele din tabelul II.4.18, aceasta pentru că la calculul

procentului obținut de către candidatul 12 nu mai este inclusă următoarea linie din tabelă:

**Tabelul II.4.20.**

JUDET	CANDIDAT	NUMAR_VOTURI
IS	12	12184

Așadar comanda se evaluează astfel:

- Mai întâi sunt filtrate liniile din tabelă

```
SELECT candidat, numar_voturi, numar_alegatori
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
WHERE numar_voturi>15000
```

**Tabelul II.4.21.**

CANDIDAT	NUMAR_VOTURI	NUMAR_ALEGATORI
1	347016	1750192
1	196508	650029
1	65084	363380
4	96744	1750192
5	25656	1750192
4	35784	650029
4	19937	363380
7	25937	1750192
11	22687	1750192
12	514366	1750192
12	105993	363380

Observați că au fost afișate doar 11 linii din totalul de 39 câte are tabela.

- Liniile obținute la pasul anterior sunt grupate pe candidați și se aplică funcțiile de grup.

```
SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
WHERE numar_voturi>15000
GROUP BY candidat
```

Tabelul II.4.22.

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.0222817982769582150245277809640393096
4	5.51689625238954537938001904037522059082
5	1.46589631309022095861482625906186292704
7	1.48195169444266686169288855165604687943
11	1.29625778200334591861921434905427518809
12	29.3512120713181287412967242185267405132

- În final sunt afișate doar acele linii obținute la pasul anterior care îndeplinesc condiția din clauza **HAVING**.

```

SELECT candidat,
       100*sum(numar_voturi)/sum(numar_alegatori)
FROM voturi v JOIN judete j
ON (v.judet=j.cod_judet)
WHERE numar_voturi>15000
GROUP BY candidat
HAVING 100*sum(numar_voturi)/sum(numar_alegatori)>5

```

Tabelul II.4.23.

CANDIDAT	100*SUM(NUMAR_VOTURI)/SUM(NUMAR_ALEGATORI)
1	22.0222817982769582150245277809640393096
4	5.51689625238954537938001904037522059082
12	29.3512120713181287412967242185267405132



## Aplicații

1. Se consideră o bază de date în care se ține evidența accidentelor care au loc în România. Baza de date este compusă din următoarele tabele:

```
Persoane (cod char(9), nume varchar2(35),  
          localitate varchar2(50))  
Mașini (cod char(7), model varchar2(20), an char(4),  
        codProprietar char(9))  
Accidente (data Date, codSofer char(9),  
           daune number(6,2), locAccident varchar2(50),  
           codMasina char(7))
```

**Observații:** O persoană poate fi proprietara mai multor mașini; o persoană poate conduce o mașină chiar dacă nu este proprietarul acesteia.

Scrieți câte o comandă pentru fiecare dintre următoarele cerințe:

a) Afișați numărul proprietarilor a căror mașini au fost implicate în accidente în perioada 1998-2002.

b) Afișați numele și codul persoanelor care au fost implicate în mai mult de două accidente în ultimele 4 luni.

c) Afișați o statistică a accidentelor care au avut loc în anul 2001 afișând pentru fiecare lună (numele lunii se va scrie în limba Română) numărul de accidente în care daunele au fost mai mici sau egale cu 500, și numărul accidentelor în care daunele au fost peste 500.

d) Determinați suma totală a daunelor produse în accidentele în care șoferii au fost din București.

e) Afișați anul mașinii care a fost implicată în cel mai vechi accident din baza de date.

f) Afișați daunele totale și numărul accidentelor care au avut loc în fiecare oraș.

g) Afișați orașele înregistrate în baza de date cu mai mult de 1000 de accidente produse pe raza lor.

h) Afișați numele persoanelor din baza de date care locuiesc în orașe în care au avut loc mai mult de 1000 de accidente.

i) Afișați numele proprietarilor de mașini accidentate și care locuiesc într-un oraș cu peste 1000 de accidente.

j) Afișați numele persoanelor care au fost implicate mai multe accidente în orașul lor de reședință.

k) Câte accidente au avut loc în data de 13 aprilie 1999?



- l) Care este numărul de accidente care au avut loc în ultimele 10 zile în Sibiu?
- m) Afișați data în care a avut loc cel mai recent accident în Iași și în care a fost implicat un Renault.
- n) Afișați pentru fiecare oraș persoanele implicate în accidente care au avut loc între orele 22:00-24:00, indiferent în ce zi.

2. Se consideră tabelele având următoarea structură:

```
Clase (codcls, nume, sala, etaj, profil, dirig)
Profesori (cod, nume, prenume, specializarea)
Incadrari (codprof, codcls, nr_ore)
Elevi (id, codcls, nume, prenume, adresa, telefon)
```

- a) Câte ore au în total elevii clasei "IX B"?
- b) Câți elevi învață în clasa "IX B"?
- c) La câte clase predă "Marinescu Ioan"?
- d) Câți profesori predau la clasa în care învață "Enescu Maria"?
- e) Câți profesori de matematică predau în școală?
- f) Câte clase cu profilul "Matematică-Informatică" sunt în școală?
- g) Câți profesori predau atât la "IX A" cât și la "IX B"?
- h) Câți profesori predau la toate clasele din școală?
- i) Câți elevi sunt în toate clasele de "Matematică-Informatică".
- j) Afișați lista tuturor elevilor de la profilul "Matematică-Informatică", grupați pe clase, și ordonați alfabetic în cadrul clasei.
- k) Aflați care este numărul maxim de ore predate de către un profesor.

3. Se consideră tabelele având următoarele coloane<sup>2</sup>:

```
course: courseno (pk), cname, cdate
department: depno (pk), dname, location, head
employee: empno (pk), surname, forenames,
          dob, address, depno (fk)
jobhistory: empno (fk), position, startdate, enddate, salary
empcourse: empno (fk), courseno (fk)
```

<sup>2</sup> Întrebările de la această problemă sunt preluate de la adresa <http://db.grussell.org/sql/index.cgi> cu acordul domnului dr. Gordon Russell

- a) Afișați câți angajați au urmat cel puțin un curs.
- b) Afișați numărul de angajați care au urmat fiecare curs.
- c) Afișați numărul de angajați din fiecare dintre departamentele cu codul cuprins între 3 și 5 inclusiv.
- d) Afișați salariul mediu al angajaților grupați pe departamente.
- e) Afișați numele (**surname**) și numărul de joburi pe care le-a avut fiecare angajat în cadrul firmei.
- f) Afișați salariul minim și salariul maxim din fiecare departament împreună cu numele departamentului.



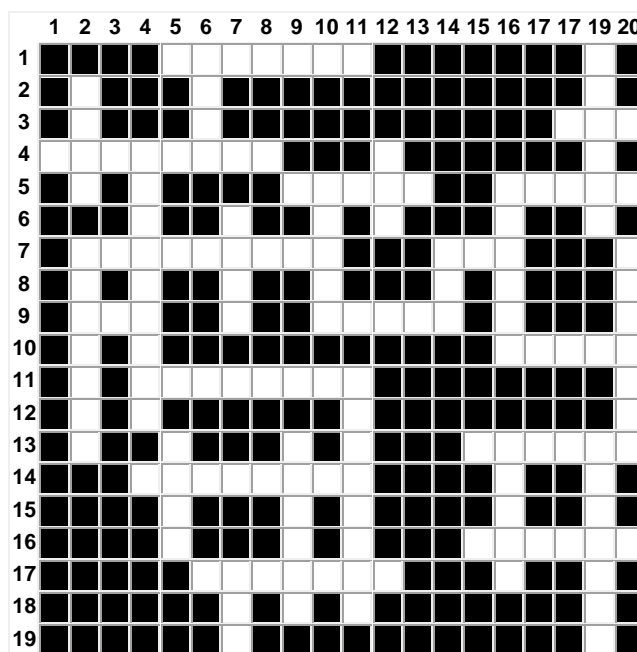
## Jocuri

Vă propunem să vă testați cunoștințele pe care vi le-ați însușit în primele patru capitole ale acestei părți prin rezolvarea a două jocuri.

1. Tabelul de mai jos conține ascunse mai multe cuvinte cheie ale limbajului SQL (nume de funcții, clauze, etc.). Pentru ca totul să fie mai distractiv, vocalele din tabel au fost înlocuite cu puncte. Cuvintele pot începe din orice poziție și se pot găsi pe orice direcție și sens (orizontal, vertical sau diagonal).

.	S	.	B	S	T	R	.	M	L	R	R
R	.	R	T	.	P	P	.	R	.	.	.
D	S	.	L	F	J	.	.	N	.	P	.
.	D	Q	C	.	C	W	S	F	D	L	N
R	.	.	.	.	.	.	S	.	.	.	D
H	T	.	.	V	.	R	.	.	N	C	.
.	.	J	N	.	L	L	.	F	G	.	C
V	.	.	T	S	.	L	.	C	T	.	.
.	T	.	M	.	S	T	.	M	P	N	D
N	.	N	.	T	C	.	P	L	M	S	.
G	R	.	.	T	.	S	T	T	.	T	.
.	V	G	S	.	M	.	W	H	.	R	.

2. Pentru al doilea exercițiu vă propunem un rebus:



#### Orizontal

1. Clauză utilizată pentru împărțirea liniilor în grupe.
3. Calculează media valorilor dintr-o coloană specificată, ignorând valorile nule.
4. Funcție care determină data ultimei zile din luna din care face parte o dată specificată.
5. Când două tabele folosite într-o interogare au coloane cu același nume folosim un \_\_\_\_ pentru tabelă pentru a evita orice confuzie. • Funcție care returnează numărul de linii în care valoarea este nenulă... dar poate fi "forțată" să includă și valorile nule utilizând \*.
7. Categorie de funcții care returnează o singură valoare pentru fiecare linie dintr-o tabelă. • Funcție care determină cea mai mică valoare dintr-o coloană specificată.

#### Vertical

2. Completează un șir de caractere la stânga cu un caracter specificat. • Funcție care returnează data și ora curentă.
4. Convertește un șir de caractere ce conține numai cifre într-un număr.
5. Funcție care permite eliminarea unor caractere de la începutul sau de la sfârșitul unui șir de caractere.
6. Completează un șir de caractere la dreapta cu un caracter specificat.
7. Operație de join în care este implicată o singură tabelă. • Funcție care înlocuiește o valoare NULL cu o valoare nenulă.
9. Funcție care returnează numărul de caractere dintr-un șir.
10. Funcție care convertește caracterele dintr-un șir la litere mici.

- |   |  |
|---|--|
| <p><b>9.</b> Adună toate valorile dintr-o coloană. • Funcție care rotunjește valoarea unei coloane sau expresii la un număr de zecimale specificat.</p> <p><b>10.</b> Funcție care șterge o parte dintre zecimalele unei coloane, expresii sau valori. Poate fi de asemenea folosită cu date calendaristice.</p> <p><b>11.</b> Tip de join care utilizează operatorul '='.</p> <p><b>13.</b> Funcție care extrage un subșir dintr-un șir.</p> <p><b>14.</b> Valoare, expresie sau nume de coloană care este transmisă unei funcții.</p> <p><b>16.</b> Funcție care simulează instrucțiunile condiționale.</p> <p><b>17.</b> Funcție care face ca șirul 'BUCUREȘTI' să arate astfel 'București'.</p> | <p><b>11.</b> Funcție care determină data următoarei zile din săptămână care urmează după o dată specificată.</p> <p><b>12.</b> Funcție care returnează cea mai mare valoare dintr-o coloană specificată.</p> <p><b>14.</b> Determină restul împărțirii a două valori.</p> <p><b>16.</b> Funcție echivalentă cu operatorul   . • Funcție care nu are nici un efect asupra șirului 'BUCUREȘTI'.</p> <p><b>19.</b> Clauză folosită pentru restricționarea grupurilor. • Convertește un șir de caractere reprezentând o dată calendaristică la o dată de tip <b>DATE</b>.</p> <p><b>20.</b> Convertește un număr sau o dată calendaristică într-o valoare de tip <b>VARCHAR2</b>.</p> |
|---|--|

(rezolvările se găsesc la pagina 313)

1. Interogări simple. Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea structurii tabelor. Constrângeri
7. Introducerea și actualizarea datelor din tabele
8. Vederi (views)
9. Secvențe. Indecși. Sinonime
10. Acordarea și revocarea drepturilor. Gestiunea tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ ce este o subinterogare
- ✓ care sunt tipurile de probleme pe care le rezolvă subinterogările
- ✓ care sunt tipurile de subinterogări
- ✓ cum se scriu și cum se folosesc subinterogările simple
- ✓ cum se scriu și cum se folosesc subinterogările multiple

Sunteți patronul unei firme. În ultima perioadă unul dintre salariații firmei, pe nume Ionescu, s-a remarcat în mod deosebit prin activitatea sa. Ați decis de aceea să îi măriți salariul și pentru a decide cu cât să-l măriți doriți să aflați care sunt persoanele cu salariu mai mare decât salariul lui Ionescu și care sunt salariile câștigate de aceștia. Cum faceți acest lucru?

Mai întâi veți determina salariul angajatului Ionescu:

```
SELECT salariul
FROM angajați
WHERE nume='Ionescu'
```

Să notăm cu *s* salariul returnat de această comandă. Acum putem afișa foarte simplu angajații cu salariu mai mare decât *s*:

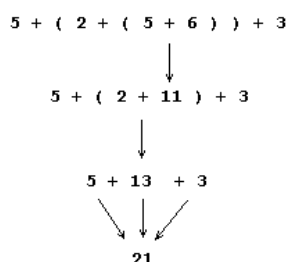
```
SELECT nume, prenume
FROM angajați
WHERE salariul>s
```

Întrebarea care se pune acum este dacă nu există posibilitatea de a uni aceste două comenzi în una singură. Răspunsul este afirmativ. Vom înlocui în a doua comandă valoarea *s* cu comanda care a generat această valoare astfel:

```
SELECT nume, prenume
FROM angajați
WHERE salariul > ( SELECT salariul
                    FROM angajați
                    WHERE nume='Ionescu' )
```

Așadar am inclus prima interogare în interiorul celei de a doua interogări. O astfel de interogare aflată în interiorul unei alte comenzi SQL se numește **subinterogare**. Subinterogările sunt întotdeauna rulate înaintea comenzii în care sunt incluse, doar pe baza rezultatelor returnate de subinterogări putându-se obține rezultatele interogării exterioare subinterogării.

Un proces similar cu modul de rulare al subinterogărilor este modul în care calculăm expresiile cu paranteze (figura II.5.1).



**Figura II.5.1.** Exemplu de proces similar

Subinterogările sunt în general folosite atunci când dorim să afișăm informații dintr-o tabelă pe baza unor informații pe care le preluăm din aceeași tabelă sau din alte tabele. De exemplu putem afișa angajații care lucrează în același departament cu angajatul **x** și sunt mai tineri decât persoana **x**.

Există două tipuri de subinterogări:

- subinterogări simple care returnează o singură linie;
- subinterogări multiple care returnează mai multe linii și/sau mai multe coloane.

Înainte de a prezenta fiecare dintre aceste tipuri de subinterogări trebuie să subliniem câteva restricții de utilizare a subinterogărilor:

- o subinterogare va fi întotdeauna inclusă în paranteză;
- subinterogarea nu poate conține clauza **ORDER BY**.

## II.5.1. Subinterogări simple

Subinterogările simple, așa cum am precizat, vor returna întotdeauna o singură valoare.

Ele pot să apară în clauza **WHERE** sau în clauza **HAVING** și sunt folosite împreună cu operatorii **<**, **>**, **<=**, **>=**, **<>**, **=**.

Vom prezenta câteva exemple folosind următoarele tabele:

```
Persoane (Id, IdFirma, Nume, Localitate, DataN)
Firme (Id, Nume, Localitate)
```

Dorim să afișăm toate persoanele care lucrează la aceeași firmă la care lucrează și Ionescu:

```
SELECT Nume FROM persoane
WHERE IdFirma = ( SELECT IdFirma
                  FROM angajati
                  WHERE nume = 'Ionescu' )
```

Același rezultat l-am putea obține cu ajutorul unui selfjoin astfel:

```
SELECT p.nume
FROM persoane p, persoane i
WHERE p.IdFirma = i.IdFirma AND
      i.nume = 'Ionescu'
```

Însă folosirea subinterogărilor este mult mai ușoară și mai naturală și în general este mai rapidă.

Iată un exemplu de folosire a operatorului <> împreună cu o subinterogare:

```
SELECT nume
FROM persoane
WHERE localitatea <> (SELECT localitatea FROM persoane
                      WHERE nume='Ionescu')
```

Comanda afișează toate persoanele care nu locuiesc în aceeași localitate cu Ionescu.

Subinterogările pot folosi funcții de grup ca în exemplul următor:

```
SELECT nume FROM persoane
WHERE DataN = (SELECT max(DataN) FROM persoane)
```

Această comandă va afișa cea mai tânără persoană din tabela **persoane**, data sa de naștere este cea mai mare, adică este cea mai recentă dată de naștere.

Similar, putem utiliza subinterogările simple în clauza **HAVING**. Să vedem cum putem afișa codul firmei cu cei mai mulți angajați:

```
SELECT IdFirma FROM persoane
GROUP BY IdFirma
HAVING count(*) = ( SELECT max(count(*))
                   FROM persoane
                   GROUP BY IdFirma )
```

Subinterogarea determină mai întâi numărul maxim de persoane angajate la o firmă, iar apoi afișează Id-ul firmei care are numărul de angajați egal cu acest maxim.

**Atenție!** Am fi tentați să scriem o comandă de forma:

```
SELECT DISTINCT IdFirma
FROM persoane
WHERE count(*) = ( SELECT max(count(*))
                  FROM persoane
                  GROUP BY IdFirma )
```

dar am precizat în capitolul anterior că funcțiile de grup **NU** pot să apară în clauza **WHERE**.

Subinterogările pot fi imbricate una în alta pe oricâte nivele. Numărul maxim de nivele de imbricare a interogărilor este teoretic nelimitat. Singura limitare care poate interveni este dată de dimensiunea buffer-elor.



În exemplul următor, am construit o interogare care afișează numele firmei care are numărul maxim de angajați. Această interogare folosește interogarea din exemplul anterior pentru a determina Id-ul firmei cu număr maxim de angajați, iar apoi caută în tabela firme numele acestei firme.

```
SELECT nume
FROM firme
WHERE Id = (SELECT IdFirma
            FROM persoane
            GROUP BY IdFirma
            HAVING count(*) = ( SELECT max(count(*))
                                FROM personae
                                GROUP BY IdFirma
                                )
            )
```

Interesant este faptul că în cadrul unei subinterogări se poate face referire la tabelele din clauza **WHERE** a interogării părinte. Astfel dacă dorim să afișăm toate persoanele care lucrează în aceeași localitate în care și locuiesc vom scrie astfel:

```
select nume
from persoane p
where localitate = ( select localitate
                    from firme f
                    where p.idfirma=f.id
                    )
```

Am folosit subinterogarea pentru a afla localitatea în care se găsește firma la care lucrează fiecare angajat în parte. Acest tip de subinterogări se numesc **subinterogări corelate**.

## II.5.2. Subinterogări multiple

Am văzut cum putem utiliza subinterogările simple. Vom studia acum modul de utilizare al subinterogărilor care returnează mai multe linii. Când o subinterogare returnează mai mult de o linie, nu mai este posibil să folosim operatorii de comparație <, >, <=, >=, <>, =, deoarece o valoare simplă nu poate fi comparată direct cu un set de valori. Va trebui să comparăm o valoare simplă cu fiecare valoare din setul de valori returnate de subinterogare. Pentru a realiza acest lucru vom folosi cuvintele cheie **ANY** și **ALL** împreună cu operatorii de comparație, pentru

a determina dacă o valoare este egală, mai mică sau mai mare decât orice valoare sau decât una din valorile din setul de date returnat de subinterogare.

Pentru a exemplifica modul de folosire a subinterogărilor multiple vom utiliza tabela **jucatori** cu următorul conținut:

**Tabelul II.5.1. Tabela Jucatori**

ID	NUME	RATING	VARSTA	LOCALITATE
18	Ion	3	30	Sibiu
11	Iulian	6	18	Brasov
22	George	3	29	Bucuresti
38	Paul	2	20	Bucuresti
13	Andrei	4	19	Sibiu
24	Marian	3	26	Cluj-Napoca
48	Ilie	-	35	Sibiu
21	Alin	2	36	Brasov
17	Radu	1	22	Cluj-Napoca
63	Vasile	7	41	Iasi

## Subinterogări multiple cu operatorul IN

Cum aflăm oare numele și localitatea jucătorilor al căror rating este egal cu al unui jucător sub 21 de ani? Vom afla mai întâi care sunt rating-urile jucătorilor sub 21 de ani:

```
SELECT rating
FROM jucatori
WHERE varsta<21
```

Vom obține trei valori ale rating-ului și anume 2, 4 și respectiv 6:

**Tabelul II.5.2.**

RATING
6
2
4

apoi vom afișa persoanele a căror rating este 2, 3 sau 6:

```
SELECT * FROM jucatori
WHERE rating IN ( 6, 2, 4 )
```

Rezultatul va fi cel din tabelul II.5.3.

Tabelul II.5.3.

ID	NUME	RATING	VARSTA	LOCALITATE
11	Iulian	6	18	Brasov
21	Alin	2	36	Brasov
38	Paul	2	20	Bucuresti
13	Andrei	4	19	Sibiu

Aceste două comenzi se pot scrie împreună în una singură prin folosirea unei subinterogări multiple astfel:

```
SELECT * FROM jucatori
WHERE rating IN ( SELECT rating FROM jucatori
                  WHERE varsta<21 )
```

Ce se întâmplă dacă o subinterogare multiplă returnează o valoare nulă iar operatorul folosit este **IN**? De exemplu, ce va afișa comanda ?

```
SELECT * FROM jucatori
WHERE rating IN ( SELECT rating FROM jucatori
                  WHERE localitate='Sibiu' )
```

Mai întâi subinterogarea va afișa rating-urile tuturor persoanelor din Sibiu:

Tabelul II.5.4.

RATING
3
4
-

deci interogarea anterioară este echivalentă cu

```
SELECT * FROM jucatori WHERE rating IN ( 3, 4, NULL)
```

sau

```
SELECT * FROM jucatori
WHERE rating=3 OR rating=4 OR rating=NULL
```

Însă din comparația cu **NULL** nu rezultă nimic (**NULL** nu poate fi comparat decât cu operatorii **IS NULL** sau **IS NOT NULL** în rest nu vom obține nici un rezultat), așadar se vor afișa doar jucătorii cu rating-ul egal cu 3 sau 4:

Tabelul II.5.5.

ID	NUME	RATING	VARSTA	LOCALITATE
24	Marian	3	26	Cluj-Napoca
22	George	3	29	Bucuresti
18	Ion	3	30	Sibiu
13	Andrei	4	19	Sibiu

Dacă însă subinterogarea va returna doar o singură valoare nulă, cum ar fi comanda:

```
SELECT rating FROM jucatori  
WHERE nume='Ilie'
```

Tabelul II.5.6.

RATING
-

atunci interogarea exterioară, neavând cu ce altă valoare să compare, nu va returna nici o linie:

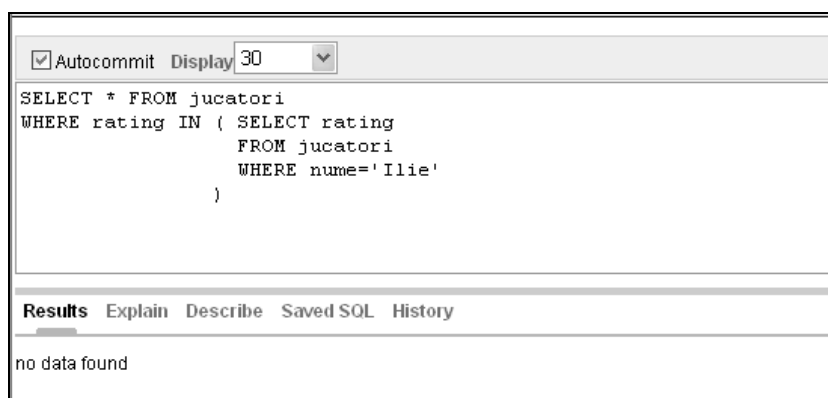


Figura II.5.2.

## Subinterogări multiple cu ALL

Fie următoarea comandă:

```
SELECT * FROM jucatori  
WHERE rating > ALL ( SELECT rating FROM jucatori  
WHERE varsta<21 )
```

Interogarea interioară returnează mulțimea valorilor rating-urilor tuturor persoanelor cu vârsta mai mică decât 21, iar interogarea exterioară va verifica fiecare persoană din tabelă pentru a vedea dacă rating-ul său este mai mare decât **fiecare** valoare returnată de către interogarea interioară.

Interogarea interioară va returna valorile 2, 4, 6 (tabelul II.5.2), deci comanda anterioară este echivalentă cu:

```
SELECT * FROM jucatori  
WHERE rating > ALL ( 2, 4, 6 )
```

sau

```
SELECT * FROM jucatori
WHERE rating>2 AND rating>4 AND rating>6
```

În concluzie, am afișat toate persoanele al căror rating este mai mare decât rating-ul tuturor persoanelor mai mici de 21 de ani.

Deci operatorul **>ALL** se poate interpreta ca **mai mare decât valoarea maximă** din mulțimea de valori returnată de către subinterogare. Similar operatorul **<ALL** se poate interpreta ca **mai mic decât valoarea minimă** din mulțimea valorilor returnate de către subinterogare

Dacă una dintre valorile returnate de către interogarea interioară este nulă atunci interogarea exterioară nu va afișa nici o linie dacă este folosită opțiunea **ALL**. Să vedem un exemplu. Dorim să afișăm toate persoanele cu rating mai mare decât rating-urile tuturor persoanelor din Sibiu:

```
select * from jucatori
where rating >ALL ( select rating
                    from jucatori
                    where localitate='Sibiu' )
```

Interogarea interioară returnează următoarele valori: 3, 4 și **NULL** (tabelul II.5.4.) și interogarea exterioară se poate scrie echivalent:

```
select * from jucatori
where rating>3 AND rating>6 AND rating>NULL
```

Condiția din clauza **where** are valoarea true doar dacă toate cele trei condiții sunt adevărate. Însă expresia "**rating>NULL**" are valoarea **NULL**, adică nu este nici adevărată, nici falsă. Așadar, condiția din clauza **WHERE** nu este adevărată niciodată și comanda nu afișează nici o linie.

## Subinterogări multiple cu ANY

Dacă folosirea opțiunii **ALL** se putea traduce printr-o condiție compusă cu operatorul **AND**, în cazul opțiunii **ANY** se va putea traduce condiția în altă condiție care folosește operatorul **OR**.

Fie următoarea comandă:

```
select * from jucatori
where rating >ANY ( SELECT rating FROM jucatori
                   WHERE vârsta<21 )
```

Am văzut că interogarea interioară returnează valorile 2, 4 și 6 (tabelul II.5.2) Comanda exterioară va afișa toți jucătorii care au un rating mai mare decât al oricărui jucător sub 21 de ani sau altfel spus, se afișează persoanele cu rating mai mare decât al **cel puțin** unei persoane cu vârsta sub 21 de ani.

**Tabelul II.5.7.**

ID	NUME	RATING	VARSTA	LOCALITATE
63	Vasile	7	41	Iasi
11	Iulian	6	18	Brasov
13	Andrei	4	19	Sibiu
18	Ion	3	30	Sibiu
24	Marian	3	26	Cluj-Napoca
22	George	3	29	Bucuresti

Putem spune că operatorul **>ANY** poate fi interpretat ca **mai mare decât valoarea minimă** din mulțimea de valori returnată de către subinterogare. Similar operatorul **<ANY** se poate interpreta ca **mai mic decât valoarea maximă** din mulțimea valorilor returnate către subinterogare.

Dacă una din valorile returnate de către interogarea interioară este nulă, interogarea exterioară poate afișa totuși ceva. De exemplu, comanda:

```
SELECT * FROM jucatori
WHERE rating >ANY ( SELECT rating FROM jucatori
                     WHERE localitate='Sibiu' )
```

va afișa

**Tabelul II.5.8.**

ID	NUME	RATING	VARSTA	LOCALITATE
63	Vasile	7	41	Iasi
11	Iulian	6	18	Brasov
13	Andrei	4	19	Sibiu

Acest lucru se întâmplă deoarece comanda dată se poate scrie echivalent:

```
SELECT * FROM jucatori
WHERE rating >ANY ( 3, 4, NULL )
```

deoarece subinterogarea returnează valorile 3, 4 și NULL (tabelul II.5.4.), și această comandă se poate scrie și

```
SELECT * FROM jucatori
WHERE rating>3 OR rating>4 OR rating>NULL
```

Condiția din **WHERE** este adevărată dacă cel puțin una din cele trei condiții este adevărată. Cum ultima condiție, **rating>NULL**, nu va fi niciodată adevărată,

este suficient ca rating-ul jucătorului să fie mai mare decât 3 sau mai mare decât 4, pentru ca el să fie afișat.

Dacă însă subinterogarea va returna o singură valoare nenulă, și nimic altceva, atunci comanda exterioară nu va afișa nimic:

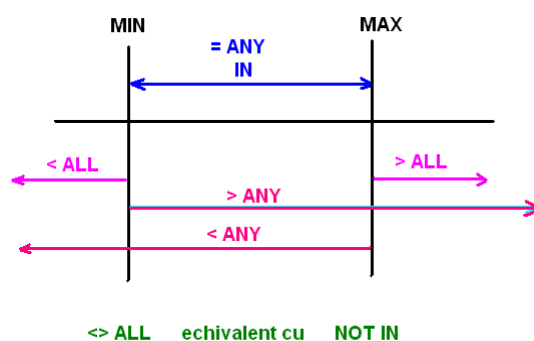
```
SELECT * FROM jucatori
WHERE rating >ANY ( SELECT rating FROM jucatori
                     WHERE nume='Ilie' )
```

**Results** Explain Describe Saved SQL History

no data found

**Figura II.5.3.**

Modul în care se pot folosi opțiunile **ANY**, **IN** și **ALL**, se pot rezuma în figura II.5.4.



**Figura II.5.4.**

Echivalențele ce se pot folosi cu aceste opțiuni sunt rezumate în tabelul următor:

**Tabelul II.5.9.**

<b>IN</b>	<b>=ANY</b>
<b>NOT IN</b>	<b>&lt;&gt; ALL</b>
<b>&lt; ANY</b>	<b>&lt; maxim</b>
<b>&gt; ANY</b>	<b>&gt; minim</b>
<b>&lt; ALL</b>	<b>&lt; minim</b>
<b>&gt; ALL</b>	<b>&gt; maxim</b>

## Subinterogări multiple cu EXISTS

Putem folosi operatorul **EXISTS** pentru a verifica dacă o subinterogare returnează vreo linie. De obicei se folosește acest operator împreună cu subinterogări corelate. De exemplu, comanda următoare afișează toți angajații care sunt managerii altor angajați:

```
SELECT employee_id, first_name, last_name
FROM employees a
WHERE EXISTS (SELECT employee_id FROM employees b
              WHERE b.manager_id=a.employee_id)
```

În subinterogare am determinat angajații coordonați de către un angajat afișat de către interogarea exterioară.

Evident, această comandă o putem transcrie cu ajutorul operatorului **IN** astfel:

```
SELECT employee_id, first_name, last_name
FROM employees a
WHERE employee_id IN
      (SELECT employee_id FROM employees b
       WHERE a.employee_id=b.employee_id)
```

Este destul de ușor de dedus că folosirea operatorului **EXISTS** oferă performanțe mai mari întrucât **IN** compară fiecare valoare returnată de către interogarea exterioară cu fiecare valoare returnată de subinterogare, pe când operatorul **EXISTS** verifică doar existența a cel puțin unei linii returnată de subinterogare, fără a face nici o comparație.

## Subinterogări multiple în clauza FROM

O subinterogare multiplă poate fi folosită și în clauza **FROM** a unei interogări ca în exemplul următor:

```
SELECT a.employee_id, first_name, last_name, nrang
FROM employees a, (SELECT manager_id, count(*) nrang
                  FROM employees GROUP BY manager_id
                  HAVING count(*)>0) b
WHERE a.employee_id=b.manager_id
```

care afișează id-ul, numele, prenumele și numărul de subalterni ai tuturor managerilor (tabelul II.5.10).



**Tabelul II.5.10.**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	NRANG
100	Steven	King	5
101	Neena	Kochhar	2
102	Lex	De Haan	1
103	Alexander	Hunold	2
124	Kevin	Mourgos	4
149	Eleni	Zlotkey	3
201	Michael	Hartstein	1
205	Shelley	Higgins	1



## Test de autoevaluare

1. Care este numărul maxim de nivele de imbricare al interogărilor:  
a) 2      b) 4      c) 8      d) 16      e) nelimitat
2. Care este ordinea de executare a clauzelor unei interogări?  
a) where, group by, funcțiile de grup, having  
b) group by, having, where, funcțiile de grup  
c) having, group by, where, funcțiile de grup  
d) funcțiile de grup, having, grup by, where
3. Tabela **A** conține două coloane având numele **col1** și respectiv **col2**. Care dintre comenzile următoare va afișa numărul de perechi distincte de forma (**col1**, **col2**) existente în tabela **A**?  
a) `SELECT count(*) FROM A`  
b) `SELECT count(DISTINCT col1, col2) FROM A`  
c) `SELECT count(*)`  
`FROM (SELECT DISTINCT col1, col2 FROM A)`  
d) `SELECT count(DISTINCT col1), count(DISTINCT) FROM A`

4. Presupunem că tabela elevi conține următoarele date:

Nume	Prenume	Media
Ionescu	Maria	10
Vasilescu	Vasile	NULL
Anghelescu	Alin	8

ce va afișa următoarea comandă?

```
SELECT avg(media) FROM elevi
```

- a) 6                      b) 9                      c) 3                      d) null                      e) 0

5. Tabela elevi conține coloanele id, nume, prenume. Care dintre următoarele comenzi afișează prenumele care apar de mai mult de trei ori în tabela elevi?

a) `SELECT prenume, count(*) FROM elevi  
HAVING count(*) > 3`

b) `SELECT prenume, count(*) FROM elevi  
WHERE count(*) > 3`

c) `SELECT DISTINCT prenume  
FROM elevi  
WHERE id IN (SELECT id FROM elevi  
GROUP BY prenume  
HAVING count(*)>3  
)`

d) `SELECT prenume FROM elevi  
GROUP BY prenume  
HAVING count(*)>3`

6. Evaluați următoarea comandă SQL:

```
SELECT id, nume, localitate  
FROM jucatori  
WHERE id IN ( SELECT id FROM jucatori  
WHERE rating>4 OR localitate='Brasov')
```

Ce valori returnează această comandă?

a) Id-ul, numele și localitatea jucătorilor care au un rating mai mare de 4 și sunt din Brașov.

b) Id-ul, numele și localitatea jucătorilor care au un rating mai mare de 4 sau sunt din Brașov.

c) Id-ul, numele și localitatea jucătorilor care au un rating mai mare de 4 sau sunt din Brașov și au un id.

d) Niciuna din variantele anterioare.

7. Câte valori poate returna o subinterogare utilizată împreună cu operatorul <>?

a) una singură                      b) cel mult două

c) cel mult 10                      d) nelimitat

(vezi baremul de corectare și răspunsurile la pagina 314)



## Aplicații

1. Revenim la baza de date privind accidentele:

```
Persoane (cod char(9), nume varchar2(35),  
          localitate varchar2(50))
```

```
Mașini (cod char(7), model varchar2(20), an char(4),  
        codProprietar char(9))
```

```
Accidente (data Date, codSofer char(9),  
          daune number(6,2), locAccident varchar2(50),  
          codMasina char(7))
```

**Observații:** O persoană poate fi proprietara mai multor mașini; o persoană poate conduce o mașină chiar dacă nu este proprietara acesteia.

a) Afișați numele proprietarului mașinii cu cel mai mare număr de accidente.

b) Afișați numele șoferului care a produs cel mai vechi accident înregistrat în baza de date.

c) Afișați persoanele care au avut accident cu una dintre mașinile al căror proprietar este 'Ionescu'.

d) Afișați codurile mașinilor care au fost implicate în accidente cu daune mai mari decât media daunelor tuturor accidentelor înregistrate în baza de date.

e) Afișați codurile mașinilor care au fost implicate în accidente cu daune mai mari decât daunele implicate în orice accident care a avut loc în București în anul 2006.

2. Următoarele întrebări se vor referi la tabelele despre mesajele postate pe un forum:

**Users**

- #UserId (number)
- UserName (varchar2)
- Cost (numeric)

**Groups**

- #GroupId (number)
- Title (varchar2)
- Category (varchar2)
- NumberOfPosts (number)
- GroupSize (number)
- Owner (number)

**Posts**

- #PostId (number)
- UserId (number)
- GroupId (number)
- ThreadId (number)
- PostText (varchar2)
- DateCreated (date)

a) Afișați numele deținătorilor de grupuri din categoria "sport" în care s-au postat mai mult de 10 mesaje.

b) Afișați titlurile tuturor grupurilor în cadrul cărora s-au postat mesaje înainte de 01.01.2006.

c) Afișați numele proprietarului grupului în cadrul căruia s-au postat cele mai puține mesaje.

d) Afișați titlurile tuturor grupurilor în cadrul cărora au postat mesaje, înainte de 01.01.2006, Ionescu și Vasilescu.

e) Afișați numele utilizatorilor care au postat exact două mesaje în grupul pe care îl dețin.

3. O firmă de publicitate în domeniul cinematografiei, dorește să facă un studiu privind filmele difuzate în timpul unui festival de film. Pentru aceasta se vor memora într-o bază de date informații despre filmele ce au rulat în timpul festivalului, producătorii acestor filme, filmele vizionate de fiecare persoană din public, preferințele fiecărui spectator. Astfel se folosesc următoarele tabele:

```
Filme (codfilm, titlu, idprod)
Producători (idprod, nume)
Persoane(id, nume, prenume, email)
Vizualizari (idpers,codfilm)
Preferințe (idpers,idfilm)
```

**Observații:** Un spectator poate prefera un film pe care nu l-a vizionat în cadrul festivalului.

Răspundeți la următoarele întrebări folosind subinterogări (evitați folosirea join-urilor).

a) Afișați numele și prenumele spectatorilor care au vizionat toate filmele prezentate în festival.

b) Afișați numele filmelor produse de către "Welles".

c) Afișați numele spectatorilor care preferă toate filmele pe care le-au văzut.

d) Afișați numele tuturor persoanelor care au văzut cel puțin un film vizionat de către "Vulturescu Ion".

e) Afișați numele tuturor persoanelor care preferă cel puțin un film preferat de către "Vulturescu Ion".

f) Afișați numele filmelor având același producător ca și filmul "Pe aripile vântului".

g) Afișați numele și prenumele persoanelor care au văzut cel puțin trei filme.

h) Afișați numele producătorilor filmelor care au în titlu cuvântul "love".

i) Afișați, pentru fiecare producător, numele și numărul de filme produse de el și care au fost prezentate în festival.

j) Afișați titlurile filmelor care nu au fost preferate de către nici un spectator.

4. Se consideră tabelele având următoarele coloane<sup>1</sup>:

```
course: courseno (pk), cname, cdate
department: depno (pk), dname, location, head
employee: empno (pk), surname, forenames,
           dob, address, depno (fk)
jobhistory: empno (fk), position, startdate, enddate, salary
empcourse: empno (fk), courseno (fk)
```

a) Folosind subinterogări, afișați numele și prenumele tuturor angajaților care lucrează în același departament cu Matthew Brownlie.

b) Folosind subinterogări, afișați codul și salariul angajaților care câștigă mai mult decât angajatul cu codul 16. În fiecare linie veți afișa atât salariul angajatului respectiv cât și salariul angajatului cu codul 16.

c) Folosind subinterogări, afișați numele complet și poziția actuală ale tuturor angajaților care au urmat un curs pe care l-a urmat și Robert Roberts.

d) Folosind subinterogări, afișați numele complet al oricărui angajat care a început un job nou în aceeași zi cu Allan Robinson.

e) Folosind subinterogări, afișați numele complet și poziția actuală ale tuturor angajaților care lucrează în același departament cu Brian Murphy și sunt mai vechi decât acesta în departamentul respectiv.

f) Folosind subinterogări, afișați numele complet al angajaților care în acest moment au același salariu cu Claire MacCallan. Ordonăți lista alfabetic după nume (surname).

g) Folosind subinterogări, afișați câți angajați au în acest moment același salariu cu Claire MacCallan.

---

<sup>1</sup> Întrebările de la această problemă sunt preluate de la adresa <http://db.grussell.org/sql/index.cgi> cu acordul domnului dr. Gordon Russell

## Crearea și modificarea structurii tabelelor. Constrângeri

# 11.6

1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ cum se creează o tabelă
- ✓ cum se pot  
adăuga/șterge/modifica  
coloanele unei tabele
- ✓ ce sunt constrângerile
- ✓ cum se pot defini  
constrângerile la nivel de  
coloană
- ✓ cum se pot defini  
constrângerile la nivel de  
tabelă
- ✓ cum se pot  
dezactiva/reactiva/șterge  
constrângerile

După etapa de modelare a bazelor de date, primul pas în realizarea unei aplicații de baze de date constă în crearea obiectelor ce compun baza de date: tabele, indecsi, vederi, sinonime, etc.

Crearea tabelelor, presupune stabilirea numelor tabelelor și a coloanelor ce le compun, stabilirea tipurilor de date pe care le au coloanele tablei, dar și declararea restricțiilor (constrângerilor) care asigură integritatea și coerența informațiilor din baza de date.

## II.6.1. Crearea tabelelor

Pentru crearea unei tabele se folosește comanda **CREATE TABLE**. Cea mai simplă formă a acestei comenzi în care, pentru moment, nu se definesc valori implicite pentru coloane și nu definim nici o restricție este:

```
CREATE TABLE numetabel
(      coloana1 tip1,
      coloana2 tip2,
      ...
      coloanan tipn )
```

unde - *numetabel* este numele atribuit tabelului nou creat. Acest nume trebuie să respecte restricțiile privind definirea numelor despre care am discutat în capitolul II.1.

- *coloana1, coloana2, ..., coloanan* sunt numele coloanelor din tabela nou creată

- *tip1, tip2, ..., tipn* reprezintă tipul datelor ce vor fi reținute în coloanele tablei nou create și dimensiunea (dacă este cazul). Principalele tipuri de date existente în Oracle au fost prezentate în capitolul I.3. Pe lângă numele tipului respectiv se precizează în paranteză lungimea tipului, respectiv numărul de caractere pentru un șir de caractere, sau numărul total de cifre și numărul de cifre de după virgulă pentru valorile numerice.

De exemplu, pentru crearea tablei corespunzătoare entității **Jucător** despre care am discutat în capitolul I.3 folosim comanda:

```
CREATE TABLE jucatori (
  nr_legitimatie NUMBER(3),
  nume VARCHAR2(30), prenume VARCHAR2(30),
  data_nasterii DATE, adresa VARCHAR2(50),
  telefon CHAR(13), email VARCHAR2(30),
  cod echipa NUMBER(3) )
```

Deocamdată nu am definit cheia primară și cheia străină.

Pentru crearea tabelii **ECHIPE** folosim comanda:

```
CREATE TABLE jucatori (  
    cod NUMBER(3),  
    nume VARCHAR2(30), localitate VARCHAR2(30),  
    adresa_club VARCHAR2(50) )
```

Iată încă un exemplu:

```
CREATE TABLE elevi (  
    id NUMBER(5),  
    nume VARCHAR2(30), prenume VARCHAR2(30),  
    bursier CHAR(1), media NUMBER(4,2) )
```

În acest exemplu, pentru tipul câmpului **media** s-au precizat două valori. Prima (4) reprezintă numărul total de cifre ale numărului, iar al doilea număr reprezintă numărul de cifre zecimale (2). Dacă sunt introduse mai mult de două zecimale, se va face rotunjire la două zecimale. La partea întreagă pot exista două cifre. Dacă numărul introdus are mai mult de două cifre la partea întreagă se va semnaliza o eroare. De asemenea, am declarat un câmp **bursier**, care ne va ajuta să memorăm dacă un elev este sau nu bursier. Însă, în Oracle nu există tipul logic (sau boolean), motiv pentru care am optat pentru tipul **CHAR(1)**, pentru un elev bursier vom memora în acest câmp valoarea 'D', pentru ceilalți elevi acest câmp rămânând necompletat.

O altă metodă de creare a unei tabeli definește structura pe baza structurii unei tabeli deja existente și în același timp, copiază datele din tabela deja existentă. Datele care se copiază din tabela deja existentă (liniile dar și coloanele ce se copiază) se precizează prin clauza **AS** urmată de o subinterogare. De exemplu, comanda următoare creează tabela **bursieri** pe baza tabelii **elevi** deja existentă:

```
CREATE TABLE bursieri  
AS SELECT id, nume, prenume FROM elevi  
WHERE bursier='D'
```

Se observă că nu sunt copiate coloanele **media** și **bursier** din tabela **elevi**.

## Definirea valorilor implicite pentru coloane

Sintaxa comenzii **CREATE TABLE** prezentată anterior este una mult simplificată. În cadrul acestei comenzi putem utiliza clauza **DEFAULT** pentru a defini o valoare implicită pentru o coloană a tabelii. Această clauză precizează ce valoare va lua un atribut atunci când, la inserarea unei linii în tabelă, nu se specifică în mod explicit valoarea atributului respectiv. Clauza **DEFAULT** apare după precizarea tipului coloanei și este urmată de constanta care definește valoarea implicită:



```
CREATE TABLE angajati
( nume varchar2(30), prenume varchar2(30),
  adresa varchar2(50) DEFAULT 'Necunoscuta',
  localitate varchar2(20) DEFAULT 'Bucuresti',
  data_ang date DEFAULT SYSDATE,
  salar NUMBER(5) DEFAULT 800 )
```

După cum se vede în exemplul anterior valoarea implicită poate fi o constantă dar poate fi de asemenea o expresie sau una dintre funcțiile speciale **SYSDATE** și **USER** (care returnează numele utilizatorului curent) dar nu poate fi numele altei coloane sau al unei funcții definite de utilizator.

Pentru o coloană pentru care nu s-a definit o valoare implicită, și nu face parte din cheia primară sau dintr-o restricție **NOT NULL** sau **UNIQUE** (despre care povestim mai târziu), sistemul va considera ca valoare implicită valoarea **NULL**.

## II.6.2. Definirea constrângerilor

După cum am precizat în prima parte a manualului, orice bază de date trebuie să stabilească regulile de integritate care să garanteze că datele introduse în baza de date sunt corecte și valide.

Aceasta înseamnă că dacă există o regulă sau restricție asupra unei entități, atunci datele introduse în baza de date respectă aceste restricții.

Regulile de integritate se definesc la crearea tabelelor folosind **constrângerile**. Constrângerile pot fi clasificate în:

- constrângeri de domeniu, care definesc valorile pe care le poate lua un atribut (**NOT NULL**, **UNIQUE**, **CHECK**)
- constrângeri de integritate a tabeli, precizând cheia primară a acesteia
- constrângeri de integritate referențială, care asigură coerența între cheile primare (sau unice) și cheile străine corespunzătoare (**FOREIGN KEY**)

Pe de altă parte constrângerile se pot clasifica după nivelul la care sunt definite în:

- constrângeri la nivel de tabelă care pot acționa asupra unei combinații de coloane
- constrângeri la nivel de coloană.

Constrângerile **NOT NULL** se pot defini doar la nivel de coloană.

Constrângerile **UNIQUE**, **PRIMARY KEY**, **FOREIGN KEY** și **CHECK** pot fi definite atât la nivel de coloană cât și la nivel de tabelă. Totuși dacă aceste constrângeri implică mai multe coloane atunci trebuie să fie definite obligatoriu la nivel de tabelă.

Dacă o restricție se definește la nivel de coloană se va folosi sintaxa:

```
nume_coloana tip_data tip_constr
```

sau

```
nume_coloana tip_data CONSTRAINT nume_constr tip_constr
```

La nivel de tabelă folosim sintaxa:

```
tip_constr
```

sau

```
CONSTRAINT nume_constr tip_constr
```

Se observă că putem decide să dăm un nume explicit unei constrângeri, ceea ce ușurează referirea ulterioară la acea constrângere, sau putem să nu definim un nume explicit, caz în care sistemul va genera un nume implicit. Dacă se folosește cuvântul **CONSTRAINT**, atunci obligatoriu acesta va fi urmat de numele dat explicit constrângerii.

Vom prezenta în continuare modul de definire al fiecăreia dintre aceste constrângeri.

## Restricția **NOT NULL**

După cum am văzut în capitolele anterioare, **NULL** este o valoare specială. Necompletarea în tabelă a unei celule conduce la completarea ei cu valoarea **NULL**, semnificând faptul că celula respectivă are de fapt o valoare nedefinită.

Într-un ERD, un atribut poate fi obligatoriu, lucru pe care îl marcăm cu o steluță în fața atributului respectiv. În baza de date această condiție se traduce prin faptul că valoarea coloanei respective trebuie obligatoriu completată, adică nu poate conține valoarea **NULL**. Pentru definirea acestui tip de restricții folosim restricția **NOT NULL** pentru coloana respectivă, fie la crearea tabelului fie mai târziu, la modificarea structurii acesteia.

La crearea tabelului, restricția **NOT NULL** se precizează pentru fiecare coloană ce trebuie să o respecte, după precizarea tipului coloanei respective astfel:

```

CREATE TABLE angajati
( nume varchar2(30) NOT NULL,
  prenume varchar2(30),
  localitate varchar2(20) DEFAULT 'Iasi' NOT NULL
  ...
)

```

Se observă că restricția **NOT NULL** a putut fi folosită în combinație cu clauza **DEFAULT**.

## Restricțiile **PRIMARY KEY** și **UNIQUE**

Cheia primară este o coloană sau o combinație de coloane care identifică în mod unic liniile unei tabele. Coloanele care fac parte din cheia primară vor fi automat de tip **NOT NULL** fără a mai fi necesară precizarea explicită. Când cheia primară este compusă dintr-o singură coloană, definirea acesteia se poate face la nivel de coloană ca în exemplul următor:

```

CREATE TABLE angajati
( cnp number(13) PRIMARY KEY
  nume varchar2(30),
  ...
)

```

sau dacă dorim să atribuim un nume constrângerii putem scrie:

```

CREATE TABLE angajati
( cnp number(13) CONSTRAINT angajati_pk PRIMARY KEY
  nume varchar2(30),
  ...
)

```

Definirea cheii primare la nivel de tabelă se poate face și atunci când cheia este compusă dintr-un singur câmp, dar este obligatorie atunci când este compusă din mai multe coloane.

De exemplu, tabela **carti** are cheia primară compusă din combinația coloanelor **titlu**, **autor**, **data\_aparitie**. Comanda de creare a acestei tabele se poate scrie:

```

CREATE TABLE carti
( titlu VARCHAR2(30),
  autor VARCHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  CONSTRAINT carti_pk
    PRIMARY KEY (titlu, autor, data_ap)
)

```

sau simplu

```

CREATE TABLE carti
( titlu VARCHAR2(30),
  autor VARCHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  PRIMARY KEY (titlu, autor, data_ap)
)

```

Sintaxa generală de definire a cheii primare este deci

```
PRIMARY KEY (lista_coloane)
```

Similar, se poate defini și restricția **UNIQUE** care precizează că valoarea coloanei definită ca **UNIQUE** sau combinația valorilor coloanelor ce definesc restricția **UNIQUE** trebuie să fie unice pentru toate liniile din tabelă. Cu alte cuvinte, într-o coloană definită ca **UNIQUE** nu pot exista valori duplicate.

**Atenție!** Coloanele definite ca **UNIQUE** pot conține valori **NULL**, iar acestea pot fi oricâte, adică valoarea **NULL** este singura valoare ce poate fi duplicată într-o coloană **UNIQUE**.

Exemple:

```

CREATE TABLE elevi
( nr_matr NUMBER(5) PRIMARY KEY,
  cnp NUMBER(13) UNIQUE,
  nume VARCHAR2(30),
  prenume VARCHAR2(30)
)

```

sau

```
CREATE TABLE elevi
( nr_matr NUMBER(5) PRIMARY KEY,
  cnp NUMBER(13) CONSTRAINT cnp_uk UNIQUE,
  nume VARCHAR2(30),
  prenume VARCHAR2(30)
)
```

sau

```
CREATE TABLE carti
( ISBN varchar2(20) PRIMARY KEY,
  titlu VARCHAR2(30),
  autor VARCHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  UNIQUE (titlu, autor, data_ap)
)
```

sau

```
CREATE TABLE carti
( ISBN varchar2(20) PRIMARY KEY,
  titlu VARCHAR2(30),
  autor VARCHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  CONSTRAINT carti_uk UNIQUE (titlu, autor, data_ap)
)
```

## Restricția FOREIGN KEY

Restricțiile referențiale sunt categoria de restricții care creează cele mai mari probleme în gestiunea bazelor de date.

Pentru exemplificarea modului de definire a cheii străine vom relua un exemplu de ERD din capitolul I.3 și anume cel din figura II.6.1.

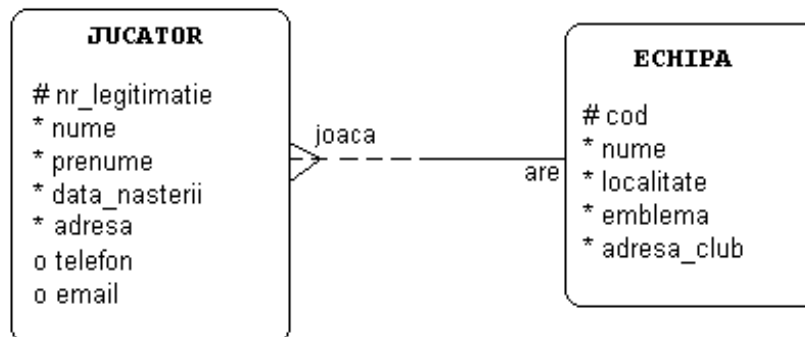


Figura II.6.1

Crearea tabelii `jucatori` corespunzătoare entității `JUCATOR` din acest ERD se va scrie:

```

CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod echipa NUMBER(3)
    REFERENCES echipe(cod)
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
)
  
```

sau

```

CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod echipa NUMBER(3) CONSTRAINT ech_fk
    REFERENCES echipe(cod),
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
)
  
```

sau la nivel de tabelă

```

CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod echipa NUMBER(3),
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30),
  FOREIGN KEY (cod echipa)
    REFERENCES echipe(cod)
)

```

sau

```

CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod echipa NUMBER(3),
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30),
  CONSTRAINT test_fk FOREIGN KEY (cod echipa)
    REFERENCES echipe(cod)
)

```

Sintaxa generală este așadar la nivel de tabelă:

```

[CONSTRAINT nume_const] FOREIGN KEY (lista_coloane)
  REFERENCES tabela_parinte(lista_coloane_referite)

```

iar la nivel de coloană

```

[CONSTRAINT nume_const]
  REFERENCES tabela_parinte(lista_coloane_referite)

```

La definirea unei chei străine se poate utiliza o clauză suplimentară **ON DELETE CASCADE** care precizează că la ștergerea unei linii din tabela părinte se vor șterge automat din tabela copil acele linii care fac referire la linia ce se șterge din tabela părinte. De exemplu, prin folosirea acestei opțiuni, la ștergerea unei echipe se vor șterge automat toți jucătorii de la acea echipă.

Această clauză se folosește astfel:

```

CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod echipa NUMBER(3) CONSTRAINT ech_fk
    REFERENCES echipe(cod) ON DELETE CASCADE,
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
)

```

O altă opțiune este `ON DELETE SET NULL` care face ca la ștergerea unui părinte, valorile cheii străine din liniile tabelului copil care fac referire la linia ștearsă vor fi setate pe `NULL`.

De exemplu la ștergerea unei echipe, jucătorii acesteia vor deveni liberi de contract, deci codul echipei la care joacă va fi setat pe `NULL`:

```

CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod echipa NUMBER(3) CONSTRAINT ech_fk
    REFERENCES echipe(cod) ON DELETE SET NULL,
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
)

```

Implicit, fără precizarea uneia din aceste două opțiuni, Oracle va interzice ștergerea unei linii din tabela părinte atâta timp cât mai există măcar o linie în tabela copil care face referire la ea.

Să vedem cum creăm acum tabela `inscrieri` corespunzătoare entității `inscriere` din figura II.6.2.

Observăm că în cheia primară intră și coloanele ce fac parte din cheia străină.



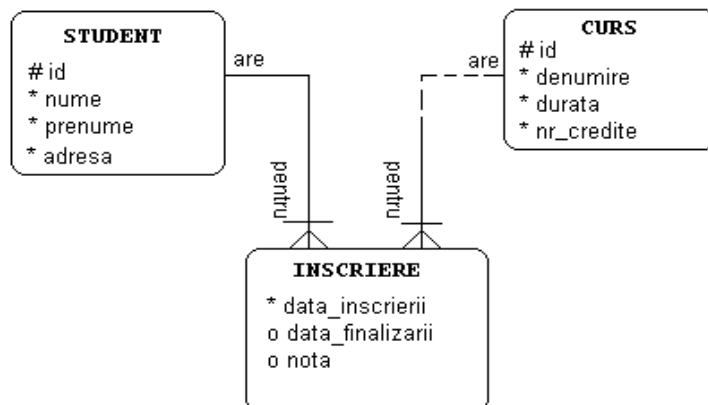


Figura II.6.2.

```

CREATE TABLE inscriere (
    id_student NUMBER(5) NOT NULL
        REFERENCES studenti(id),
    id_curs NUMBER(5) NOT NULL REFERENCES cursuri(id),
    data_inscrierii DATE DEFAULT sysdate NOT NULL,
    data_finalizarii DATE,
    nota NUMBER (4,2),
    PRIMARY KEY (id_student, id_curs, data_inscrierii)
)
  
```

## Restricția CHECK

Acest tip de constrângeri specifică o condiție ce trebuie să fie îndeplinită de datele introduse în coloana (sau coloanele) asupra căreia acționează. O astfel de constrângere poate limita valorile care pot fi introduse în cadrul unei coloane.

Iată câteva exemple de reguli de validare pentru tabela elevi care pot fi implementate cu ajutorul constrângerilor de tip **CHECK**:

- numele și prenumele unui elev trebuie să înceapă cu o majusculă, restul literelor fiind litere mici;
- nota unui elev nu poate fi mai mare de 10;
- câmpul bursier poate avea doar valorile 'D' și NULL;

- numărul de absențe nemotivate va fi cel mult egal cu numărul total de absențe.

Crearea tabelului `elevi` în această situație se poate scrie astfel:

```
CREATE TABLE elevi
( nr_matr NUMBER(5) PRIMARY KEY,
  cnp NUMBER(13) CONSTRAINT cnp_uk UNIQUE,
  nume VARCHAR2(30) NOT NULL
    CHECK nume=LTRIM(INITCAP(nume)),
  prenume VARCHAR2(30) NOT NULL
    CHECK prenume=LTRIM(INITCAP(prenume)),
  bursier CHAR(1) CHECK bursier='D',
  nota NUMBER(4,2)
    CONSTRAINT nota_ck CHECK nota<=10
  total_abs NUMBER(3),
  abs_nemotiv NUMBER(3),
  CHECK (abs_nemotiv<=total_abs)
)
```

### II.6.3. Modificarea structurii unei tabele

Modificarea structurii unui tabel se realizează cu ajutorul comenzii `ALTER TABLE`, permițând adăugarea sau ștergerea unei coloane, modificarea definiției unei coloane, crearea unei noi constrângeri sau ștergerea unor constrângeri existente.

Vom prezenta în continuare, pe scurt, fiecare dintre aceste operații.

#### Adăugarea unei noi coloane

Se realizează folosind clauza `ADD` a comenzii `ALTER TABLE`. Sintaxa este similară cu cea a creării unei coloane în cadrul comenzii `CREATE TABLE`.

De exemplu, comanda următoare adaugă o coloană `nrgoluri` la tabela `jucatori`:

```
ALTER TABLE jucatori
ADD nrgoluri NUMBER(4)
```

Coloana nou creată va deveni ultima coloană a tabelului. Dacă tabela conține deja date, coloana adăugată va fi completată cu **NULL** în toate liniile existente. De aceea nu vom putea adăuga o coloană cu restricția **NOT NULL** la o tabelă ce conține deja date.

Așadar o comandă de forma:

```
ALTER TABLE test ADD ex NUMBER(3) NOT NULL
```

sau

```
ALTER TABLE test ADD ex NUMBER(3) PRIMARY KEY
```

Sunt permise doar dacă tabela nu conține deja date.

Însă comanda

```
ALTER TABLE test ADD ex NUMBER(3) UNIQUE
```

poate fi folosită în orice moment, deoarece după cum am precizat, o coloană **UNIQUE** poate conține oricâte valori **NULL**.

## Ștergerea unei coloane

Se realizează folosind clauza **DROP COLUMN** a comenzii **ALTER TABLE**:

```
ALTER TABLE elevi DROP COLUMN bursier
```

Așa cum este și normal, ștergerea unei coloane duce automat și la ștergerea restricțiilor definite pentru aceasta și care nu implică și alte coloane.

De exemplu, dacă tabela **elevi** a fost creată cu ajutorul comenzii de la pagina 234, putem șterge fără probleme coloana **nume**:

```
ALTER TABLE elevi DROP COLUMN nume
```

chiar dacă avem definită o restricție de tip **CHECK** la nivelul acestei coloane. De asemenea, putem șterge coloana **nr\_matr**, chiar dacă aceasta este cheia primară a tabelului:

```
ALTER TABLE elevi DROP COLUMN nr_matr
```

Însă se va genera o eroare dacă încercăm să ștergem coloana **abs\_nemotiv**, din cauza restricției definite la nivel de tabelă și care implică coloanele **abs\_nemotiv** și **total\_abs**.

O variantă ar fi să ștergem mai întâi toate restricțiile în care apare coloana ce dorim să o ștergem, sau să folosim clauza **CASCADE CONSTRAINTS** astfel:

```
ALTER TABLE elevi DROP COLUMN abs_nemotiv  
CASCADE CONSTRAINTS
```

## Modificarea unei coloane

Poate fi făcută cu clauza **MODIFY**, ca în exemplul următor:

```
ALTER TABLE elevi MODIFY prenume VARCHAR2(50)
```

prin care am modificat tipul coloanei **prenume** de la **VARCHAR2(30)** la **VARCHAR2(50)**, deoarece am descoperit la un moment dat că există elevi al căror prenume (compus) are mai mult de 30 de caractere.

Mărirea numărului de caractere pentru o coloană de tip șir de caractere se poate face fără nici o problemă, însă micșorarea acestei dimensiuni se poate face doar dacă tabela este goală sau coloana respectivă conține doar valori **NULL**.

Tot cu opțiunea **MODIFY** se poate modifica, sau se poate stabili o valoare implicită, dacă nu există deja una, astfel:

```
ALTER TABLE elevi MODIFY bursier CHAR(1)
DEFAULT 'D'
```

Însă această valoare implicită nu va afecta liniile deja existente în tabelă, ci doar liniile ce vor fi introduse în continuare.

## Adăugarea unei constrângeri

Sintaxa comenzii pentru adăugarea unei constrângeri la nivel de tabelă este:

```
ALTER TABLE nume_tabela
ADD CONSTRAINT nume_constr definitie_constr
```

sau

```
ALTER TABLE nume_tabela
ADD definitie_constr
```

De exemplu, comanda următoare definește cheia primară pentru o tabelă fictivă:

```
ALTER TABLE tabelaexemplu
ADD PRIMARY KEY (coloana1)
```

Această comandă poate fi scrisă echivalent și

```
ALTER TABLE tabelaexemplu
ADD CONSTRAINT tabelaexemplu_pk PRIMARY KEY (coloana1)
```

Singura constrângere ce nu poate fi adăugată în acest fel este NOT NULL, care poate fi adăugată doar prin modificarea coloanei respective folosind MODIFY:

```
ALTER TABLE tabelaexemplu
MODIFY coloana2 VARCHAR2(20) NOT NULL
```

## Ștergerea unei constrângeri

Ștergerea unei constrângeri se face folosind opțiunea DROP CONSTRAINT astfel:

```
ALTER TABLE nume_tabela
DROP CONSTRAINT nume_constrangere
```

sau

```
ALTER TABLE nume_tabela DROP PRIMARY KEY
```

sau

```
ALTER TABLE nume_tabela
DROP UNIQUE(lista_coloane)
```

## Activarea/dezactivarea unei constrângeri

În unele situații, este necesară o dezactivare temporară și apoi reactivarea unei constrângeri. Acest lucru se realizează astfel:

```
ALTER TABLE nume_tabela DISABLE/ENABLE
CONSTRAINT nume_constrangere [CASCADE]
```

sau

```
ALTER TABLE nume_tabela DISABLE/ENABLE
PRIMARY KEY [CASCADE]
```

sau

```
ALTER TABLE nume_tabela DISABLE/ENABLE
UNIQUE (coloana1,coloana2,...) [CASCADE]
```

Clauza CASCADE precizează că și constrângerile dependente sunt de asemenea afectate.



## Test de autoevaluare

1. Care dintre liniile următoarei comenzii va genera o eroare?

```
CREATE TABLE elevi*1 (  
    id# NUMBER(5),  
    nume CHAR(50),  
    data_1 DATE DEFAULT sysdate )
```

- a) prima linie
- b) a doua linie
- c) a treia linie
- d) a patra linie

2. Care dintre următoarele afirmații este adevărată?

- a) nu puteți scădea numărul de caractere al unei coloane de tip **CHAR**
- b) nu puteți mări dimensiunea unei coloane de tip **VARCHAR2**
- c) nu puteți specifica poziția unei coloane adăugate la o tabelă existentă
- d) nu puteți converti o coloană de tip **CHAR** la tipul **VARCHAR2**.

3. Trebuie să creați o tabelă **comenzi**. Această tabelă trebuie să conțină o coloană **data** care să specifice data în care a fost făcută comanda și care va fi completată cu data curentă atunci când nu este furnizată explicit o valoare la adăugarea unei noi comenzi.

Care dintre următoarele secvențe trebuie introdusă în comanda **CREATE TABLE** pentru a defini această coloană?

- a) **data DATE SYSDATE**
- b) **data DATE DEFAULT**
- c) **data DATE = SYSDATE**
- d) **data DATE DEFAULT SYSDATE**
- e) **data DATE = (SELECT \* FROM dual)**

4. Cheia primară a tabeli **elevi** este formată din câmpul **nr\_matr** de tip numeric. Tabela conține deja datele tuturor elevilor din școală. De curând s-a modificat modul de codificare a numerelor matricole și noile numere matricole vor

conține pe lângă cifre și litere. De aceea trebuie să modificați tipul coloanei `nr_matr`. Cum realizați acest lucru?

- a) `ALTER elevi MODIFY (nr_matr VARCHAR2(15))`
- b) `ALTER elevi TABLE  
MODIFY COLUMN (nr_matr VARCHAR2(15))`
- c) `ALTER TABLE elevi  
MODIFY (nr_matr VARCHAR2(15))`
- d) `ALTER TABLE elevi  
REPLACE (nr_matr VARCHAR2(15))`
- e) nu poate fi modificat tipul coloanei `nr_matr`.

5. Care dintre următoarele șiruri **NU** poate fi folosit pentru a denumi o tabelă? (există mai multe variante de răspuns)

- a) `time1`                      b) `date`                      c) `data`
- d) `time`                      e) `time*`                      f) `$time`
- g) `datetime`

6. Care dintre liniile următoarei comenzi va genera o eroare?

```
CREATE TABLE furnizori (  
    id NUMBER,  
    nume VARCHAR2(50),  
    adresa VARCHAR2(50),  
    localitate VARCHAR2(30),  
    CONSTRAINT pk1 PRIMARY KEY (id),  
    CONSTRAINT n1 NOT NULL(nume) )
```

- a) 1                      b) 2                      c) 3                      d) 4
- e) 5                      f) 6                      g) 7

7. Doriți să adăugați o restricție `NOT NULL` coloanei `col1` de tip `NUMBER(3)` din tabela `tab1`. Pe care dintre următoarele comenzi o veți folosi?

- a) `ALTER TABLE tab1 MODIFY col1 CONSTRAINT NOT NULL`
- b) `ALTER TABLE tab1 MODIFY (col1 NOT NULL)`
- c) `ALTER TABLE tab1 MODIFY col1 NUMBER(3) NOT NULL`
- d) `ALTER TABLE tab1 ADD CONSTRAINT NOT NULL (col1)`
- e) `ALTER TABLE tab1 ADD CONSTRAINT nn NOT NULL (col1)`

8. Trebuie să adăugați o restricție de tip **FOREIGN KEY** coloanei **col1** din tabela **tab1**, astfel ca aceasta să facă referire la coloana **col2** din tabela **tab2**.

Pe care dintre următoarele comenzi o veți folosi?

- a) **ALTER TABLE tab1**  
    **ADD CONSTRAINT fk1 FOREIGN KEY (col1)**  
    **REFERENCES tab2 (col2)**
- b) **ALTER TABLE tab1**  
    **MODIFY (CONSTRAINT fk1 FOREIGN KEY (col1)**  
    **REFERENCES tab2 (col2)**
- c) **ALTER TABLE tab2**  
    **ADD CONSTRAINT fk1 FOREIGN KEY (col2)**  
    **REFERENCES tab1(col1)**
- d) **ALTER TABLE tab2**  
    **MODIFY (CONSTRAINT fk1 FOREIGN KEY (col2)**  
    **REFERENCES tab1 (col1)**

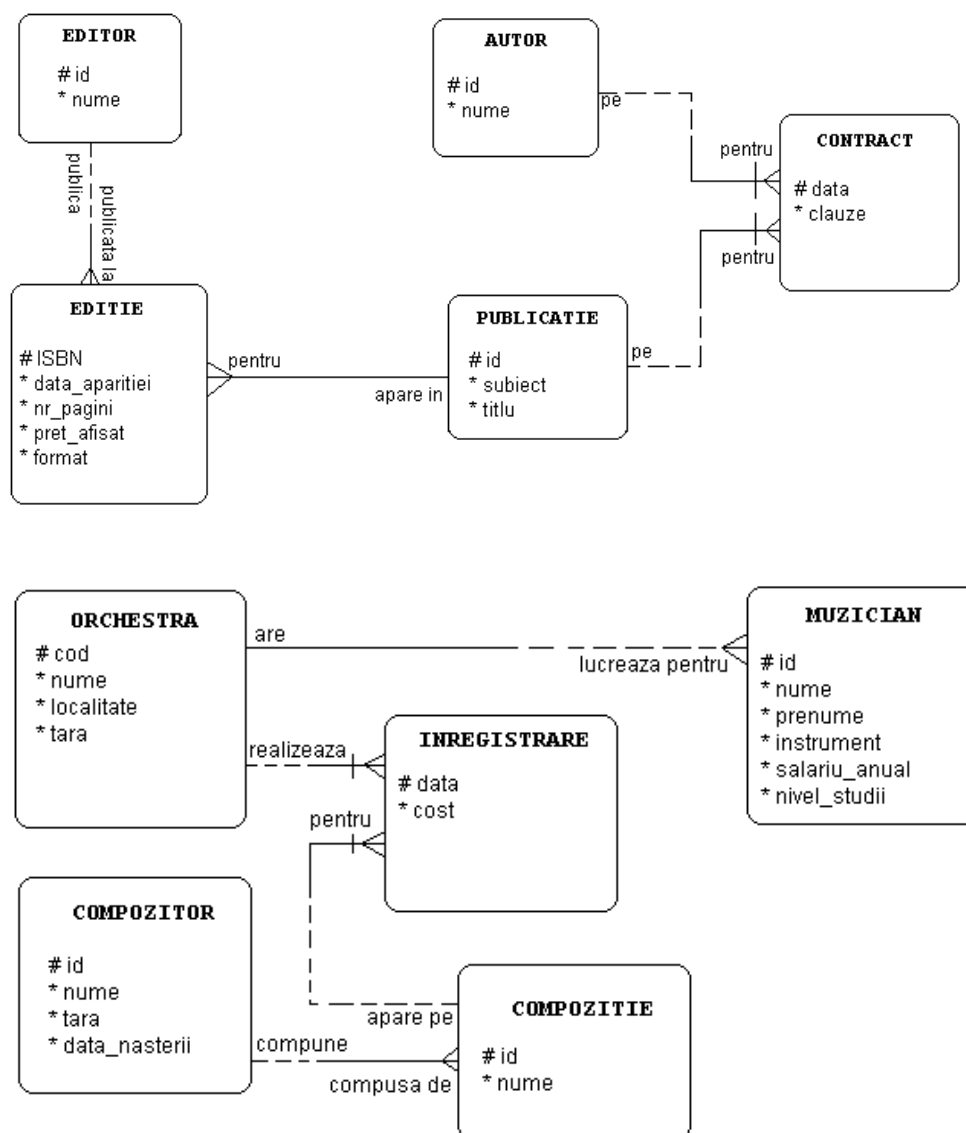
(vezi baremul de corectare și răspunsurile la pagina 314)





## Aplicații

1. Scrieți câte o comandă de creare pentru fiecare tabelă corespunzătoare unei entități din următoarele ERD-uri:



## Introducerea și actualizarea datelor din tabele

11.7

1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea structurii tabelelor.  
Constrângeri
7. Introducerea și actualizarea datelor din tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea drepturilor. Gestiunea tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ cum adăugați o nouă linie într-o tabelă
- ✓ cum ștergeți o linie dintr-o tabelă
- ✓ cum modificați valorile dintr-o tabelă

Până în acest moment am exemplificat diverse comenzi pe tabele care am presupus că există deja în baza de date și sunt deja încărcate cu date. Însă atunci când veți face propriile voastre aplicații va trebui să știți să introduceți singuri date în tabele, să modificați unele dintre aceste date, să ștergeți la un moment dat o parte dintre ele, etc.

## II.7.1. Adăugarea datelor în tabele

Pentru a adăuga linii într-o tabelă se utilizează comanda **INSERT**. Forma generală a acestei comenzi este următoarea:

```
INSERT INTO nume_tabela (lista_coloane)  
VALUES (lista_valori);
```

unde *nume\_tabela* este numele tabelii în care vom insera noua linie,

*lista\_coloane* precizează exact coloanele pe care dorim să le populăm. Această listă este opțională (ea poate lipsi).

*lista\_valori* specifică valorile pe care le vor lua, pe rând, coloanele din lista de coloane.

Lista de coloane și lista de valori trebuie să aibă același număr de elemente, și în plus coloanele și valorile din cele două liste trebuie să corespundă ca ordine și tip.

Valorile specificate în listă (sau cele implicite) într-o comandă **INSERT**, trebuie să satisfacă toate constrângerile aplicabile coloanelor respective (ca de exemplu **PRIMARY KEY**, **CHECK**, **NOT NULL**).

Dacă la rularea unei comenzi **INSERT** este generată o eroare de sintaxă, sau a fost încălcată o constrângere, linia nu este adăugată la tabelă ci se va genera un mesaj de eroare.

Atunci când din lista de coloane este omisă o coloană, Oracle va completa valoarea acelei coloane cu **NULL**, cu excepția situației când a fost definită o valoare implicită pentru coloana respectivă. În acest caz, Oracle completează coloana cu valoarea implicită. Dacă omiteți din lista de coloane o coloană care nu poate avea valoarea **NULL** (s-a definit o restricție **NOT NULL** sau **PRIMARY KEY**), și nu este definită o valoare implicită pentru acea coloană, se va genera o eroare.

Pentru a exemplifica modul de funcționare al comenzii **INSERT** vom crea tabela **jucători**:

```
create table jucatori(  
    id NUMBER(5) PRIMARY KEY,  
    nume VARCHAR2(30) NOT NULL,
```

```

    prenume VARCHAR2(30),
    rating NUMBER(1) CHECK (rating between 1 and 5),
    varsta NUMBER(2),
    localitatea VARCHAR2(30) DEFAULT 'Timisoara',
    email VARCHAR2(30) UNIQUE
)

```

O comandă completă de inserare a unei linii în această tabelă se poate scrie:

```

insert into jucatori (id, nume, prenume, rating, varsta,
                    localitatea, email)
values (18, 'Ionescu', NULL, 3, 30,
        'Sibiu', 'user18@games.ro')

```

Fără a mai specifica coloanele putem scrie următoarea comandă, în care am ținut cont de ordinea coloanelor în tabelă:

```

insert into jucatori values (11, 'Georgescu',
                            'Valeriu', 1, 18, 'Bucuresti', 'user11@games.ro')

```

Comanda următoare are ca efect completarea coloanelor `id`, `nume`, `prenume` cu valorile specificate în lista de valori iar coloanele `rating`, `varsta`, `localitatea`, `email` cu valorile implicite pentru aceste coloane, adică `'Timisoara'` pentru `localitate` și respectiv `NULL` pentru `rating`, `varsta`, `email`:

```

insert into jucatori (id, nume, prenume)
values (22, 'Vasilescu', 'Anca')

```

☒ Autocommit    Display

```
select * from jucatori
```

**Results**
Explain
Describe
Saved SQL
History

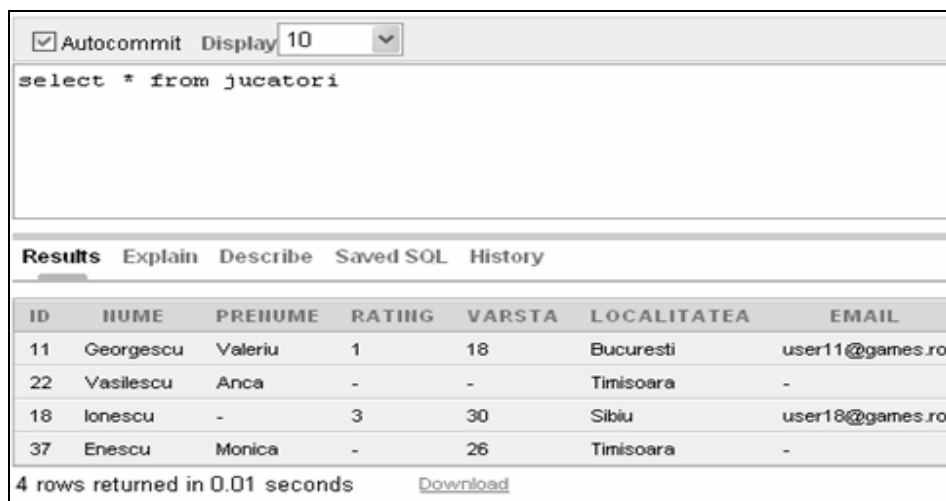
ID	HUME	PRENUME	RATING	VARSTA	LOCALITATEA	EMAIL
11	Georgescu	Valeriu	1	18	Bucuresti	user11@games.ro
22	Vasilescu	Anca	-	-	Timisoara	-
18	Ionescu	-	3	30	Sibiu	user18@games.ro

3 rows returned in 0.01 seconds
[Download](#)

Figura II.7.1

Deși câmpul `email` are definită o restricție `UNIQUE`, putem insera încă o valoare `NULL` în această coloană, doar valorile nenule trebuind a fi unice. Observați în comanda următoare cum s-a precizat că dorim setarea valorii implicite și a valorii `NULL` pentru câmpurile `localitate`, `rating` și `email`.

```
insert into jucatori (id, nume, prenume, rating, varsta,
                    localitatea, email)
values (37, 'Enescu', 'Monica', NULL, 26,
        DEFAULT, NULL)
```

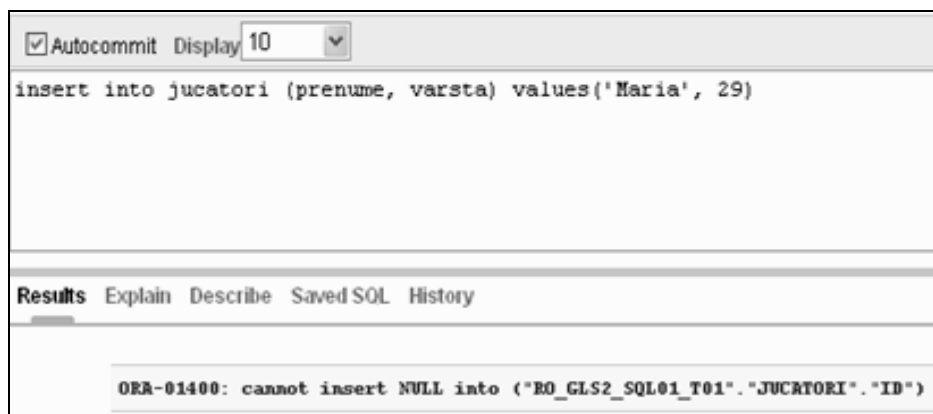


The screenshot shows a SQL query execution window. At the top, there is a checkbox for 'Autocommit' and a 'Display' dropdown set to '10'. The query entered is 'select \* from jucatori'. Below the query, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing a table with 7 columns: ID, NUME, PRENUME, RATING, VARSTA, LOCALITATEA, and EMAIL. The table contains 4 rows of data. At the bottom, it states '4 rows returned in 0.01 seconds' and has a 'Download' link.

ID	NUME	PRENUME	RATING	VARSTA	LOCALITATEA	EMAIL
11	Georgescu	Valeriu	1	18	Bucuresti	user11@games.ro
22	Vasilescu	Anca	-	-	Timisoara	-
18	Ionescu	-	3	30	Sibiu	user18@games.ro
37	Enescu	Monica	-	26	Timisoara	-

Figura II.7.2.

Nu putem însă inițializa coloanele `id` sau `nume` cu o valoare implicită, această valoare implicită fiind în acest caz valoare `NULL`, care nu este permisă pentru cheia primară sau pentru o coloană având restricția `NOT NULL`:



The screenshot shows a SQL query execution window. At the top, there is a checkbox for 'Autocommit' and a 'Display' dropdown set to '10'. The query entered is 'insert into jucatori (prenume, varsta) values('Maria', 29)'. Below the query, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing an error message: 'ORA-01400: cannot insert NULL into ("RO\_GLS2\_SQL01\_T01"."JUCATORI"."ID")'.

Figura II.7.3.

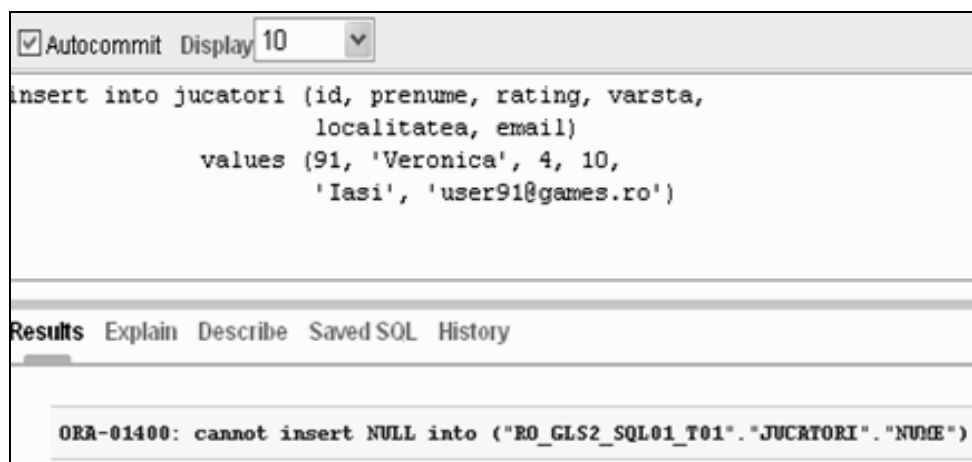


Figura II.7.4.

Pentru completarea unui câmp putem folosi o subinterogare ca în următorul exemplu:

```
insert into jucatori (id, nume, prenume)
values ((select max(id)+1 from jucatori),
        'Plesca', 'Ovidiu')
```

The screenshot shows a SQL IDE interface. At the top, there is a toolbar with a checked 'Autocommit' checkbox and a 'Display' dropdown menu set to '10'. Below the toolbar, the SQL editor contains the following code:

```
select * from jucatori
```

Below the editor, there is a tabbed interface with 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying a table of data:

ID	NUME	PRENUME	RATING	VARSTA	LOCALITATEA	EMAIL
11	Georgescu	Valeriu	1	18	Bucuresti	user11@games.ro
22	Vasilescu	Anca	-	-	Timisoara	-
43	Marin	Adriana	1	32	Brasov	user43@yahoo.com
18	Ionescu	-	3	30	Sibiu	user18@games.ro
37	Enescu	Monica	-	26	Timisoara	-
44	Plesca	Ovidiu	-	-	Timisoara	-

At the bottom of the results tab, it says '6 rows returned in 0.01 seconds' and there is a 'Download' link.

Figura II.7.5.

În Oracle este permisă adăugarea mai multor linii simultan prin preluarea datelor din alte tabele, cu ajutorul unei subinterogări. Comanda următoare preia toți angajații din tabela **employees** care au **job\_id**-ul egal cu 'IT\_PROG' și îi inserează în tabela **jucători**:

```
insert into jucatori (id, nume)
select employee_id, last_name from employees
where job_id='IT_PROG'
```

<input checked="" type="checkbox"/> Autocommit Display 30						
select * from jucatori						
Results Explain Describe Saved SQL History						
ID	HUME	PREHUME	RATING	VARSTA	LOCALITATEA	EMAIL
11	Georgescu	Valeriu	1	18	Bucuresti	user11@games.ro
22	Vasilescu	Anca	-	-	Timisoara	-
43	Marin	Adriana	1	32	Brasov	user43@yahoo.com
103	Hunold	-	-	-	Timisoara	-
104	Ernst	-	-	-	Timisoara	-
107	Lorentz	-	-	-	Timisoara	-
18	Ionescu	-	3	30	Sibiu	user18@games.ro
37	Enescu	Monica	-	26	Timisoara	-
44	Plesca	Ovidiu	-	-	Timisoara	-
9 rows returned in 0.01 seconds <a href="#">Download</a>						

Figura II.7.6.

## II.7.2. Ștergerea datelor dintr-o tabelă

Ștergerea uneia sau mai multor linii dintr-o tabelă se face utilizând comanda **DELETE** a cărei sintaxă este:

```
DELETE FROM nume_tabela WHERE conditie
```

Liniile care se vor șterge sunt selectate folosind clauza **WHERE**:

```
DELETE FROM jucatori WHERE id>100
```

Ștergerea liniilor se poate face și pe baza valorilor returnate de către o subinterogare:

```
DELETE FROM jucatori WHERE id <
      (SELECT id FROM jucatori WHERE nume='Ionescu')
```

Dacă este omisă clauza **WHERE**, se vor șterge toate liniile din tabelă, însă structura tabelii rămâne (se șterge doar conținutul tabelii, nu și tabela propriu-zisă). Deci comanda:

```
DELETE FROM jucatori
```

șterge toate liniile din tabela **jucatori**. **Atenție!** Aceste linii nu vor mai putea fi recuperate.

## II.7.3. Modificarea datelor dintr-o tabelă

Modificarea uneia sau mai multor înregistrări (linii) dintr-o tabelă se realizează cu comanda **UPDATE** care are sintaxa:

```
UPDATE nume_tabela
SET   coloana1 = valoare1,
      coloana2 = valoare2,
      ...
WHERE conditie
```

ca în următorul exemplu:

```
update jucatori
SET prenume='Emilian' WHERE id=18
```

care modifică (completează) prenumele jucătorului cu id-ul 18.

Modificarea valorilor unei linii se poate face pe baza valorilor returnate de către o subinterogare. Astfel, dacă dorim să îi atribuim jucătorului cu id-ul 44 același rating ca cel al jucătorului cu codul 18, iar vârsta să fie cu 5 mai mare decât vârsta jucătorului cu codul 43, vom scrie:

```
UPDATE jucatori
SET rating=(SELECT rating FROM jucatori WHERE id=18),
    varsta=(SELECT varsta+5 FROM jucatori WHERE id=43)
WHERE id=44
```

Dacă o subinterogare utilizată la actualizarea valorilor dintr-o coloană nu returnează nicio valoare, atunci câmpul respectiv va fi inițializat cu **NULL**:



```

UPDATE jucatori
SET rating = (SELECT rating FROM jucatori WHERE id=200)
WHERE id=44

```

Înainte de rularea acestei comenzi conținutul tabelului `jucatori` era cel din figura II.7.7, iar după rularea sa, cel din figura II.7.8. Se observă că inițial rating-ul jucătorului 44 era 3, iar după rularea comenzii, acesta a devenit `NULL`.

☒ Autocommit

Display

30

▼

```
select * from jucatori
```

Results

Explain

Describe

Saved SQL

History

ID	NUME	PRENUME	RATING	VARSTA	LOCALITATEA	EMAIL
11	Georgescu	Valeriu	1	18	Bucuresti	user11@games.ro
22	Vasilescu	Anca	-	-	Timisoara	-
43	Marin	Adriana	1	32	Brasov	user43@yahoo.com
103	Hunold	-	-	-	Timisoara	-
104	Ernst	-	-	-	Timisoara	-
107	Lorentz	-	-	-	Timisoara	-
18	Ionescu	Emilian	3	30	Sibiu	user18@games.ro
37	Enescu	Monica	-	26	Timisoara	-
44	Plesca	Ovidiu	3	37	Timisoara	-

Figura II.7.7.

☒ Autocommit    Display

```
select * from jucatori
```

Results

Explain

Describe

Saved SQL

History

ID	NUME	PRENUME	RATING	VARSTA	LOCALITATEA	EMAIL
11	Georgescu	Valeriu	1	18	Bucuresti	user11@games.ro
22	Vasilescu	Anca	-	-	Timisoara	-
43	Marin	Adriana	1	32	Brasov	user43@yahoo.com
103	Hunold	-	-	-	Timisoara	-
104	Ernst	-	-	-	Timisoara	-
107	Lorentz	-	-	-	Timisoara	-
18	Ionescu	Emilian	3	30	Sibiu	user18@games.ro
37	Enescu	Monica	-	26	Timisoara	-
44	Plesca	Ovidiu	-	37	Timisoara	-

Figura II.7.8.

Interesant este că o comandă de forma:

```
UPDATE jucatori
SET rating = (SELECT rating FROM jucatori WHERE id=18),
    varsta = (SELECT varsta+5 FROM jucatori WHERE id=18)
WHERE id=44
```

se poate scrie și astfel:

```
UPDATE jucatori
SET (rating, varsta) =
    (SELECT rating, varsta FROM jucatori WHERE id=18)
WHERE id=44
```



## Aplicații

1. Creați tabela **MyEmployees** având următoarele câmpuri

<b>ID</b>	<b>NUMBER(4) NOT NULL</b>
<b>LAST_NAME</b>	<b>VARCHAR2 (25)</b>
<b>FIRST_NAME</b>	<b>VARCHAR2 (25)</b>
<b>USERID</b>	<b>VARCHAR2 (8)</b>
<b>SALARY</b>	<b>NUMBER (9,2)</b>

2. Verificați dacă tabela a fost creată corect folosind comanda **DESC**.

3. Inserați următoarele linii în tabela **MyEmployees**

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	795
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

4. Schimbați numele angajatului 3 în **Drexler**.

5. Modificați salariul tuturor angajaților cu salariu mai mic de 900 la 1000.

6. Ștergeți angajatul "**Betty Dancs**" din tabela **MyEmployees**.

7. Goliți întreaga tabelă.



## Aplicații recapitulative

1. Creați tabela **ANGAJATI** implementând și constrângerile necesare conform precizărilor din tabelul următor:

Nume coloană	Tip	Lung	Cheia Primară	Cheia străină	Obligatoriu (NOT NULL)	Unique	Check	Default
Id	int		PK					
Prenume	char	20						unknown
Nume	char	20			Not Null	unique		
Job	char	3		referă tabela joburi (câmp jobId)				
DataAng	date				Not Null		>=1/1/1990	

2. Inserați în tabela **ANGAJATI** următoarele linii:

Id	Prenume	Nume	Job	DataAng
101	John	News	DBD	1-NOV-78
103	June	Arbough	EEG	23-JUN-99
104	Anne	Ramoras	SYA	30-FEB-92
106	William	Smithfield	PRG	31-OCT-02
114	Annelise	Jones		
118		Frommer		

3. Creați tabela **JOBURI** implementând și constrângerile necesare conform precizărilor din tabelul următor:

Nume coloană	Tip	Lung	Cheia Primară	Cheia străină	Obligatoriu (NOT NULL)	Unique	Check	Default
jobId	char	3	PK					
JobClas	char	30						
tarifOrar	decimal	5,2					>=15	15

4. Inserați în tabela **JOBURI** următoarele linii:

JobId	JobClass	tarifOrar
APD	Application Designer	48.10
DBD	Database Designer	105.00
EEG	Elec. Engineer	84.50
GSP	General Support	18.36
PRG	Programmer	37.75
SYA	Systems Analyst	96.75

5. Prenumele angajatului cu **id**-ul 118 este **Jim**. Modificați câmpul prenume din tabela **angajati** în mod corespunzător.

6. Afișați pentru fiecare angajat numele, prenumele, **job**-ul și tariful orar corespunzător.

7. Modificați comanda anterioară astfel încât să fie afișate doar persoanele a căror tarif orar este mai mic de 100.

8. Ștergeți din tabela **ANGAJATI** toate persoanele angajate înainte de 20 aprilie 2000.

9. Adăugați un nou atribut la tabela **Joburi** numită **tarifPesteNorma** și definiți o restricție **NOT NULL** la nivelul acestei coloane.

10. Adăugați o constrângere la tabela **JOBURI** care va asigura ca valoarea câmpului **tarifOrar** să fie mai mare sau egală de 25.

11. Măriți tariful orar pentru Database Designer cu 10%.

12. Ștergeți din tabela **JOBURI**, jobul având codul **PRG**.

13. Modificați tabela **JOBURI** astfel încât pentru orice înregistrare ce se inserează și pentru care nu se precizează valoarea câmpului **Job**, aceasta să fie inițializată cu valoarea **'NA'**.

14. Adăugați o constrângere care să asigure că întotdeauna câmpul **tarifOrar** va fi completat la inserarea unei noi linii.

15. Modificați coloana **tarifOrar** astfel încât aceasta să memoreze numere de lungime cel mult 12 (inclusiv punctual zecimal).

16. Ștergeți constrângerile definite la nivelul coloanei **tarifOrar**.

17. Ștergeți coloana **tarifPesteNormă** din tabela **JOBURI**.

18. Ștergeți coloana **tarifOrar** din tabela **JOBURI**.

19. Ștergeți tabela **JOBURI**.

20. Ștergeți tabela **ANGAJATI**.

1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. **Vederi (views)**
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

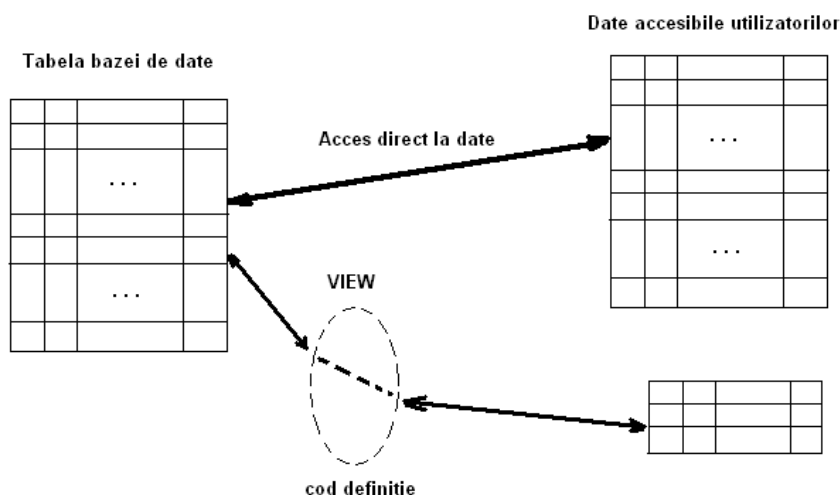
- ✓ ce este o vedere
- ✓ care sunt avantajele folosirii vederilor
- ✓ cum se creează o vedere
- ✓ cum se șterge o vedere
- ✓ cum se pot actualiza datele dintr-o tabelă prin intermediul unei vederi
- ✓ când se pot modifica datele prin intermediul unei vederi

Uneori, din motive de securitate, ați dori să nu permiteți anumitor utilizatori să aibă acces nelimitat la o tabelă, ci doar la datele ce se găsesc în anumite coloane ale acestei tabeli.

De exemplu, într-o firmă, contabilul firmei nu va avea acces la coloanele ce se referă la proiectele în care sunt implicați la momentul actual fiecare angajat al firmei, însă va avea cu siguranță acces la date privind salariul, tariful orar cu care este plătit fiecare angajat, numărul de ore lucrate etc. Pe de altă parte, bibliotecarul de la biblioteca firmei, nu va avea acces la datele privind salarizarea personalului ci doar la datele personale ale angajaților (adresa, telefon, email, etc).

Pentru a putea da acces parțial la o tabelă utilizatorilor vom folosi ceea ce numim vederi (sau views). O vedere este o tabelă virtuală, pentru care nu sunt memorate date propriu-zise ci doar definiția vederii, care are rolul de filtrare a datelor.

Vederile sunt reprezentări logice ale tabelelor existente și funcționează ca niște ferestre prin intermediul cărora pot fi vizualizate și modificate datele din tabelele fizice (fig. II.8.1).



**Figura II.8.1.** Acces direct și indirect (printr-o vedere) la o tabelă

Pe lângă faptul că oferă protecție mărită a datelor, vederile mai au un mare avantaj: ele reduc în mod considerabil complexitatea interogărilor pe care utilizatorii trebuie să le scrie. O vedere poate fi construită folosind operații complexe de join, care rămân "ascunse" utilizatorului vederii respective, care va folosi interogări simple.

La crearea unei vederi se va folosi o subinterogare, oricât de complexă, însă aceasta **NU** poate folosi clauza **ORDER BY**.

## II.8.1. Crearea și ștergerea vederilor

Sintaxa generală a comenzii pentru crearea unei vederi este:

```
CREATE OR REPLACE VIEW nume_nedere
AS subinterogare
```

Opțiunea `OR REPLACE` poate lipsi, aceasta fiind utilă atunci când dorim să modificăm o vedere deja existentă.

De exemplu, următoarea comandă creează o vedere simplă pe baza tabelului `employees`:

```
CREATE OR REPLACE VIEW v1 AS
( SELECT first_name || ' ' || last_name as Nume,
      salary
  FROM employees WHERE department_id=20)
```

După cum am precizat, o vedere se poate construi folosind mai multe tabele, ca în exemplul următor:

```
CREATE OR REPLACE VIEW v2 AS
( SELECT a.nume || ' ' || a.prenume AS Angajat,
      b.nume || ' ' || b.prenume AS Sef,
      c.nume as Firma, d.nume as Job
  FROM angajat a, angajat b
 WHERE a.id_manager = b.id(+) and
      a.idFirm=c.idFirm(+) and a.idJob=d.idJob(+)
)
```

**Observație.** În subinterogarea care definește o vedere, toate expresiile (nu și coloanele simple) trebuie să aibă asociate un alias pentru a putea fi ulterior referite în interogări.

Cum putem interoga aceste vederi? Ele pot fi folosite ca orice tabelă obișnuită, atât în interogări cât și în operațiile de actualizare (adăugare, modificare, ștergere), asupra acestora din urmă însă vom reveni în paragrafele următoare. Putem scrie de exemplu:

```
SELECT nume, salary FROM v1
WHERE nume like '%a%'
```

sau

```
SELECT angajat, sef, firma, job
FROM v2
```

O vedere poate fi ștearsă cu comanda

```
DROP VIEW nume_vedere
```

**Atenție!** Ștergerea unei vederi nu afectează în niciun fel datele din tabelele pe baza cărora s-a creat vederea. Toate modificările realizate asupra tabelelor prin intermediul vederii rămân valabile și după ștergerea acestora.

## II.8.2. Actualizarea datelor prin intermediul vederilor

În acest paragraf vom folosi pentru exemplificare tabelele `jucatori` și `echipe` create cu ajutorul următoarelor comenzi:

```
CREATE TABLE jucatori(  
    id NUMBER(5) PRIMARY KEY,  
    nume VARCHAR2(30) NOT NULL,  
    prenume VARCHAR2(30),  
    rating NUMBER(1) CHECK (rating BETWEEN 1 AND 5),  
    varsta NUMBER(2),  
    localitatea VARCHAR2(30) DEFAULT 'Timisoara',  
    email VARCHAR2(30) UNIQUE  
)
```

Să creăm acum următoarele vederi:

```
CREATE OR REPLACE VIEW v1_JucatoriTm AS  
( SELECT id, nume, varsta, localitatea FROM jucatori  
  WHERE localitatea = 'Timisoara' )
```

și

```
CREATE OR REPLACE VIEW v2_Jucatori AS  
( SELECT nume, prenume FROM jucatori  
  WHERE rating IS NOT NULL)
```

Așadar am creat o vedere pentru toți jucătorii din Timișoara. Putem interoga simplu această vedere:

```
SELECT * FROM v1_JucatoriTm
```

rezultatul fiind cel din tabelul următor:



Tabelul II.8.1.

ID	NUME	VARSTA	LOCALITATEA
22	Vasilescu	-	Timisoara
103	Hunold	-	Timisoara
104	Ernst	-	Timisoara
107	Lorentz	-	Timisoara
37	Enescu	26	Timisoara
44	Plesca	37	Timisoara

iar comanda

```
SELECT * FROM v2_Jucatori
```

va afișa

Tabelul II.8.2.

NUME	PRENUME
Georgescu	Valeriu
Marin	Adriana
Ionescu	Emilian

Vom încerca acum, pe rând, să vedem cum funcționează fiecare operație de actualizare a datelor.

O vedere poate fi creată folosind opțiunea **WITH READ OPTION**, prin intermediul unei astfel de vederi neputându-se efectua nici o operație de actualizare. Aceste vederi sunt folosite doar pentru vizualizarea datelor:

```
CREATE OR REPLACE VIEW v4_JucatoriTm AS
( SELECT id, nume, varsta, localitatea
  FROM jucatori
  WHERE localitatea = 'Timisoara' )
WITH READ ONLY
```

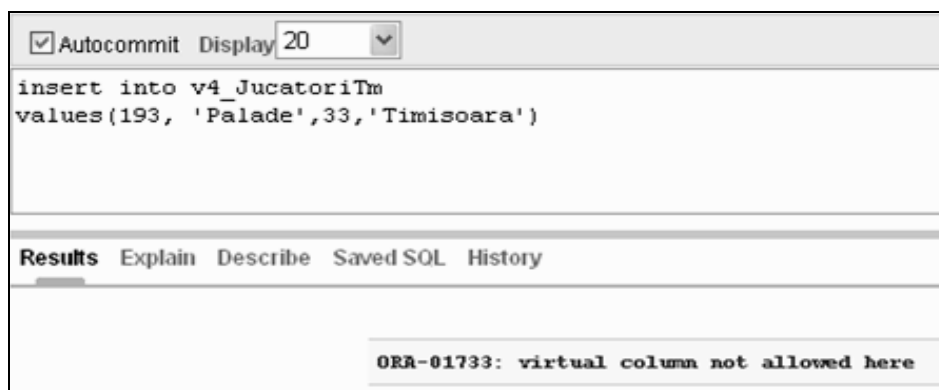


Figura II.8.2.

## Inserarea datelor prin intermediul vederilor

Încercăm să inserăm câte o înregistrare în tabela jucători prin intermediul celor două vederi create anterior:

```
insert into v1_JucatoriTm
values(210, 'Alexandrescu',41,'Iasi')
```

Comanda funcționează perfect (fig. II.8.3), deși jucătorul nou inserat nu respectă domeniul vederii `v1_JucatoriTm`, adică deși putem vizualiza prin intermediul acestei vederi doar jucătorii din Timișoara, am reușit totuși să inserăm un jucător din altă localitate. Acest lucru ar putea crea probleme de securitate (am creat vederea tocmai pentru a restricționa drepturile utilizatorilor).

<input checked="" type="checkbox"/> Autocommit Display 20						
select * from jucatori						
Results Explain Describe Saved SQL History						
ID	NUME	PRENUME	RATING	VARSTA	LOCALITATEA	EMAIL
11	Georgescu	Valeriu	1	18	Bucuresti	user11@games.ro
22	Vasilescu	Anca	-	-	Timisoara	-
43	Marin	Adriana	1	32	Brasov	user43@yahoo.com
103	Hunold	-	-	-	Timisoara	-
104	Ernst	-	-	-	Timisoara	-
107	Lorentz	-	-	-	Timisoara	-
210	Alexandrescu	-	-	41	Iasi	-
18	Ionescu	Emilian	3	30	Sibiu	user18@games.ro
37	Enescu	Monica	-	26	Timisoara	-
44	Plesca	Ovidiu	-	37	Timisoara	-
10 rows returned in 0.01 seconds <a href="#">Download</a>						

Figura II.8.3.

Această problemă poate fi rezolvată prin folosirea opțiunii `WITH CHECK OPTION` la crearea vederii. Vom crea o nouă vedere `v3_jucatoriTm` folosind această opțiune:

```
CREATE OR REPLACE VIEW v3_JucatoriTm AS
( SELECT id, nume, varsta, localitatea FROM jucatori
  WHERE localitatea = 'Timisoara' )
WITH CHECK OPTION
```

De această dată nu mai putem insera valori care sunt în afara domeniului vederii (fig. II.8.4).

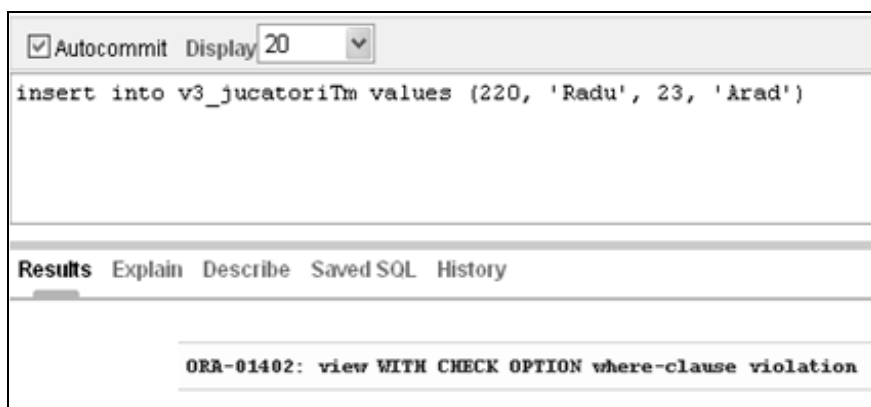


Figura II.8.4.

Prin intermediul vederii `v2_jucatori` nu vom putea insera linii în tabela `jucatori`, deoarece prin intermediul vederii nu avem acces la câmpul `id`, care fiind cheie primară nu poate fi inițializat cu valoarea implicită `NULL` (fig. II.8.5).

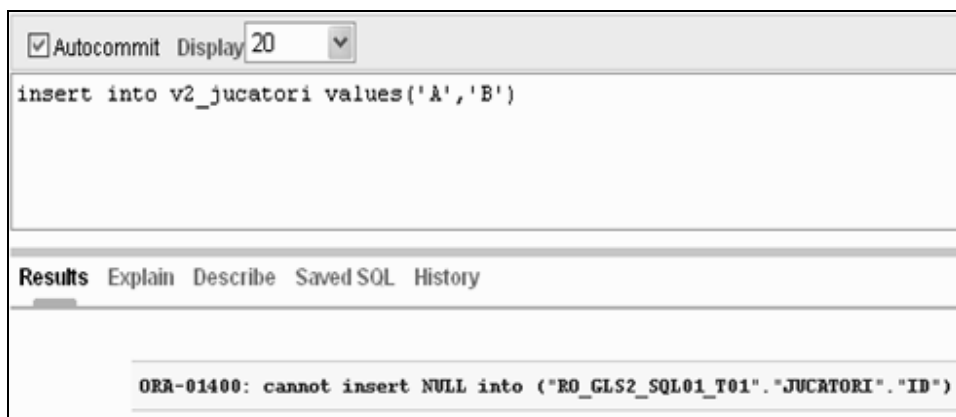


Figura II.8.5.

## Ștergerea datelor prin intermediul vederilor

La ștergerea unei înregistrări vom folosi comanda **DELETE** cu formatul deja cunoscut. Evident nu vom putea șterge din tabelă decât liniile accesibile prin vederea respectivă. De aceea comanda:

```
DELETE FROM v1_jucatoriTm WHERE id=43
```

nu va genera nici o eroare, însă nu va șterge nici o linie întrucât jucătorul având id-ul 43 este din **Brasov**, deci nu avem acces la el prin intermediul vederii **v1\_jucatoriTm**.

Similar, nu vom putea folosi în clauza **WHERE** a comenzii **DELETE** coloane care nu sunt vizibile din vederea respectivă. Comanda

```
DELETE FROM v2_jucatori WHERE id=43
```

va genera o eroare, deoarece câmpul **id** este inaccesibil vederii (fig. II.8.6.)

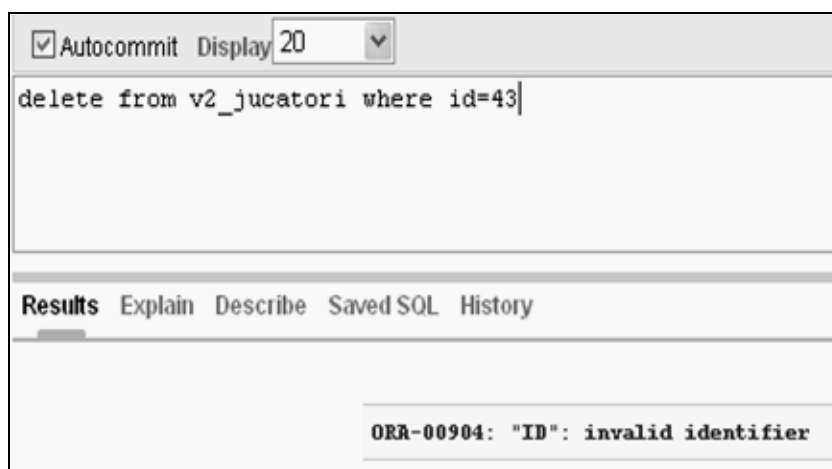


Figura II.8.6.

Comenzile

```
delete from v2_jucatori where prenume='Emilian'
```

și

```
delete from jucatori where id=107
```

sunt perfect funcționale.

## Modificarea datelor prin intermediul vederilor

Ca și în cazul celorlaltor operații de actualizare vom putea modifica doar valorile liniilor și coloanelor care sunt vizibile din vederea respectivă:

```
update v1_jucatoriTm
set varsta=13
where id=103
```

## Restricții privind utilizarea vederilor

Operațiile de actualizare a datelor prin intermediul vederilor NU pot fi realizate în următoarele condiții:

- actualizarea datelor (**ștergere, modificare, inserare**) nu se poate efectua dacă subinterogarea cu care s-a creat vederea folosește:
  - funcții de grup
  - clauza **GROUP BY**
  - clauza **DISTINCT**
  - pseudocoloanele **ROWNUM** sau **ROWID**
- nu se poate **modifica** un câmp calculat al unei vederi:

De exemplu, dacă s-a creat vederea

```
CREATE VIEW v5 AS
( SELECT id, nume, nvl(rating,0) rating
  FROM jucatori)
```

vom putea actualiza câmpurile **id** și **nume**:

```
UPDATE v5
SET nume='Eminescu'
WHERE id=37
```

dar nu putem modifica valoarea din câmpul **rating** (fig. II.8.7).

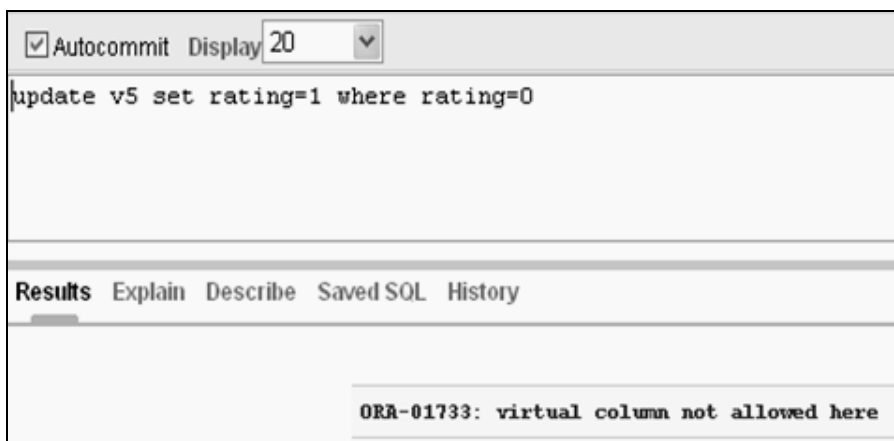


Figura II.8.7.

- Nu se poate insera o linie într-o tabelă prin intermediul unei vederi decât dacă toate coloanele **NOT NULL** ale tabelului sunt prezente în vedere.



## Aplicații<sup>1</sup>

Se consideră tabelele având următoarele coloane:

```
course: courseno (pk), cname, cdate
department: depno (pk), dname, location, head
employee: empno (pk), surname, forenames,
          dob, address, depno (fk)
jobhistory: empno (fk), position, startdate, enddate,
            salary
empcourse: empno (fk), courseno (fk)
```

1. Creați o vedere **secondview** prin intermediul căreia puteți vizualiza toți angajații cu **empno** mai mic decât 5.
2. Creați o vedere prin intermediul căreia puteți afișa numărul de cursuri pe care le-a urmat fiecare angajat.
3. Creați o vedere prin intermediul căreia puteți vizualiza numărul de joburi pe care fiecare angajat le-a avut.
4. Creați o interogare care combină vederile de la întrebările 2 și 3 pentru a afișa câte joburi și câte cursuri a urmat fiecare angajat. Nu trebuie să creați din nou vederile ci doar să le folosiți în interogare pe cele scrise la punctele anterioare. Se vor afișa toți angajații chiar și cei care nu au urmat nici un curs. Indicație: utilizați un **OUTER JOIN**.
5. Creați o vedere care are aceeași structură cu tabela **department** dar care conține doar departamentele a căror **head** (șef) este egal cu **empno**-ul angajatului cu numele (**surname**) **Jones**.
6. Utilizând vederea creată la întrebarea 5 (**NU** tabela **department**) afișați **surname** și **forenames** tuturor angajaților împreună cu numele departamentului în care aceștia lucrează. Dacă departamentul unui angajat nu este accesibil prin intermediul vederii, afișați în locul numelui departamentului un spațiu. Folosiți un **OUTER JOIN**.
7. Modificați interogarea de la punctul 6 astfel încât în locul numelui departamentului celor a căror departament nu este accesibil prin intermediul vederii, să se afișeze textul **"UNKNOWN"** (cu majuscule).

---

<sup>1</sup> Aplicațiile propuse în această secțiune sunt preluate de la adresa <http://db.grussell.org/sql/index.cgi> cu acordul domnului dr. Gordon Russell.

1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ ce sunt secvențele
- ✓ cum se creează și cum se  
șterge o secvență
- ✓ cum se modifică o secvență
- ✓ cum se folosește o secvență
- ✓ ce sunt indecșii și care sunt  
avantajele lor
- ✓ când este indicată folosirea  
indecșilor
- ✓ cum se creează și cum se  
șterge un index
- ✓ ce sunt sinonimele
- ✓ cum se creează și cum se  
șterge un sinonim
- ✓ cum se folosesc sinonimele

## II.9.1. Secvențe

Imaginați-vă că trebuie să adăugați în baza de date a școlii, datele personale ale noilor elevi veniți în școala voastră în clasa a IX-a. Fiecărui elev trebuie să-i asociați un `id` unic în întreaga bază de date. Nu știți însă exact care sunt `id`-urile elevilor deja existenți în baza de date, pentru a ști care sunt `id`-urile "libere". Cum rezolvați oare această problemă?

O variantă ar fi ca la inserarea unui nou elev să determinați cel mai mare `id` existent în baza de date și să-i asociați elevului nou inserat un `id` cu o unitate mai mare decât cel mai mare `id`. Veți scrie o comandă de forma:

```
INSERT INTO elevi (id, nume, prenume, ...)
VALUES ( SELECT max(id)+1 FROM elevi,
          'Ionescu', 'Ioan', ...)
```

O astfel de soluție poate genera probleme în cazul accesului concurrent la baza de date, când este posibil ca doi utilizatori diferiți să încerce să insereze doi elevi cu același `id`.

Soluția este folosirea secvențelor. Secvențele sunt obiecte ale bazei de date care generează automat, în mod secvențial, liste de numere. Acestea sunt utile când o tabelă folosește o cheie primară artificială, ale cărei valori dorim să le generăm automat.

### Crearea și ștergerea secvențelor

Sintaxa pentru crearea unei secvențe este următoarea:

```
CREATE SEQUENCE nume_secventa
START WITH n1
INCREMENT BY n2
MAXVALUE n3 | NOMAXVALUE
MINVALUE n4 | NOMINVALUE
CACHE n5 | NOCHACE
CYCLE | NOCYCLE
```

Să explicăm pe rând care este rolul fiecărei opțiuni din această comandă:

- **START WITH n1** – precizează de la ce valoare va începe generarea valorilor. Această opțiune este utilă atunci când câmpul pentru care dorim să generăm valori, folosind această secvență, conține deja valori. În acest caz, vom preciza în `n1` o valoare mai mare decât toate valorile deja existente în coloana respectivă. Dacă această opțiune nu este prezentă, se va începe implicit de la valoarea 1.



- **INCREMENT BY n2** – precizează intervalul dintre două numere din secvență. Poate fi un număr întreg pozitiv sau negativ, dar nu poate fi zero. Dacă se precizează o valoare negativă, atunci valorile se vor genera în ordine descrescătoare, altfel se vor genera în ordine crescătoare. Dacă omiteți această opțiune, valoarea implicită a incrementului va fi 1.
- **MAXVALUE n3** și respectiv **MINVALUE n4** – aceste clauze specifică cea mai mare, respectiv cea mai mică valoare returnată de către secvență. **n3** și respectiv **n4** ce trebuie să fie numere întregi cu maximum 9 cifre.
- **NOMAXVALUE** – valoarea maximă generată va fi 2147483647 pentru o secvență cu increment pozitiv, respectiv -1 pentru o secvență cu increment negativ.
- **NOMINVALUE** – valoarea maximă generată va fi 1 pentru o secvență cu increment pozitiv, respectiv -2147483647 pentru o secvență cu increment negativ.
- **CACHE n5** – această opțiune este folosită din considerente de eficiență. Cu această opțiune se vor genera simultan **n5** valori din secvență, și numai atunci când acestea se vor epuiza se vor genera următoarele **n5** valori. În acest fel se vor face mai puține modificări asupra bazei de date.
- **CYCLE** | **NOCYCLE** – dacă specificați opțiunea **CYCLE** atunci când secvența a ajuns la valoarea maximă (respectiv minimă pentru o secvență cu increment negativ), secvența va reîncepe să genereze valori începând cu **MINVALUE** (respectiv **MAXVALUE** pentru o secvență cu increment negativ). Evident, dacă utilizați opțiunea **CYCLE** nu există nici o garanție privind unicitatea valorilor generate.

De exemplu, comanda:

```
CREATE SEQUENCE sec1
START WITH 1 INCREMENT BY 1
```

creează o secvență care va genera valori din 1 în 1, începând cu 1, adică va genera în ordine valorile 1, 2, 3, etc.

Comanda

```
CREATE SEQUENCE sec2
START WITH 120 INCREMENT BY -3
```

creează o secvență care va genera valori descrescătoare din 3 în 3, începând cu 120, adică va genera în ordine valorile 120, 117, 114, etc.

Ștergerea unei secvențe se face simplu cu comanda **DROP SEQUENCE**.

## Utilizarea secvențelor

Să vedem cum generăm efectiv valorile din secvență. Vom folosi două pseudocoloane speciale numite **NEXTVAL** și respectiv **CURRVAL**. **NEXTVAL** generează următoarea valoare din secvență, în timp ce **CURRVAL** este folosită pentru a afla care a fost valoarea care tocmai a fost generată.

Pentru exemplificare, creăm secvența

```
CREATE SEQUENCE sec3  
START WITH 5 INCREMENT BY 3
```

și tabela

```
CREATE TABLE test(nr number(3))
```

și rulăm de 3 ori comanda:

```
INSERT INTO test values(sec3.NEXTVAL)
```

În acest fel, conținutul tabelului este 5, 8, 11 (fig. II.9.1):

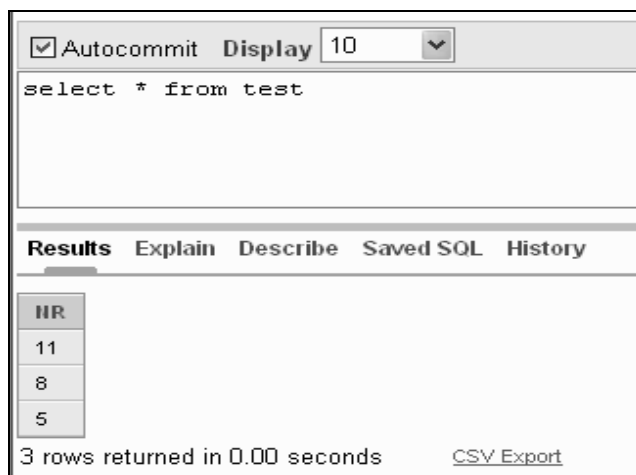


Figura II.9.1.

Dacă rulăm acum comanda

```
SELECT sec3.currval FROM dual
```

se va afișa valoarea 11, adică exact ultima valoare generată de către secvență.

**Atenție!** Pseudocoloanele **NEXTVAL** și **CURRVAL** nu pot fi folosite în următoarele contexte:

- în clauza **SELECT** a unei vederi;
- într-o comandă **SELECT** care folosește opțiunea **DISTINCT**;
- într-o comandă **SELECT** care folosește clauzele **GROUP BY**, **HAVING** sau **ORDER BY**;

- într-o subinterogare din cadrul unei comenzi **SELECT**, **DELETE** sau **UPDATE**;
- Într-o opțiune **DEFAULT** a comenzii **CREATE TABLE** sau **ALTER TABLE**.

## Modificarea secvențelor

Comanda **ALTER SEQUENCE** care permite modificarea unei secvențe are sintaxa similară cu cea a comenzii **CREATE SEQUENCE**:

```
CREATE SEQUENCE nume_secventa
  INCREMENT BY n2
  MAXVALUE n3 | NOMAXVALUE
  MINVALUE n4 | NOMINVALUE
  CACHE n5 | NOCHACE
  CYCLE | NOCYCLE
```

Modificarea unei secvențe va afecta doar valorile ce se vor genera ulterior. La modificarea unei secvențe trebuie să se țină cont de câteva restricții. De exemplu, nu se poate stabili o valoare în clauza **MAXVALUE** care să fie mai mică decât ultima valoare care a fost deja generată de către secvență.

Să experimentăm puțin opțiunea de modificare a unei secvențe. Să rulăm, pe rând, următoarele comenzi:

```
CREATE SEQUENCE sec4;
CREATE TABLE test1 (n NUMBER(2), v NUMBER(2));
INSERT INTO test1 values(1, sec4.NEXTVAL);
INSERT INTO test1 values(2, sec4.NEXTVAL);
INSERT INTO test1 values(3, sec4.NEXTVAL);
INSERT INTO test1 values(4, sec4.CURRVAL);
ALTER SEQUENCE sec4 INCREMENT BY -5 MINVALUE -200;
INSERT INTO test1 values(5, sec4.NEXTVAL);
INSERT INTO test1 values(6, sec4.NEXTVAL);
INSERT INTO test1 values(7, sec4.NEXTVAL);
```

După aceste comenzi, conținutul tabelului **test** va fi cel din tabelul următor:

**Tabelul II.9.1.**

N	V
1	1
2	2
3	3
4	3
5	-2
6	-7
7	-12

**Atenție!** În Oracle Database Express Edition este posibil ca referirea la pseudocoloana **CURRVAL** să nu funcționeze în mod corespunzător.

## II.9.2. Indecși

Să presupunem că am creat o tabelă cu comanda:

```
CREATE TABLE test (  
    id integer, content varchar )
```

și am inserat o mulțime de linii în această tabelă. La un moment dat avem nevoie să rulăm o interogare de forma:

```
SELECT content FROM test WHERE id = 5;
```

Server-ul bazei de date va trebui să parcurgă întreaga tabelă `test`, linie de linie, pentru a căuta toate liniile pentru care `id`-ul este 5. Dacă tabela conține foarte multe linii și doar puține linii (poate chiar nici una) vor fi returnate de către interogarea anterioară, această metodă este clar ineficientă.

Pentru a avea un acces direct și rapid la liniile unei tabele, se vor folosi indecși.

Indecșii unei tabele funcționează similar cu indexul unei cărți de specialitate. Într-un astfel de index, aflat de obicei la sfârșitul unei cărți se găsesc principalii termeni și concepte întâlnite în cartea respectivă, sortați alfabetic, indicându-se în dreptul fiecărui termen pagina sau paginile la care poate fi întâlnit termenul respectiv în carte. O persoană interesată de un anumit termen, nu va citi întreaga carte, ci va căuta în index pagina sau paginile corespunzătoare.

Există două tipuri de indecși:

- **indecși unici** – sunt generați automat pentru coloanele ce fac parte din cheia primară sau asupra cărora s-a definit o constrângere **UNIQUE**.
- **indecși non-unici** – care sunt definiți de către utilizator.

Crearea unui index se realizează cu comanda:

```
CREATE INDEX nume_index  
ON nume_tabela(coloana1, coloana2, ... , coloanan)
```

De exemplu, dacă dorim să creștem viteza operațiilor de căutare după coloana `nume` din tabela `elevi`, vom crea următorul index:

```
CREATE INDEX elevi_idx1  
ON carti(nume)
```

Într-un index putem include mai multe coloane ale unei tabele, ca în următorul exemplu:

```
CREATE INDEX elevi_idx2  
ON carti(nume, prenume)
```

De asemenea pot fi incluse în index expresii, nu doar coloane ale unei tabeli:

```
CREATE INDEX elevi_idx3
ON carti(UPPER(ume), UPPER(prenume))
```

Pentru a șterge un index folosiți comanda **DROP INDEX**. Indecșii pot fi adăugați și șterși în orice moment fără a afecta tabela pe care o indexează în nici un fel, ei fiind fizic și logic independenți de tabela pe care o indexează. Totuși, atunci când veți șterge o tabelă, se vor șterge automat toți indecșii definiți pe tabela respectivă.

Odată creat un index, nu mai este necesară nici o intervenție, acesta fiind actualizat automat după fiecare modificare efectuată asupra tabeli. De asemenea indexul va fi folosit automat în interogări care pot câștiga de pe urma folosirii sale.

Un index definit pe o coloană care face parte dintr-o condiție de join, poate duce la creșterea semnificativă a vitezei de executare a join-ului respectiv.

Așadar, este indicată crearea unui index atunci când:

- coloana care se indexează conține o plajă mare de valori;
- coloana care se indexează conține multe valori nule (valorile nule nu sunt incluse în index);
- una sau mai multe coloane sunt frecvent folosite împreună în clauza **WHERE** sau în condițiile de join;
- tabela este mare și majoritatea interogărilor returnează un număr mic de linii din această tabelă (~5% din numărul total de înregistrări)

Când NU este indicat să creați un index? Atunci când:

- tabela este mică, în acest caz căutarea secvențială este acceptabilă;
- coloanele nu sunt foarte des folosite în clauza **WHERE** a interogărilor;
- majoritatea interogărilor returnează un număr mare de înregistrări (mai mult de 5% din numărul total de înregistrări);
- se efectuează multe operații de inserare, ștergere sau modificare asupra tabeli. După fiecare astfel de operație sistemul trebuie să actualizeze indexul, operație consumatoare de timp;
- Coloanele indexate sunt referite cel mai adesea ca parte a unor expresii.

### II.9.3. Sinonime

După cum știți sinonimul este un cuvânt cu exact același înțeles cu un alt cuvânt, adică un cuvânt care poate fi folosit în locul altui cuvânt

Similar în dialectul bazelor de date, administratorul unei baze de date poate defini nume echivalente pentru un obiect al bazei de date.

În principal, vom defini un sinonim pentru un obiect al bazei de date pentru a simplifica referirea la acel obiect.

De exemplu, pentru a interoga **tabela1** din schema unui alt utilizator, fie acesta **user1**, atunci vom referi această tabelă prin prefixarea numelui tabelii cu numele utilizatorului în a cărei schemă se găsește tabela, adică vom scrie **user1.tabela1**. Dacă numele utilizatorului este însă **RO\_L2\_SQL01\_S12**, iar tabela se numește **d\_track\_listings**, va trebui să scriem

**RO\_L2\_SQL01\_S12.d\_track\_listings**

pentru a ne referi la acea tabelă, ceea ce este destul de neplăcut. Pentru aceasta vom defini un sinonim mai scurt pentru tabela respectivă.

Sintaxa comenzii de creare a unui sinonim este:

```
CREATE [PUBLIC] SYNONYM nume_sinonim
FOR obiect
```

De exemplu:

```
CREATE SYNONYM ana_track
FOR RO_L2_SQL01_S12.d_track_listings
```

În continuare, vom putea folosi acest sinonim în locul numelui complet al tabelii.

Se pot defini sinonime pentru tabele, vederi, secvențe, proceduri sau alte obiecte ale bazei de date.

Opțiunea **PUBLIC** este folosită de către administratorul bazei de date pentru a crea un sinonim accesibil tuturor utilizatorilor bazei de date. În mod implicit, un sinonim este privat.

Ștergerea unui sinonim se face cu comanda **DROP SYNONYM**.



## Test de autoevaluare

1. Tabela **elevi** a fost creată cu următoarea comandă:

```
CREATE TABLE elevi
( nr_matr NUMBER(5) PRIMARY KEY,
  cnp NUMBER(13) UNIQUE,
  nume VARCHAR2(30),   prenume VARCHAR2(30),
  adresa VARCHAR2(50), telefon CHAR(13) )
```

Pentru ce coloană (sau coloane) se va crea în mod automat un index pentru această tabelă?

- a) doar pentru coloana **nr\_matr**;
- b) doar pentru coloana **cnp**;
- c) doar pentru coloana **nume**;
- d) pentru coloanele **nume** și **prenume**;
- e) pentru coloanele **nr\_matr** și **cnp**.

2. Ce se întâmplă dacă rulați următoarea comandă?

```
CREATE PUBLIC SYNONYM part FOR linda.product;
```

- a) Se creează un nou index.
- b) Este atribuit un privilegiu asupra unui obiect.
- c) Este atribuit un drept de sistem.
- d) Este eliminată necesitatea prefixării numelui obiectului cu numele schemei.

3. Ce se va întâmpla dacă am creat un index pe toate cele șase coloane ale tabelului **elevi**?

- a) Viteza operațiilor de modificare va crește.
- b) Toate interogările care conțin o clauză **WHERE** vor rula mai încet.
- c) Operațiile de ștergere din tabelă se vor executa mai încet.
- d) Toate interogările **SELECT** pe tabela **elevi** vor fi mai rapide.

4. Când un index va scădea viteza unei interogări?

- a) Dacă tabela este mică.
- b) Dacă coloana indexată este utilizată în clauza **WHERE**.
- c) Dacă coloana indexată conține o paletă largă de valori.
- d) Dacă coloana indexată conține un număr mare de valori nule.

5. Care dintre următoarele afirmații este adevărată?

- a) O secvență poate fi folosită doar pentru a genera valori pentru cheia primară a unei tabele.
- b) O secvență poate fi folosită pentru mai multe tabele ale aceleiași scheme.
- c) Crearea unei secvențe face ca numerele să fie memorate în tabelă.
- d) O secvență scade eficiența unei aplicații prin accesarea valorilor memorate în memoria cache.

6. Tabela **elevi** conține următoarele coloane:

```
nr_matr NUMBER(5) PRIMARY KEY,
cnp NUMBER(13) UNIQUE,
nume VARCHAR2(30), prenume VARCHAR2(30),
media NUMBER(4,2)
```

Ați creat o secvență `elevi_sec` pentru a genera valori secvențiale pentru coloana `nr_matr`. Apoi ați modificat structura tabelului `elevi` cu comanda:

```
ALTER TABLE elevi MODIFY (media NUMBER(5,3))
```

Care dintre următoarele afirmații este corectă?

- a) Secvența este ștearsă.
- b) Secvența rămâne nemodificată.
- c) Precizia secvenței este modificată.
- d) Secvența este reinițializată la valoarea minimă.

7. Care dintre următoarele comenzi va afișa următoarea valoare dintr-o secvență `elevi_secv`?

- a) `SELECT NEXTVAL(elevi_secv) FROM SYS.DUAL`
- b) `SELECT elevi_secv.NEXTVAL FROM elevi`
- c) `SELECT elevi_secv.NEXTVAL FROM dual`
- d) `SELECT NEXTVAL(elevi_secv) FROM elevi`
- e) `SELECT elevi_secv NEXTVAL FROM elevi`

8. Ce comandă trebuie să folosească `Ion` pentru a crea un sinonim privat pentru referirea tabelului `elevi` existentă în schema utilizatorului `Vasile`?

- a) `CREATE SYNONYM el FOR vasile.elevi`
- b) `CREATE PUBLIC el SYNONYM FOR vasile.elevi`
- c) `CREATE PUBLIC SYNONYM el FOR vasile.ion`
- d) `CREATE PRIVATE SYNONYM el FOR vasile.elevi`

9. Ați rulat pe rând, cu succes, următoarele comenzi:

```
CREATE SEQUENCE ex;  
SELECT ex.nextval FROM dual;  
INSERT INTO elevi (nr_matr) VALUES (ex.nextval);  
SELECT ex.currval FROM dual;  
ALTER SEQUENCE ex INCREMENT 8;
```

Ce valoare va fi afișată la rularea comenzii ?

```
SELECT ex.nextval FROM dual
```

- a) 3
- b) 4
- c) 10
- d) 8

(vezi baremul de corectare și răspunsurile la pagina 314)



## Alocarea și revocarea drepturilor. Gestiunea tranzacțiilor

# II.10

În acest capitol veți afla:

- |  |  |
|--|--|
| <ol style="list-style-type: none"><li>1. Interogări simple.<br/>Sortarea datelor</li><li>2. Funcții singulare</li><li>3. Interogări multiple</li><li>4. Gruparea datelor</li><li>5. Subinterogări</li><li>6. Crearea și modificarea<br/>structurii tabelor.<br/>Constrângeri</li><li>7. Introducerea și<br/>actualizarea datelor din<br/>tabele</li><li>8. Vederi (views)</li><li>9. Secvențe. Indecși.<br/>Sinonime</li><li>10. Acordarea și revocarea<br/>drepturilor. Gestiunea<br/>tranzacțiilor</li><li>11. Realizarea proiectelor</li><li>12. Aplicații recapitulative</li></ol> | <ul style="list-style-type: none"><li>✓ cum putem controla accesul<br/>utilizatorilor la baza de date</li><li>✓ care sunt categoriile de<br/>drepturi ce se pot atribui<br/>utilizatorilor</li><li>✓ cum acordăm drepturi<br/>utilizatorilor la anumite<br/>obiecte ale bazei de date</li><li>✓ cum revocăm anumite<br/>drepturi ale utilizatorilor</li><li>✓ ce sunt rolurile</li><li>✓ cum se creează și cum se<br/>șterge un rol</li><li>✓ cum se alocă drepturi rolurilor</li><li>✓ cum se acordă un rol unui<br/>utilizator</li><li>✓ ce este o tranzacție</li><li>✓ când începe o tranzacție</li><li>✓ când se termină o tranzacție</li><li>✓ cum devin permanente<br/>modificările făcute într-o<br/>tranzacție asupra bazei de<br/>date</li><li>✓ cum se anulează modificările<br/>făcute într-o tranzacție asupra<br/>bazei de date</li></ul> |
|--|--|

## II.10.1. Drepturi și roluri

V-ați întrebat vreodată ce ar însemna ca elevii dintr-o școală să aibă acces liber la catalog și să poată face orice modificare doresc în catalog? Dar dacă orice utilizator conectat la internet ar avea acces nerestricționat la baza de date a CIA, NASA, a unei bănci și așa mai departe?

Evident, în viața reală accesul în anumite locuri este restricționat. Dacă faci parte dintr-un anumit grup restrâns de persoane, ca de exemplu angajații băncii, poți avea acces în anumite zone restricționate sau la anumite resurse la care alte persoane nu au acces.

Ca și în lumea reală și în cazul bazelor de date trebuie să putem defini o serie de drepturi pentru utilizatorii bazei de date, sau să restricționăm accesul acestora la anumite obiecte ale bazei de date.

Controlul securității în Oracle se asigură prin specificarea: utilizatorilor bazei de date, schemelor, privilegiilor (drepturilor) și rolurilor.

### Utilizatorii bazei de date și schemele

Fiecare bază de date are o listă cu nume de utilizatori. Pentru a accesa baza de date un utilizator trebuie să folosească o aplicație și să se conecteze cu un nume potrivit. Fiecărui nume de utilizator îi este asociată o parolă. Orice utilizator are un domeniu de securitate care determină privilegiile și rolurile, cota de spațiu pe disc alocat și limitele de resurse ce le poate utiliza (timp CPU etc).

### Privilegiile

Privilegiul este dreptul unui utilizator de a executa anumite instrucțiuni SQL. Privilegiile pot fi:

- **privilegii de sistem** - permit utilizatorilor să execute o gamă largă de instrucțiuni SQL, ce pot modifica datele sau structura bazei de date. Aceste privilegii se atribuie de obicei numai administratorilor bazei de date.
- **privilegii de obiecte** - permit utilizatorilor să execute anumite instrucțiuni SQL numai în cadrul schemei sale, și nu asupra întregii baze de date.

Acordarea privilegiilor reprezintă modalitatea prin care acestea pot fi atribuite utilizatorilor. Există două căi de acordare **explicit** (privilegiile se atribuie în mod direct utilizatorilor) și **implicit** (prin atribuirea acestora unor roluri, care la rândul lor sunt acordate utilizatorilor).

## Rolurile

Rolurile sunt grupe de privilegii, care se atribuie utilizatorilor sau altor roluri. Rolurile permit:

- Reducerea activităților de atribuire a privilegiilor. Administratorul bazei de date în loc să atribuie fiecare privilegiu tuturor utilizatorilor va atribui aceste privilegii unui rol, care apoi va fi disponibil utilizatorilor;
- Manipularea dinamică a privilegiilor. Dacă se modifică un privilegiu de grup, acesta se va modifica în rolul grupului. Automat, modificarea privilegiului se propagă la toți utilizatorii din grup;
- Selectarea disponibilităților privilegiilor. Privilegiile pot fi grupate pe mai multe roluri, care la rândul lor pot fi activate sau dezactivate în mod selectiv;
- Proiectarea unor aplicații inteligente. Se pot activa sau dezactiva anumite roluri în funcție de utilizatorii care încearcă să utilizeze aplicația.

Un rol poate fi creat cu parolă pentru a preveni accesul neautorizat la o aplicație. Această tehnică permite utilizarea parolei la momentul pornirii aplicației, apoi utilizatorii pot folosi aplicația fără să mai cunoască parola.

Pentru acordarea unui drept unui anumit utilizator **vasile** se va folosi comanda **GRANT**. De exemplu, pentru a se conecta la baza de date, un utilizator trebuie să aibă permisiunea de a crea o sesiune. Acest drept se alocă de către un utilizator privilegiat (utilizatorul **system** de exemplu) prin comanda

```
GRANT CREATE SESSION TO vasile
```

Acum utilizatorul **vasile** se poate conecta la baza de date.

Revocarea unui drept unui anumit utilizator se face folosind comanda **REVOKE** ca în exemplul următor:

```
REVOKE CREATE SESSION FROM vasile
```

## II.10.2. Drepturile de sistem

Un drept de sistem permite unui utilizator să efectueze anumite operații asupra bazei de date precum executarea comenzilor DDL. Cele mai uzuale drepturi sistem sunt prezentate în tabelul următor:

**Tabelul II.10.1. Privilegii sistem**

Drept	Permite...
CREATE SESSION	conectarea la baza de date
CREATE SEQUENCE	crearea secvențelor
CREATE SYNONYM	crearea sinonimelor
CREATE TABLE	crearea tabelor
CREATE ANY TABLE	crearea unor tabele în orice schemă, nu doar în propria schemă
DROP TABLE	ștergerea tabelor
DROP ANY TABLE	ștergerea unor tabele din orice schemă nu doar din schema proprie
CREATE PROCEDURE	crearea de proceduri memorate
EXECUTE ANY PROCEDURE	executarea unei proceduri în orice schemă
CREATE USER	crearea de utilizatori
DROP USER	ștergerea utilizatorilor
CREATE VIEW	crearea vederilor

## Acordarea drepturilor de sistem

După cum am precizat, acordarea drepturilor se face folosind comanda **GRANT**. În exemplul următor se acordă câteva drepturi sistem utilizatorului **ion**:

```
GRANT CREATE SESSION, CREATE USER, CREATE TABLE TO ion;
```

Se poate de asemenea folosi opțiunea **WITH ADMIN OPTION** care permite unui utilizator să aloce și el drepturile primite cu această opțiune, mai departe, altor utilizatori:

```
GRANT EXECUTE ANY PROCEDURE TO ion WITH ADMIN OPTION;
```

Dreptul acordat utilizatorului **ion**, de a executa orice procedură poate fi acordat de acesta mai departe utilizatorului **george**. Pentru aceasta **ion** se va conecta la baza de date folosind comanda

```
CONNECT ion/test
```

unde **ion** este username-ul, iar **test** este parola și apoi va acorda dreptul lui **george**:

```
GRANT EXECUTE ANY PROCEDURE TO george;
```

Un drept se poate alocă tuturor utilizatorilor bazei de date folosind opțiunea **PUBLIC** ca în următorul exemplu:

```
CONNECT system/manager
```

```
GRANT EXECUTE ANY PROCEDURE TO PUBLIC;
```

În acest moment orice utilizator al bazei de date are dreptul de a executa o procedură în orice schemă.

## II.10.3. Drepturile la nivel de obiect

Un drept la nivel de obiect permite unui utilizator să execute anumite acțiuni asupra obiectelor bazei de date, ca de exemplu executarea anumitor comenzi **DML** pe tabelele bazei de date. **GRANT INSERT ON adm.elevi** permite unui utilizator să insereze linii noi în tabela **elevi** din schema **adm**. Cele mai des întâlnite drepturi la nivel de obiect sunt prezentate în tabelul următor:

**Tabelul II.10.2.** Privilegii la nivel de obiect

Drept	Permite ...
<b>SELECT</b>	Interogarea tabelii
<b>INSERT</b>	Inserarea de noi linii în tabelă
<b>UPDATE</b>	Modificarea valorilor din tabelă
<b>DELETE</b>	Ștergerea datelor din tabelă
<b>EXECUTE</b>	Executarea unor proceduri memorate

## Acordarea drepturilor la nivel de obiect

Veți utiliza de asemenea comanda **GRANT**. Exemplul următor acordă utilizatorului **ion** dreptul de **SELECT**, **INSERT**, și **UPDATE** pe tabela **elevi** și dreptul de **SELECT** asupra tabelii **angajați**:

```
GRANT SELECT, INSERT, UPDATE ON adm.elevi TO ion;
GRANT SELECT ON profesori.angajati TO ion;
```

Următoarea comandă permite utilizatorului **ion** să modifice doar valorile din coloanele **prenume** și **adresa**, din tabela **elevi**, utilizatorului **ion**:

```
GRANT UPDATE (prenume,adresa) ON adm.elevi TO ion;
```

Folosind opțiunea **WITH GRANT OPTION** veți permite utilizatorului să acorde mai departe dreptul primit și altor utilizatori:

```
GRANT SELECT ON adm.elevi TO ion WITH GRANT OPTION;
```

Dreptul de a interoga tabela **adm.elevi** poate fi acum acordat de către **ion** oricărui alt utilizator:

```
CONNECT ion/test
GRANT SELECT ON adm.elevi TO george;
```

**Revocarea drepturilor la nivel de obiect** se va face folosind comanda **REVOKE**. Următoarea comandă revocă dreptul de inserare de noi linii la tabela **elevi** utilizatorului **ion**:

```
REVOKE INSERT ON elevi FROM ion;
```

Comanda va fi rulată din contul **adm**.

**Observație!** Dacă am acordat un drept unui utilizator **A** folosind opțiunea **WITH GRANT OPTION**, iar acest utilizator **A** a acordat și el la rândul lui dreptul altor utilizatori **B**, **C** și **D**, atunci când vom revoca dreptul utilizatorului **A**, va fi revocat automat acel drept și tuturor utilizatorilor cărora utilizatorul **A** le-a acordat acel drept, respectiv utilizatorilor **B**, **C** și **D**.

## II.10.4. Gestiunea rolurilor

După cum am precizat la începutul capitolului, putem crea un rol, prin intermediul căruia vom putea acorda drepturi unui grup de utilizatori având rolul respectiv, lucru mult mai ușor decât acordarea drepturilor fiecărui utilizator separat.

De exemplu, în loc să acordăm drepturi de **select**, **insert** și **update** mai multor utilizatori:

```
GRANT SELECT, INSERT, UPDATE ON adm.elevi TO ion;
GRANT SELECT, INSERT, UPDATE ON adm.elevi TO vasil;
GRANT SELECT, INSERT, UPDATE ON adm.elevi TO gheorghe;
GRANT SELECT, INSERT, UPDATE ON adm.elevi TO maria;
GRANT SELECT, INSERT, UPDATE ON adm.elevi TO alin;
```

e mai comod să creăm un rol, să acordăm drepturi pentru acest rol și apoi să acordăm rolul respectiv celor cinci utilizatori. Vom scrie așadar:

```
CREATE ROLE profi;
GRANT SELECT, INSERT, UPDATE ON adm.elevi TO profi;
GRANT profi TO ion, vasil, gheorghe, maria, alin;
```

În orice moment putem șterge un rol folosind comanda **DROP ROLE**. Aceasta va duce la revocarea tuturor drepturilor acordate utilizatorilor prin intermediul acestui rol.

Să dăm un exemplu mai complex de acordare a drepturilor și privilegiilor. Să presupunem că rulăm pe rând următoarele comenzi:

```
CONNECT hr/test;
CREATE ROLE r1;
CREATE ROLE r2;
GRANT SELECT, INSERT, DELETE ON hr.elevi TO r1
    WITH GRANT OPTION;
GRANT DELETE, UPDATE ON hr.elevi TO r2
    WITH GRANT OPTION;
GRANT r1 TO user1
```

```

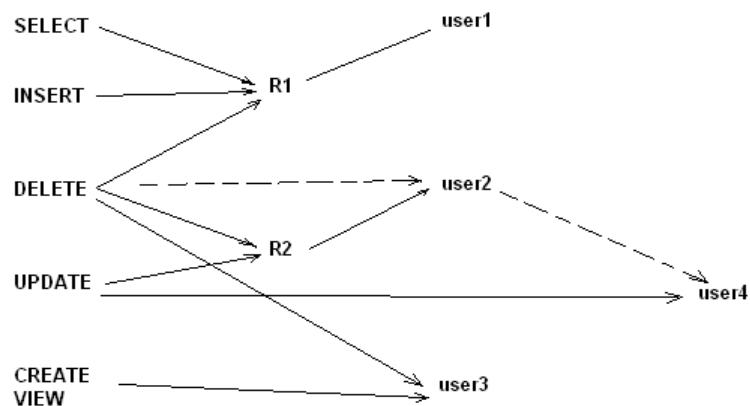
GRANT r2 TO user2
GRANT CREATE VIEW TO user3 WITH GRANT OPTION
GRANT DELETE ON hr.elevi TO user3
GRANT UPDATE ON hr.elevi TO user4
CONNECT user2/pas2
GRANT DELETE ON hr.elevi TO user4
GRANT UPDATE ON hr.elevi TO user4

```

În acest moment utilizatorii au următoarele drepturi (figura II.10.1.):

**Tabelul II.10.3.**

UTILIZATOR	DREPT
user1	SELECT, INSERT, DELETE ON hr.elevi
user2	DELETE, UPDATE ON hr.elevi
user3	DELETE ON hr.elevi CREATE VIEW
user4	DELETE, UPDATE ON hr.elevi



**Figura II.10.1.** Schema de acordare a drepturilor

Dacă acum ștergem rolul **r2**:

```
DROP ROLE r2
```

utilizatorul **user2** va pierde dreptul de **DELETE** și **UPDATE** asupra tabelii **hr.elevi** și prin intermediul său va pierde dreptul de **DELETE** și utilizatorul **user4**, care a primit acest drept de la **user2**. Deși **user4** a primit de la **user2** și dreptul de **UPDATE**, el nu va pierde acest drept deoarece a primit acest drept și direct de la utilizatorul **SYSTEM**. Așadar, după ștergerea rolului **r2**, drepturile utilizatorilor sunt următoarele:

Tabelul II.10.4.

UTILIZATOR	DREPT
user1	SELECT, INSERT, DELETE ON hr.elevi
user2	-
user3	DELETE ON hr.elevi CREATE VIEW
user4	UPDATE ON hr.elevi

## II.10.5. Gestiunea tranzacțiilor

O tranzacție este un grup de comenzi SQL care sunt văzute ca o singură unitate. Imaginați-vă o tranzacție ca un grup de comenzi SQL care nu pot fi separate, și al căror efect este în întregime salvat în baza de date, fie este în întregime anulat. Să ne gândim la efectuarea unui transfer bancar dintr-un cont în alt cont. O comandă **UPDATE** va efectua operația de scădere a sumei de bani tranzacționată dintr-un cont, iar o altă comandă **UPDATE** va adăuga suma respectivă la cel de al doilea cont. Dacă ambele operații decurg normal fără probleme, atunci ele vor deveni **ambele** permanente. Dacă una dintre aceste două comenzi eșuează (de exemplu nu poate fi contactată banca în care se depun banii) atunci ambele comenzi vor fi anulate. E normal să renunțăm la scăderea sumei de bani dintr-un cont, dacă aceștia nu pot fi depuși în celălalt cont, în caz contrar ar duce la pierderea banilor respectivi.

În general, o tranzacție poate fi formată din mai multe comenzi **INSERT**, **UPDATE** și **DELETE**.

Pentru a face permanentă o tranzacție folosiți comanda **COMMIT**. Dacă doriți să renunțați la modificările efectuate în cadrul unei tranzacții trebuie să rulați o comandă **ROLLBACK**.

Comanda **ROLLBACK** fără nici un parametru, încheie tranzacția curentă și renunță la toate modificările făcute în cadrul acestei tranzacții. Aveți însă posibilitatea definirii în cadrul unei tranzacții a unui așa numit punct de întoarcere, sau punct de salvare. Odată definit un astfel de punct de salvare, veți putea renunța doar la o parte dintre modificările făcute în cadrul tranzacției curente.

Definirea unui punct de revenire se face cu comanda **SAVEPOINT** având sintaxa:

**SAVEPOINT nume\_punct\_de\_revenire**

Revenirea la un punct de revenire se face cu comanda **ROLLBACK** astfel:

**ROLLBACK TO nume\_punct\_de\_revenire**

Definirea punctelor de revenire este utilă în cazul unor tranzacții mari, când în cazul în care faceți o greșeală nu trebuie să renunțați la toate operațiile din cadrul tranzacției ci doar la o parte dintre acestea.



O tranzație, fiind un grup de comenzi SQL tratate ca un întreg, trebuie să stabilim unde începe o tranzație și unde se termină aceasta.

O tranzație începe la întâlnirea unuia dintre următoarele evenimente:

- În momentul conectării la baza de date și la începerea rulării primei comenzi **DML** (**INSERT**, **UPDATE**, **DELETE**).
- La terminarea unei tranzații anterioare și la rularea următoarei comenzi **DML**.

O tranzație se termină când apare unul dintre următoarele evenimente:

- La executarea unei comenzi **COMMIT** sau **ROLLBACK** (fără nici un parametru, întrucât **ROLLBACK TO ...** nu termină tranzația ci doar revine la un punct precizat din cadrul tranzației curente)
- La executarea unei comenzi **DDL** (**CREATE**, **ALTER**, **DROP**, **RENAME**, **TRUNCATE**), caz în care este executată automat comanda **COMMIT**.
- La executarea unei comenzi **DCL** (**GRANT** sau **REVOKE**) caz în care este executată automat comanda **COMMIT**.
- Vă deconectați de la baza de date. Dacă ieșiți normal din **SQL\*Plus** cu comanda **Exit**, sau dați **Logout** din Oracle Database Express Edition atunci are loc un **COMMIT** automat. Dacă ieșirea se face anormal, cum ar fi în cazul unei pene de curent, atunci se execută în mod automat o comandă **ROLLBACK**.
- Executați o comandă **DML** care eșuează, caz în care are loc un **ROLLBACK** automat pentru acea singură comandă.

Să experimentăm acum modul de folosire a tranzațiilor.

**Atenție!** În Oracle Database Express Edition toate comenzile sunt autocommit, și nu vor fi recunoscute comenzile **COMMIT**, **ROLLBACK** sau **SAVEPOINT**. Pentru acest exercițiu puteți rula comenzile SQL în linia de comandă. Pentru aceasta alegeți din meniul **Start, Programs, Oracle Database 10g Express Edition** opțiunea **Run SQL Command Line**. Se va deschide o fereastră în care vă veți conecta la baza de date folosind comanda

**CONNECT**

Introduceți username-ul (**hr**) și parola și în acest moment puteți rula orice comandă SQL.

Pentru a experimenta folosirea tranzațiilor vom crea următoarea tabelă:

```
create table savepoint_test ( n number )
```

Inserăm acum câteva linii în această tabelă:

```
insert into savepoint_test values (1);  
insert into savepoint_test values (2);  
insert into savepoint_test values (3);
```

Definim acum un punct de salvare:

```
savepoint sp1;
```

și mai inserăm câteva linii în tabelă:

```
insert into savepoint_test values (10);  
insert into savepoint_test values (20);  
insert into savepoint_test values (30);
```

Definim un nou punct de salvare:

```
savepoint sp2;
```

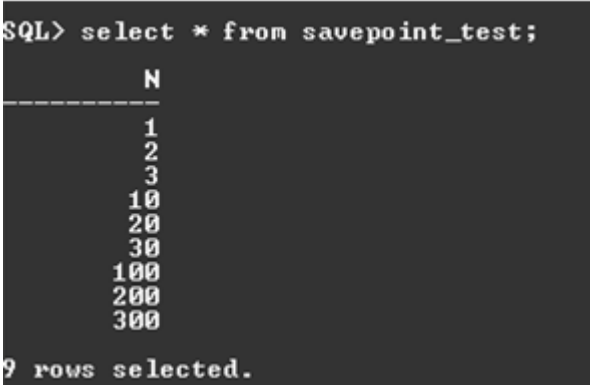
și inserăm în final încă trei linii:

```
insert into savepoint_test values (100);  
insert into savepoint_test values (200);  
insert into savepoint_test values (300);
```

Verificăm acum dacă datele au fost inserate în tabelă:

```
select * from savepoint_test;
```

și vedem că toate datele au fost inserate:



```
SQL> select * from savepoint_test;  
  
-----  
N  
-----  
1  
2  
3  
10  
20  
30  
100  
200  
300  
  
9 rows selected.
```

**Figura II.11.1.**

Revenim acum la punctul de revenire sp2

```
ROLLBACK TO sp2
```

și verificăm conținutul tablei:

```
select * from savepoint_test;
```

Observați că ultimele linii inserate după definirea punctului de salvare sp2 au fost șterse din tabelă (figura II.11.2.).

```

SQL> rollback to sp2;
Rollback complete.
SQL> select * from savepoint_test;

-----
      N
-----
      1
      2
      3
     10
     20
     30
6 rows selected.
SQL>

```

Figura II.11.2.

Inserăm alte trei linii:

```

insert into savepoint_test values (111);
insert into savepoint_test values (222);
insert into savepoint_test values (333);

```

testăm conținutul tabelii:

```

select * from savepoint_test;

```

```

SQL> insert into savepoint_test values(333);
1 row created.
SQL> select * from savepoint_test;

-----
      N
-----
      1
      2
      3
     10
     20
     30
     11
     22
     33
9 rows selected.

```

Figura II.11.3.

Revenim la punctul de salvare `sp2`:

```

ROLLBACK TO sp2

```

și verificăm conținutul tabelii:

```

select * from savepoint_test;

```

Evident ultimele trei linii nu se mai găsesc în tabelă conținutul tabelului fiind același cu cel din figura II.11.2. Dacă revenim acum la punctul de salvare sp1, în tabelă nu mai rămân decât trei linii (figura II.11.4.)

```
ROLLBACK TO sp1
select * from savepoint_test;
```

```
SQL> select * from savepoint_test;

----- N
      1
      2
      3
     10
     20
     30

6 rows selected.
SQL> rollback to sp1;
Rollback complete.
SQL> select * from savepoint_test;

----- N
      1
      2
      3

SQL>
```

Figura II.11.4.

Schematic, tranzacția anterioară arată ca în figura II.11.5:

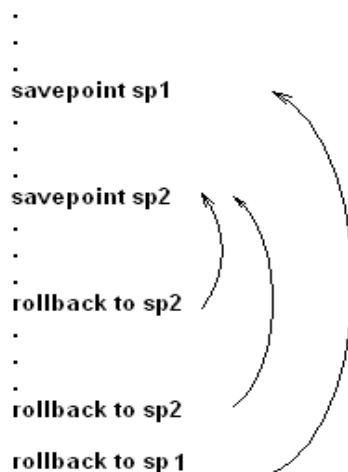


Figura II.11.5.



## Aplicație

Aflați care este conținutul tabelului `tab1` după rularea următoarelor comenzi.

```
CREATE TABLE tab1 (n1 NUMBER(3))
INSERT INTO tab1 VALUES (1);
SAVEPOINT s1;
INSERT INTO tab1 VALUES (2);
SAVEPOINT s2;
INSERT INTO tab1 VALUES (3);
ROLLBACK to s2;
INSERT INTO tab1 VALUES (4);
ALTER TABLE tab1 MODIFY (n1 NUMBER(5));
INSERT INTO tab1 VALUES (5);
SAVEPOINT s3;
INSERT INTO tab1 VALUES (6);
ROLLBACK to s3;
INSERT INTO tab1 VALUES (7);
SAVEPOINT s4;
INSERT INTO tab1 VALUES (8);
ROLLBACK;
INSERT INTO tab1 VALUES (9);
COMMIT;
```

1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
- 11. Realizarea proiectelor**
12. Aplicații recapitulative

În acest capitol veți afla:

- ✓ cum să creați în mod interactiv tabelele bazei de date
- ✓ cum să creați pagina principală a aplicației folosind Application Builder
- ✓ cum să creați și să modificați formularele și rapoartele aplicației

În acest capitol vom prezenta câteva elemente de bază privind realizarea proiectelor folosind HTML DB Application Builder, o facilitate puternică oferită de Oracle Database 10g Express Edition.

Vă propunem ca exercițiu crearea unei aplicații care să gestioneze informațiile despre echipele de fotbal din campionatul național și jucătorii acestor echipe.

### II.11.1. Crearea tabelelor bazei de date

Puteți opta pentru scrierea comenzii de creare a fiecărei tabelă în parte, sau puteți opta pentru crearea interactivă a acestora. Pentru aceasta accesați din pagina principală a aplicației Oracle Database Express Edition, opțiunea **Object Browser**, iar în fereastra ce se deschide dați clic pe săgeata din dreapta butonului "Create" și alegeți opțiunea table (fig. II.11.1).

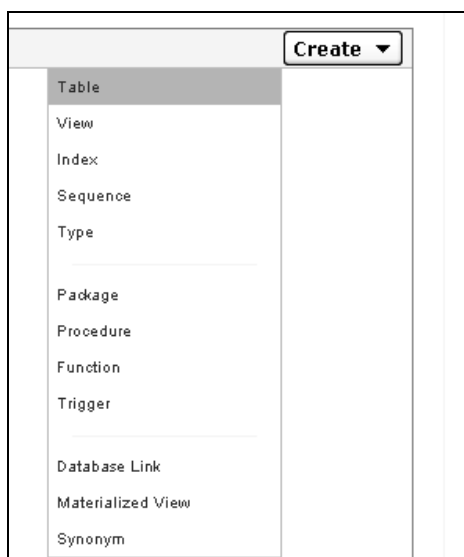


Figura II.11.1.

Completați apoi informațiile pentru crearea tabelă **Echipe** (figura II.11.2). După completarea acestor date acționați butonul **Next**, și alegeți opțiunea "Populated from a new sequence", ceea ce înseamnă că valoarea cheii primare va fi completată automat prin intermediul unei secvențe.

În fereastra următoare apăsați **Next** fără nicio modificare, iar apoi **Finish** și în final **Create**.

Create Table Cancel Next >

\* Table Name  ☐ Preserve Case

Column Name	Type	Precision	Scale	Not Null	Move
<input type="text" value="Cod"/>	NUMBER	<input type="text" value="3"/>	<input type="text"/>	<input type="checkbox"/>	▼ ▲
<input type="text" value="Nume"/>	VARCHAR2		<input type="text" value="40"/>	<input type="checkbox"/>	▼ ▲
<input type="text" value="Localitate"/>	VARCHAR2		<input type="text" value="40"/>	<input type="checkbox"/>	▼ ▲
<input type="text" value="Adresa"/>	VARCHAR2		<input type="text" value="50"/>	<input type="checkbox"/>	▼ ▲
<input type="text"/>	- Select Datatype -				▼ ▲
<input type="text"/>	- Select Datatype -				▼ ▲
<input type="text"/>	- Select Datatype -				▼ ▲
<input type="text"/>	- Select Datatype -				▼ ▲

Add Column

**Figura II.11.2.** Crearea interactivă a tabelelor

Structura tabelii **Echipe** este următoarea:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
<a href="#">ECHIPE</a>	<a href="#">COD</a>	Number	-	3	0	1	-
	<a href="#">NUME</a>	Varchar2	40	-	-	-	✓
	<a href="#">LOCALITATE</a>	Varchar2	40	-	-	-	✓
	<a href="#">ADRESA</a>	Varchar2	50	-	-	-	✓
	<a href="#">TELEFON</a>	Char	13	-	-	-	✓

Repetati apoi pașii anteriori pentru crearea tabelii **Jucători** care are următoarea structură:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
<a href="#">JUCATORI</a>	<a href="#">ID</a>	Number	-	3	0	1	-
	<a href="#">NUME</a>	Varchar2	40	-	-	-	-
	<a href="#">PRENUME</a>	Varchar2	40	-	-	-	-
	<a href="#">DATAN</a>	Date	7	-	-	-	✓
	<a href="#">TELEFON</a>	Char	13	-	-	-	✓
	<a href="#">EMAIL</a>	Varchar2	40	-	-	-	✓
	<a href="#">IDECHIPA</a>	Number	-	3	0	-	✓

De această dată trebuie ca după definirea cheii primare să definiți și cheia străină, care face referire la codul echipei în care joacă un jucător (fig. II.11.3).



Foreign Keys Cancel < Previous Next >

Foreign Key	Columns	Referenced Table	Referenced Columns	Action
JUCATORI_FK	IDECHIPA	ECHIPE	COD	cascade ✕

Add Foreign Key Add

\* Name: JUCATORI\_fk

☐ Disallow Delete  
☒ Cascade Delete  
☐ Set Null on Delete

Select Key Column(s): ID, Nume, Prenume, Data, Telefon

\* Key Column(s): IDECHIPA

\* References Table: ECHIPE

Select Reference Column(s): Nume, Localitate, Adresa

\* Referenced Column(s): COD

**Figura II.11.3.** Crearea interactivă a cheii străine

Tot din fereastra **Object Browser** puteți să și populați cu date tabelele bazei de date. Pentru aceasta veți da click pe numele tablei, iar apoi din panoul din partea dreaptă alegeți opțiunea **Data** și **Insert Row** (fig. II.11.4).

Tables Search Refresh

COUNTRIES  
CUVINTE  
DEPARTMENTS  
DEPT  
ECHIPE  
EMP  
EMPLOYEES  
JOBS  
JOB\_HISTORY  
JUCATORI  
LOCATIONS  
REGIONS

**JUCATORI**

Table Data Indexes Model Constraints Grants Statistics UI Default

Add Column Modify Column Rename Column Drop Column Rename

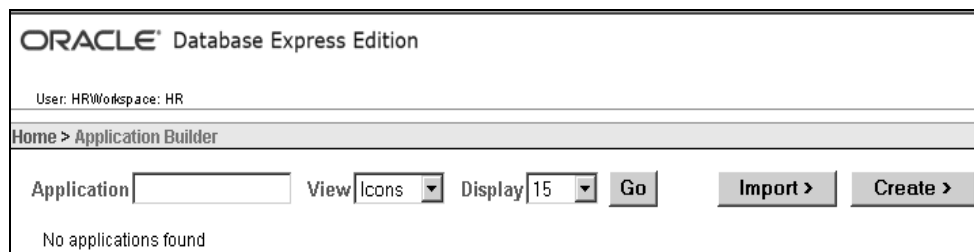
Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(3,0)	No	-	1
NUME	VARCHAR2(40)	No	-	-
PRENUME	VARCHAR2(40)	No	-	-
DATAN	DATE	Yes	-	-
TELEFON	CHAR(13)	Yes	-	-
EMAIL	VARCHAR2(40)	Yes	-	-
IDECHIPA	NUMBER(3,0)	Yes	-	-

1 - 7

**Figura II.11.4.**

## II.11.2. Crearea aplicației și a paginii principale

Reveniți acum în pagina principală și alegeți opțiunea **Application Builder**.

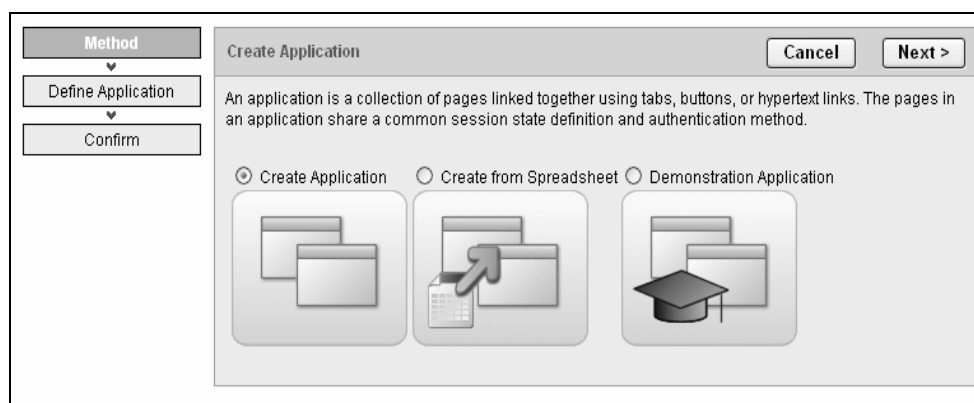


The screenshot shows the Oracle Database Express Edition interface. At the top, it says "ORACLE® Database Express Edition". Below that, it says "User: HRWorkspace: HR". The breadcrumb navigation shows "Home > Application Builder". There is a search bar labeled "Application" with a dropdown menu set to "Icons" and a "Display" dropdown set to "15". There are "Go", "Import >", and "Create >" buttons. Below the search bar, it says "No applications found".

**Figura II.11.5.** Pagina principală **Application Builder**

Folosind **Create Application Builder**, vom crea pagina principală a aplicației care va include un formular și un raport asociat tabelii **Echipe** și un raport și formular asociat tabelii **Jucători**. Pentru acest lucru realizați următorii pași:

1. Acționați butonul **Create** din pagina Application Builder.
2. Acceptați opțiunea implicită, **Create Application** (fig. II.11.6) și apăsați **Next**.
3. Completați numele aplicației cu **Campionatul de fotbal** și treceți la următoarea pagină.



The screenshot shows the "Create Application" dialog box. On the left, there is a "Method" section with a dropdown menu set to "Define Application" and a "Confirm" button. The main area of the dialog box has a title bar "Create Application" with "Cancel" and "Next >" buttons. Below the title bar, there is a text description: "An application is a collection of pages linked together using tabs, buttons, or hypertext links. The pages in an application share a common session state definition and authentication method." There are three radio buttons: "Create Application" (selected), "Create from Spreadsheet", and "Demonstration Application". Below the radio buttons, there are three icons: a folder icon, a folder icon with an arrow pointing to it, and a folder icon with a graduation cap on top.

**Figura II.11.6.**

4. La tipul paginii alegeți **Blank**, introduceți în caseta Page name **Home** și dați click pe **Add Page**.

5. Vom adăuga acum un formular la această pagină pe baza tabelului **ECHIPE**.
  - a. Selectați "Report and form" ca tip de pagină;
  - b. Din caseta **Subordinate to Page** alegeți opțiunea **Home** iar în caseta **Table Name** introduceți numele tabelului **Echipe**;
  - c. Click pe **Add Page**.

Observați că au fost adăugate în lista paginilor două link-uri, unul către raportul **Echipe** și subordonat acestuia, formularul **Echipe**. De fapt am creat un raport în care avem posibilitatea de a actualiza datele din tabelă, de aceea am realizat implicit și un formular.

6. Repetați pasul anterior pentru a adăuga un formular și un raport și pentru tabela **Jucători** și apoi apăsați butonul **Next**.
7. La pasul următor al wizard-ului optăm pentru varianta fără tab-uri "**No Tabs**".
8. În pagina **Shared Components** alegem **No** și apăsăm de două ori la rând pe **Next**.
9. Alegem modelul interfeței pentru aplicație și apăsăm **Next** și apoi **Create**.

Page	Page Name	Page Type	Source Type	Source
1	Home	Blank	-	-
2	ECHIPE	Report	Table	ECHIPE
3	ECHIPE	Form	Table	ECHIPE
4	JUCATORI	Report	Table	JUCATORI
5	JUCATORI	Form	Table	JUCATORI

**Add Page**

Select Page Type:

☐ Blank
 ☐ Report
 ☐ Form
 ☐ Tabular Form
 ☒ Report and Form

Action: Add a report with an edit form on a second page

Subordinate to Page: - Top Level Page -

Table Name:

☐ Include Analysis Pages

**Figura II.11.7.**

Acum avem creată aplicația. Pentru a o rula, acționăm butonul **Run Application** (fig. II.11.8, II.11.9).

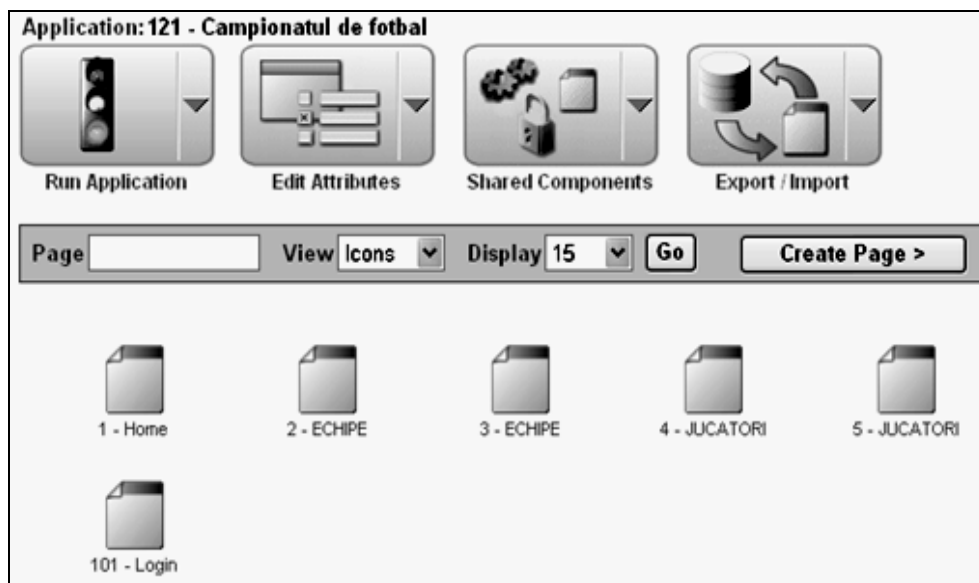


Figura II.11.8.

Figura II.11.9.

Se observă că implicit a fost creată o pagină de **login**, unde veți introduce user-ul și parola voastră (implicit **HR** cu parola setată de voi la instalare).

În timpul rulării aplicației putem oricând reveni la editarea acesteia prin acționarea butonului "**Edit Application**" sau a butonului "**Edit Page...**", dacă dorim să edităm pagina curentă.

Din formularul creat (fig. II.11.10.) putem sorta datele după o coloană dând clic pe antetul coloanei respective, sau putem modifica datele dintr-o linie prin acționarea butonașului în formă de creion din dreptul liniei respective, astfel putem adăuga noi linii, etc.

ECHIBE

Reset

Create

Search

Display

15

Go

	Nume▲	Localitate	Telefon	Adresa
	CFR Cluj	Cluj Napoca	0264-598833	Str. Republicii nr. 109, Cluj-Napoca, cod 400489
	Dinamo	Bucuresti	021-2303485	Calea Floreasca, Nr. 18-20, Bucuresti, Sector 1
	FC Arges	Pitesti	0248217214	Strada Armand Calinescu nr. 15, Pitesti
	FC National	Bucuresti	410.66.06	Dr. Lister nr. 37, sector 5, Bucuresti
	FC Vaslui	Vaslui	0235-316097	Str. Decebal 1, Vaslui 730227
	Farul	Constanta	0241-616.142	Str.Primaverii nr. 2, Constanta
	Gloria Bistrita	Bistrita	0263-212998	Gloria Bistrita
	Jiul Petrosani	Petrosani	0254-545472	Str. Lunca Nr. 100, 332061 Petrosani
	Otelul Galati	Galati	0236.464677	Galati, str. Anghel Saligny nr. 2 Galati 800686
	Pandurii Tg. Jiu	Tirgu Jiu	0253/223588	Str. Victoriei, nr. 22
	Politehnica Iasi	Iasi	0232.213.018	Aleea Grigore Ghica Voda nr. 12-24 Iasi
	Politehnica Timisoara	Timisoara	0256/309.852	Bd. Regele Ferdinand I nr. 2 Timisoara
	Rapid	Bucuresti	021-223.44.90	Bd. Bucurestii Noi, Nr. 170, Bucuresti
	Steaua	Bucuresti	021.411.46.56	Blvd. Ghencea nr. 45 sector 6 Bucuresti
	UTA	Arad	0257/289.204	Calea Aurel Vlaicu, nr. 39

row(s) 1 - 15 of 16

Next

Figura II.11.10. Formularul/raport Echipe

### II.11.3. Adăugarea câmpurilor calculate unui formular sau raport

Dorim acum ca la raportul **Echipe** să adăugăm un nou câmp care să afișeze numărul de jucători de la fiecare echipă, memorați în tabela **Jucători**.

- Dați clic pe **Edit Page 2** din partea de jos a ecranului în care este afișat raportul **Echipe**.
- Localizați secțiunea **Regions**. Apăsați pe linkul **Echipe** din această zonă.

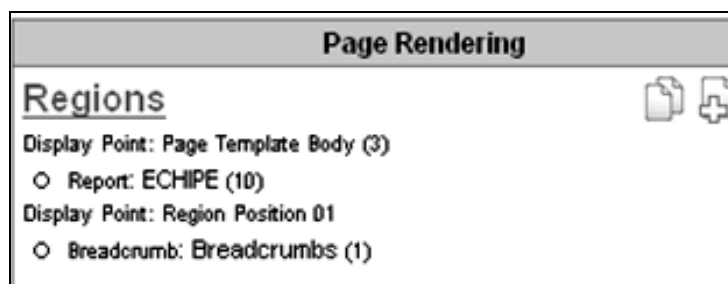


Figura II.11.11. Secțiunea Regions

3. În caseta Region Source, modificați comanda **SELECT** ca mai jos după care vom apăsa butonul "Apply Changes".

```
select e.cod "Cod", e.numa "Nume",
       e.localitate "Localitate",
       e.telefon "Telefon", e.adresa "Adresa",
       count(j.id) "Numar jucatori"
from ECHIPE e, jucatori j
where e.cod=j.idechiba(+) and
(instr(upper(e.numa),
       upper(nvl(:P2_REPORT_SEARCH,e.NUMA))) > 0 or
instr(upper(e.LOCALITATE),
       upper(nvl(:P2_REPORT_SEARCH,e.LOCALITATE))) > 0 or
instr(upper(e.ADRESA),
       upper(nvl(:P2_REPORT_SEARCH,e.ADRESA))) > 0 or
instr(upper(e.TELEFON),
       upper(nvl(:P2_REPORT_SEARCH,e.TELEFON))) > 0 )
group by e.cod, e.numa, e.localitate, e.telefon, e.adresa
```

Prin această modificare însă au dispărut butonașele în formă de creion care permiteau editarea datelor din formular (fig. II.11.12). Pentru a rezolva acest lucru vom parcurge încă câțiva pași suplimentari după cum urmează:

4. Apăsăm acum pe cuvântul **Report** din secțiunea **Regions** al paginii 2, adică a raportului **Echipe**. Se va afișa pagina din figura II.11.13.
5. Dați click pe butonul sub formă de creion din fața câmpului **Cod**.
6. În secțiunea **Column Link** a acestei pagini dați click pe butonul [Icon1] de sub caseta "Link Text" pentru ca în raportul **Echipe** în fața fiecărei linii, să apară butonașul sub formă de creion.

Report ECHIBE

Reset

Create

Search

Display 15

Go

Cod	Nume	Localitate	Telefon	Adresa	Numar Jucatori
47	Universitatea Craiova	Craiova	0251/414726	Strada Sfântul Dumitru, Nr. 1	0
46	Jiul Petrosani	Petrosani	0254-545472	Str. Lunca Nr. 100, 332061 Petrosani	0
43	FC Arges	Pitesti	0248217214	Strada Armand Calinescu nr. 15, Pitesti	0
49	Pandurii Tg. Jiu	Tirgu Jiu	0253/223588	Str. Victoriei, nr. 22	2
25	FC National	Bucuresti	410.66.06	Dr. Lister nr. 37, sector 5, Bucuresti	0
42	Politehnica Iasi	Iasi	0232.213.018	Aleea Grigore Ghica Voda nr. 12-24 Iasi	0
45	FC Vaslui	Vaslui	0235-316097	Str. Decebal 1, Vaslui 730227	0
22	Farul	Constanta	0241-616.142	Str.Primaverii nr. 2, Constanta	0
27	Politehnica Timisoara	Timisoara	0256/309.852	Bd. Regele Ferdinand I nr. 2 Timisoara	0
21	Steaua	Bucuresti	021.411.46.56	Blvd. Ghencea nr. 45 sector 6 Bucuresti	0
26	CFR Cluj	Cluj Napoca	0264-598833	Str. Republicii nr. 109, Cluj-Napoca, cod 400489	0
24	Rapid	Bucuresti	021-223.44.90	Bd. Bucurestii Noi, Nr. 170, Bucuresti	0
44	Gloria Bistrita	Bistrita	0263-212998	Gloria Bistrita	0
23	Dinamo	Bucuresti	021-2303485	Calea Floreasca, Nr. 18-20, Bucuresti, Sector 1	2
48	UTA	Arad	0257/289.204	Calea Aurel Vlaicu, nr. 39	0

row(s) 1 - 15 of 16

Next

Figura II.11.12.

Region Definition

Report Attributes

Report Attributes

Cancel

Apply Changes

Column Attributes

Layout and Pagination

Sorting

Messages

Report Export

External Processing

Break Formatting

Column Attributes

Headings Type:

Column Names

Column Names (InitCap)

Custom

PL/SQL

None

Alias	Link	Edit	Heading	Column Alignment	Heading Alignment	Show	Sum	Sort	Sort Sequence
Cod			Cod	left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-
Nume			Nume	left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-
Localitate			Localitate	left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-
Telefon			Telefon	left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-
Adresa			Adresa	left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-
Numar jucatori			Numar Jucatori	left	center	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	-

When moving the last column further down, it will show up as the first column of your report.

When moving the first column up, it will be moved to the end of your report.

Layout and Pagination

Report Template

template: 7. Standard

HTML1

ILook 11

ILook 21

ILook 31

ILook 41


ICSV1

IXML1

Figura II.11.13. Pagina Report Attributes


7. În caseta **Page** introduceți valoarea 3, însemnând că la apăsarea butonului din fața unei linii se va deschide formularul 3 care ne permite să edităm datele din linia respectivă.
8. Pentru ca formularul 3 să preia datele din linia pe care o edităm, trebuie să precizăm în caseta **Item 1** câmpul din formularul 3 care va face legătura dintre raport și formular (**P3\_COD**) iar în caseta **Value** de unde se va prelua valoarea de legătură și anume valoarea curentă a câmpului cod din raport (**#Cod#**) (fig. II.11.14).







**Column Link**

Link Text:  

Link Attributes:

Target:  Page:  ☐ Reset Pagination

Request:  Clear Cache:  

	Name		Value	
Item 1	<input type="text" value="P3_COD"/>		<input type="text" value="#Cod#"/>	
Item 2	<input type="text" value=""/>		<input type="text" value=""/>	
Item 3	<input type="text" value=""/>		<input type="text" value=""/>	

Page Checksum:


**Figura II.11.14. Secțiunea Column Link**

9. Acționați butonul **"Apply Changes"** din partea de sus a paginii.
10. Testați aplicația realizată până la momentul actual.

## II.11.4. Crearea listelor de valori

Ați încercat să adăugați câțiva jucători noi? Ce observație puteți face? Precis v-ați lovit de problema codului echipelor. Doriți să adăugați un jucător la o echipă dar din păcate nu cunoașteți codul acesteia.

Pe această problemă o vom rezolva în continuare prin definirea unei liste cu toate echipele din tabela **Echipe**.

1. În pagina de definire a proprietăților raportului **Jucători** (clic pe **"4-Jucători"** din pagina principală de editare a aplicației) apăsați butonul  din secțiunea **"List of Values"**.



2. Apăsați **Next**, iar în următoarea fereastră introduceți un nume pentru lista de valori, de exemplu **listaEchipe** și alegeți opțiunea **Dynamic**.
3. În pagina următoare introduceți următoarea comandă select:

```
Select nume d, cod v FROM echipe
ORDER BY d
```

și apăsați butonul **"Create List Of Value"**

Acum trebuie să modificăm câmpul **IdEchipă** din raportul **Jucători** pentru a afișa numele echipei, nu codul acesteia.

1. În pagina de definire a proprietăților raportului **Jucători** (clic pe **"4-Jucători"** din pagina principală de editare a aplicației) apăsați link-ul **Report** din secțiunea **Regions**.
2. Apăsați butonul de editare din fața câmpului **IdEchipă**.
3. În secțiunea **Tabular Form Element** selectați din lista **Display As** opțiunea **Display As Text (based on LOV, does not save state)**. Aceasta înseamnă că în raport se va afișa valoarea corespunzătoare din lista de valori, însă nu o putem modifica din această fereastră (figura II.11.15.).
4. În secțiunea **List Of Values**, selectați numele listei (**listaEchipe**) în caseta **Named LOV**, asigurați-vă că este aleasă opțiunea **Yes** pentru **Display null**, iar în caseta **NULL Text** introduceți textul **"- neassignat unei echipe -"** (figura II.11.15.).
5. Apăsați **Apply Changes**

O modificare asemănătoare trebuie făcută și formularului **Jucători**.

1. În pagina de definire a proprietăților formularului **Jucători** (clic pe **"5-Jucători"** din pagina principală de editare a aplicației) apăsați link-ul **P5\_IDECHIPA** din secțiunea **Items**.
2. Alegeți la **Display As** opțiunea **Select List**.
3. În secțiunea **List Of Values**, selectați numele listei (**listaEchipe**) în caseta **Named LOV**, asigurați-vă că este aleasă opțiunea **Yes** pentru **Display null**, iar în caseta **NULL display value** introduceți textul **"- neassignat unei echipe -"**.
4. Apăsați **Apply Changes**

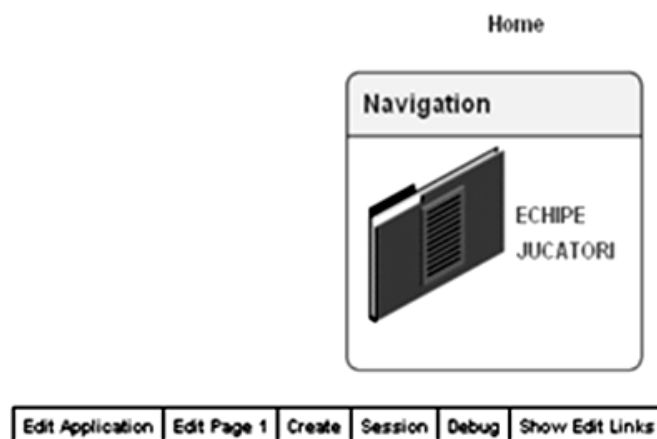
Tabular Form Element	
Display As	Display as Text (based on LOV, does not save state)
Date Picker Format Mask	- Select Date Format -
Element Width	Number of Rows
Element Attributes	
Element Option Attributes	
Default Type	No Default
Default	
Reference Table Owner	HR
Reference Table Name	JUCATORI
Reference Column Name	IDECHIPA

List of Values	
Named LOV	LISTAECHIBE
Display Null	Yes
Null Text	- neassignat unei e
Display Extra Value	Yes
Null Value	
LOV Query (select DISPLAY_VALUE, RETURN_VALUE from ...)	

**Figura II.11.15.** Secțiunea Column Link

În acest moment putem spune că aplicația noastră este terminată și funcționează așa cum ne doream:



**Figura II.11.16.** Pagina Home a aplicației

Home > JUCATORI

Action Processed.

JUCATORI

Reset
Create

Search  Display 15

	Nume ▲	Prenume	Echipa	Telefon	Email	Data
	Dragan	Persic	Pandurii Tg. Jiu			
	Ionescu	Claudiu	Politehnica Iasi			02-FEB-89
	Naicu	Victor	Pandurii Tg. Jiu			
	Niculescu	Claudiu	UTA			
	Serban	Denis	Dinamo			

1 - 5

[Spread Sheet](#)

Edit Application
Edit Page 4
Create
Session
Debug
Show Edit Links

Figura II.11.17. Pagina raportului Jucători

Home > JUCATORI > JUCATORI

JUCATORI

Cancel
Delete
Apply Changes

Nume

Prenume

Data

Telefon

Email

Echipa

Edit Application
Edit Page 5
Create
Session
Debug
Show Edit Links

Figura II.11.18. Pagina formularului Jucători

Home > ECHIPE

ECHIPE Reset Create

Search  Display 15 Go



Cod	Name	Localitate	Telefon	Adresa	Numar Jucatori
	Universitatea Craiova	Craiova	0251/414726	Strada Sfântul Dumitru, Nr. 1	0
	Jiul Petrosani	Petrosani	0254-545472	Str. Lunca Nr. 100, 332061 Petrosani	0
	FC Arges	Pitesti	0248217214	Strada Armand Calinescu nr. 15, Pitesti	0
	Politehnica Iasi	Iasi	0232.213.018	Aleea Grigore Ghica Voda nr. 12-24 Iasi	1
	Pandurii Tg. Jiu	Tingu Jiu	0253/223588	Str. Victoriei, nr. 22	2
	FC National	Bucuresti	410.66.06	Dr. Lister nr. 37, sector 5, Bucuresti	0
	FC Vaslui	Vaslui	0235-316097	Str. Decebal 1, Vaslui 730227	0
	Farul	Constanta	0241-616.142	Str.Primaverii nr. 2, Constanta	0
	Politehnica Timisoara	Timisoara	0256/309.852	Bd. Regele Ferdinand I nr. 2 Timisoara	0
	Steaua	Bucuresti	021.411.46.56	Blvd. Ghencea nr. 45 sector 6 Bucuresti	0
	CFR Cluj	Cluj Napoca	0264-598833	Str. Republicii nr. 109, Cluj-Napoca, cod 400489	0

Figura II.11.19. Pagina raportului Echiipe

Home > ECHIPE > ECHIPE

ECHIPE Cancel Delete Apply Changes

Name

Localitate

Adresa

Telefon

Edit Application Edit Page 3 Create Session Debug Show Edit Links

Figura II.11.20. Pagina formularului Echiipe



## Aplicații

Realizați un miniproiect folosind **Application Builder**.

Alegeți ca temă de proiect una dintre temele propuse la finalul părții de proiectare a bazelor de date și realizați aplicația pentru o parte a acestui proiect (3-4 tabele).

Pregătiți apoi o prezentare a proiectului pe care să o susțineți în fața colegilor de clasă.

Nu uitați să scrieți documentația aplicației, în care veți explica modul în care ați realizat proiectul, etapele, dar și modul de utilizare a aplicației create.

1. Interogări simple.  
Sortarea datelor
2. Funcții singulare
3. Interogări multiple
4. Gruparea datelor
5. Subinterogări
6. Crearea și modificarea  
structurii tabelor.  
Constrângeri
7. Introducerea și  
actualizarea datelor din  
tabele
8. Vederi (views)
9. Secvențe. Indecși.  
Sinonime
10. Acordarea și revocarea  
drepturilor. Gestiunea  
tranzacțiilor
11. Realizarea proiectelor
12. Aplicații recapitulative

În acest capitol vi se propun o serie de exerciții cu ajutorul cărora veți recapitula o mare parte din cunoștințele acumulate de-a lungul acestei părți a manualului.

<sup>1</sup>Se consideră tabela **bbc**, care memorează câteva date statistice despre țările lumii. Această tabelă conține următoarele coloane: **nume**, **regiune**, **suprafata**, **populație**, **pib** (pib=produsul intern brut și reprezintă valoarea tuturor bunurilor și serviciilor produse de o țară în cursul unui an).

1. Afișați țările care au populația cel puțin egală cu 200 milioane.
2. Afișați numele țărilor și produsul intern brut pe cap de locuitor pentru țările având o populație mai mare sau egală de 200 milioane.
3. Afișați numele și populația (în milioane locuitori) ale țărilor din Asia (Asia este regiunea).
4. Afișați numele și populația următoarelor țări 'Franța', 'Germania' și 'Italia'.
5. Afișați țările a căror nume conțin cuvântul 'Unite'.
6. Afișați numele țărilor a căror populație este mai mare decât populația din 'Rusia'.
7. Afișați numele și regiunea țărilor din regiunile în care se găsesc 'India' și 'Iran'.
8. Afișați țările din Europa al căror pib pe cap de locuitor este mai mare decât pib-ul pe cap de locuitor din 'Marea Britanie'.
9. Ce țări au o populație mai mică decât a Canadei dar mai mare decât a Algeriei?
10. Ce țări au pib-ul mai mare decât al oricărei țări europene?
11. Aflați cea mai mare țară din fiecare regiune (ca suprafață).
12. Unele țări au populația de trei ori mai mare decât cea a oricărei țări vecine (din aceeași regiune). Afișați numele acelor țări.
13. Afișați populația totală din întreaga lume.
14. Afișați numele tuturor regiunilor, o singură dată fiecare.
15. Afișați pib-ul total al tuturor țărilor din Africa.
16. Câte țări au o suprafață de cel puțin 1000000?
17. Care este populația totală din Franța, Germania și Spania?

---

<sup>1</sup> Aplicațiile 1-75 sunt preluate de pe site-urile <http://sqlzoo.net/> sau <http://db.grussell.org> sau folosesc idei preluate de pe aceste situri.

18. Pentru fiecare regiune, afișați numele regiunii și numărul de țări din regiunea respectivă.

19. Pentru fiecare regiune afișați numărul de țări cu cel puțin 10 milioane de locuitori.

20. Afișați regiunile a căror populație totală este cel puțin 100 milioane.

Se consideră următoarele trei tabele referitoare la câștigătorii medaliilor olimpice la tenis de-a lungul anilor:

- **medalii** – având coloanele **an**, **culoare**, **cine**, **tara**
- **tari** – având coloanele **id**, **nume**
- **locatii** – având coloanele **an**, **oras**, **tara**, reprezentând localitatea și codul țării în care s-au desfășurat jocurile olimpice în fiecare an.

Câmpul **tara** din tabela **medalii** este cheia străină a acestei tabele și face referire la coloana **id** din tabela **tari**.

21. Afișați numele tenismenilor care au câștigat medaliile din 2000 și țara din care provin aceștia.

22. Afișați ce sportivi din Suedia au câștigat medalii olimpice și ce fel de medalie.

23. Afișați anii în care China a câștigat medalia de aur.

24. Afișați cine a câștigat medaliile olimpice la jocurile desfășurate în Barcelona.

25. Afișați în ce oraș/orașe a câștigat medalii 'Jing Chen'. Afișați orașul și culoarea medaliei câștigate.

26. Afișați cine și în ce oraș a câștigat medalia de aur.

Se consideră o bază de date despre filme. Aceasta conține următoarele trei tabele:

- **movie** – cu coloanele **id**, **title**, **yr**, **score**, **votes**, **director**
- **actor** – cu coloanele **id**, **nume**
- **casting** – cu coloanele **movieid**, **actorid**, **ord**

27. Afișați filmele lansate în 1962. Afișați **id**-ul, și **title**

28. Afișați anul filmului 'Citizen Kane'.

29. Afișați toate filmele din seria Star Trek, incluzând **id**-ul, titlul și anul.



30. Ce titluri au filmele cu `id`-urile 1, 2, 3?
31. Ce `id` are filmul 'Casablanca'?
32. Afișați distribuția filmului 'Casablanca'.
33. Afișați numele filmelor în care a jucat 'Harrison Ford'.
34. Afișați numele filmelor în care a jucat 'Harrison Ford' dar nu în rolul principal (`ord>1`).
35. Afișați numele filmelor din 1962 împreună cu numele actorilor în rolul principal.
36. Care a fost anul în care 'John Travolta' a apărut în cele mai multe filme? Afișați anul și numărul de filme în care a jucat.
37. Listați titlurile filmelor apărute sub conducerea lui 'Julie Andrew' precum și numele actorilor principali ai acestor filme.
38. Afișați lista actorilor care au avut cel puțin 10 roluri principale.
39. Afișați numele filmele din 1978 în ordinea descrescătoare a numărului de actori care au jucat în acestea.
40. Afișați toți actorii care au lucrat sub îndrumarea lui 'Art Garfunkel'.

Datele despre membrii parlamentului din 2007 sunt memorate într-o bază de date conținând următoarele tabele:

- `parlamentari` – cu coloanele `nume`, `partid`, `circumscripție`
- `partide` – având coloanele `cod`, `nume`, `leader`.

Coloana `partid` din tabela `parlamentari` reprezintă codul partidului și este cheia străină a tabelii care face referire la coloana `cod` din tabela `partide`.

Majoritatea parlamentarilor aparțin unui partid, însă există și parlamentari independenți. Unele partide au un leader care este membru în parlament. Leader-ii de partide care nu sunt membrii în parlament, nu sunt memorați în baza de date, deci câmpul `leader` din tabela `partide` va avea în acest caz valoarea `null`.

41. Afișați parlamentarii independenți.
42. Afișați lista tuturor partidelor și a leader-ilor acestora.
43. Afișați partidele și leader-ii acestora, pentru partidele care au leader în parlament.
44. Afișați toate partidele care au cel puțin un membru în parlament.

45. Afișați lista ordonată alfabetic a tuturor parlamentarilor împreună cu numele partidului din care fac parte. Includeți în listă și parlamentarii independenți.

46. Afișați numele partidelor care au membrii în parlament. Afișați pentru fiecare partid și numărul de parlamentari.

47. Afișați numele tuturor partidelor împreună cu numărul de parlamentari ai partidului. Dacă un partid nu are niciun parlamentar se va afișa valoarea 0.

Se consideră următoarele tabele referitoare la cursele aeriene ale companiei **TAROM**:

**Zboruri** (zno, dela, la, distanta,  
oraDecolare, oraAterizare, pret)

**Aeronave** (ano, nume, autonomie)

**Certificari** (idAng, ano)

**Angajati** (idAng, nume, salariu)

**Observații:** Tabela **Angajați** memorează date despre piloți, dar și despre alți angajați. Fiecare pilot este certificat să zboare cu anumite aeronave, și doar piloții sunt certificați să zboare. Câmpul **autonomie** din tabela **aeronave** se referă la distanța maximă pe care o poate parcurge nava fără escală de realimentare.

48. Afișați numele tuturor navelor pe care este certificat să zboare fiecare pilot cu salariul mai mare de 880,000.

49. Pentru fiecare pilot care este certificat pentru mai mult de trei aeronave afișați id-ul, numele și prenume și valoarea maximă a autonomiilor de zbor a aeronavelor pentru care este certificat fiecare astfel de pilot.

50. Afișați numele piloților al căror salariu este mai mic decât prețul celui mai ieftin zbor de la Los Angeles la Honolulu.

51. Afișați numele tuturor aeronavelor cu autonomie de zbor mai mare de 1000 mile și pentru fiecare astfel de aeronavă afișați numele și media salariilor tuturor piloților certificați pentru acea aeronavă.

52. Afișați numele tuturor piloților certificați pe cel puțin o aeronavă de tip Boeing (numele aeronavei conține cuvântul Boeing).

53. Afișați codul tuturor aeronavelor (**ano**) care pot fi utilizate, fără escală, pe ruta de la Los Angeles la Chicago.

54. Afișați rutele ce pot fi pilotate de toți piloții cu salariul mai mare decât 10000.

55. Afișați numele tuturor piloților ce pot opera pe aeronave cu autonomie mai mare de 300 mile, dar care nu sunt certificați pe nicio aeronavă Boeing.

56. Un client dorește să zboare de la Madison la New York cu cel mult două zboruri. Afișați pentru toate variantele, ore plecării din Madison, ora sosirii la New York, știind că ora sosirii nu trebuie să fie mai târziu de 6 p.m.

57. Calculați diferența dintre salariul mediu al tuturor piloților și salariul mediu al tuturor angajaților (inclusiv piloții).

58. Afișați numele și salariul fiecărui angajat care nu este pilot și care are salariul mai mare decât salariul mediu al tuturor piloților.

59. Afișați numele piloților care sunt certificați doar pentru aeronave cu autonomie mai mare de 1000 mile.

60. Afișați numele piloților care sunt certificați doar pentru aeronave cu autonomie mai mare de 1000 mile, dar sunt certificați pentru cel puțin două astfel de aeronave.

61. Afișați numele piloților care sunt certificați doar pentru aeronave cu autonomie mai mare de 1000 mile și sunt certificați pe cel puțin un aparat de tip Boeing.

Se consideră următoarele tabele:

**Angajați** (idAng, nume, datan, salariu)

**Angajari** (idAng, idDep, pct\_timp)

**Departamente** (idDep, nume, buget, idManager, etaj)

**Observații:** Un angajat poate lucra în mai mult de un departament, câmpul **pct\_timp** din tabela **angajari** memorând procentul de timp lucrat de un angajat la un anumit departament.

62. Afișați numele și vârsta (număr întreg de ani împliniți la data curentă) fiecărui angajat care lucrează atât în departamentul Hardware cât și în departamentul Software.

63. Pentru fiecare departament cu mai mult de 20 de norme (de exemplu doi angajați cu procent de 50% în acest departament ocupă împreună o normă), afișați id-ul departamentului, și numărul de angajați care lucrează pentru acel departament.

64. Afișați numele fiecărui angajat care are salariul mai mare decât bugetul tuturor departamentelor în care lucrează.

65. Afișați numele managerilor care conduc departamentele cu buget mai mare de 1 milion.

66. Afișați numele angajaților care conduc departamentele cu cel mai mare buget.

67. Dacă un angajat conduce mai mult de un departament, el gestionează suma tuturor bugetelor departamentelor conduse de el. Afișați numele angajaților care gestionează mai mult de 5 milioane.

68. Afișați numele managerilor cu cele mai mari trei bugete gestionate.

69. Afișați numele managerilor care conduc doar departamente cu bugete mai mari de 1 milion, dar cel puțin unul dintre departamentele conduse are buget mai mare de 5 milioane.

70. Afișați numele angajaților care lucrează la cel puțin un departament de la etajul 10.

71. Afișați numele angajaților care conduc mai mult de 3 departamente aflate la același etaj.

72. Afișați numele tuturor angajaților care lucrează în departamentul cu cel mai mare buget dintre toate departamentele la care lucrează Ionescu Ioan.

73. Afișați numele tuturor angajaților care lucrează la același etaj la care se găsește cel puțin unul dintre departamentele la care lucrează Ionescu Ioan.

74. Afișați numele angajaților al căror salariu este mai mare decât salariile tuturor managerilor din departamentele în care lucrează.

75. Afișați numele angajaților care lucrează într-un singur departament și care nu au nici cel mai mare și nici cel mai mic salariu din acel departament.

76. Să se creeze o tabelă **matr** conținând următoarele coloane: **val**, **lin**, **col**, fiecare linie a acestei tabele memorând câte un element nenul al unei matrice pătratice. Liniile și coloanele se vor numerota începând cu 1. Dimensiunea matricei se va memora într-o linie a tablei în care coloanele **lin** și **col** vor rămâne necompletate (valoarea **null**). De exemplu, matricei:

$$\begin{pmatrix} 1 & 2 & 0 \\ 4 & 5 & 6 \\ 7 & 0 & 0 \end{pmatrix}$$

îi va corespunde următorul conținut al tablei **matr**:

val	lin	col
1	1	1
2	1	2
4	2	1
5	2	2
6	2	3
7	3	1
3	-	-

77. Inserați în tabela **matr**, toate elementele nenule ale unei matrice cu 5 linii și 5 coloane.

78. Considerând că cele două diagonale ale matricei împart matricea în patru zone: nord, sud, est și vest, diagonalele nefăcând parte din nici una dintre aceste zone, se cere să se afișeze:

- a) suma elementelor din nordul matricei;
- b) suma elementelor pare din sudul matricei;
- c) numărul de elemente pozitive din estul matricei;
- d) valoarea maximă a elementelor din vestul matricei;
- e) suma elementelor de pe diagonala principală a matricei;
- f) suma elementelor de pe diagonala secundară a matricei.

**79.** Afișați elementele nenule din matrice care sunt egale cu elementele simetrice față de diagonala secundară. De exemplu pentru matricea

$$\begin{pmatrix} 5 & 2 & 0 \\ 4 & 5 & 2 \\ 7 & 0 & 5 \end{pmatrix}$$

se va afișa:

VAL	LIN	COL
5	1	1
2	2	3
5	3	3
2	1	2

**80.** Afișați suma elementelor de pe fiecare linie a matricei.

**81.** Să se afișeze indicii liniilor care conțin cel puțin 2 elemente nule. De exemplu pentru matricea

$$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 3 & 4 & 0 & 2 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 3 & 0 \end{pmatrix}$$

se va afișa

LIN
1
2
3

**82.** Afișați suma elementelor aflate pe pătratele concentrice ale matricei. De exemplu pentru matricea

$$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 3 & 4 & 0 & 2 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 3 & 0 \end{pmatrix}$$

se va afișa

LIN	SUMA
1	18
2	7

**83.** Să se ștergă linia a doua și coloana a doua a matricei. **Atenție!** nu trebuie doar să ștergeți unele articole din tabelă, ci va trebui să faceți anumite modificări asupra unora dintre înregistrările rămase în tabelă.

**84.** Ștergeți conținutul tabelului `matr`, și reluați exercițiile **76-83**, pentru o matrice de 7 linii și 7 coloane.

În tabela **persoane** se memorează arborii genealogici ai mai multor familii. Fiecare linie a tabelului va memora următoarele informații despre o persoană: **id**, **nume**, **prenume**, **datan**, **idtata**, **idmama**. Atenție! În câmpurile **idtata** și **idmama** se vor memora id-urile părinților naturali, nu ne interesează cine sunt eventualii părinți adoptivi.

**85.** Afișați numele și prenumele tuturor fraților mai mici ai lui **'Ionescu Ioan'**, născut pe **25 septembrie 1950**.

**86.** Afișați numele și prenumele tuturor fraților mai mari ai lui **'Ionescu Ioan'** născut pe **25 septembrie 1950**.

**87.** Afișați numele și prenumele bunicilor de sex masculin ai lui **'Ionescu Ioan'** născut pe **25 septembrie 1950**.

**88.** Afișați numele și prenumele unchilor și mătușilor lui **'Ionescu Ioan'** născut pe **25 septembrie 1950**.

**89.** Câți nepoți de la copii are **'Ionescu Ioan'** născut pe **25 septembrie 1950**?

Se consideră tabelele:

**boli** – conținând coloanele **codb** (codul bolii) și **nume**

**medicamente** – conținând coloanele **codm** (codul medicamentului), **nume**

**indicatii** – cu coloanele **codb** și **codm** memorând toate medicamentele indicate în anumite afecțiuni

**contraindicatii** – cu coloanele **codb** și **codm** memorând toate medicamentele contraindicate în anumite afecțiuni

**90.** Afișați medicamentele al căror nume se termină în literele **'ol'**.

**91.** Afișați toate medicamentele indicate în cazul ulcerului.

**92.** Afișați toate medicamentele contraindicate în cazul ulcerului.

**93.** Afișați medicamentele care sunt indicate în cazul ulcerului dar care nu sunt contraindicate în cazul hipertensiunii arteriale.

**94.** Afișați numele bolilor pentru care există mai puțin de 3 medicamente indicate și mai mult de 5 medicamente contraindicate.

**95.** Afișați medicamentele care sunt indicate în mai mult de 5 afecțiuni și sunt contraindicate în mai puțin de 3 afecțiuni.

## Bareme de corectare și notare

### PARTEA I.1.

**Barem de corectare și notare pentru testul de autoevaluare**

**Răspunsurile corecte sunt:** 1-b, 2-c, 3-b, 4-d, 5-b, 6-c

6 x 1.5 = **9 p**

**Oficiu 1 p**

**Total 10 p**

**Barem de notare (pentru ambele teste de evaluare):**

1. 3 x 0,2 = **0,6 p**

2. 8 relații x 0,30 p = **2,4 p** (0,15 puncte pentru cardinalitatea relației și 0,15 puncte pentru opționalitate)

3. 2 x 2 p = **4 p** (pentru fiecare relație se acordă 4x0.15p=**0,60 p** pentru opționalitatea relațiilor, 4x0.15p=**0.60 p** pentru cardinalitatea relațiilor, **0,3 p** pentru alegerea entității de intersecție, **0,3 p** pentru attributele entității de intersecție, **0,2 p** pentru UID)

4. 2 x 0.5 = **1 p**

**Oficiu 2 p**

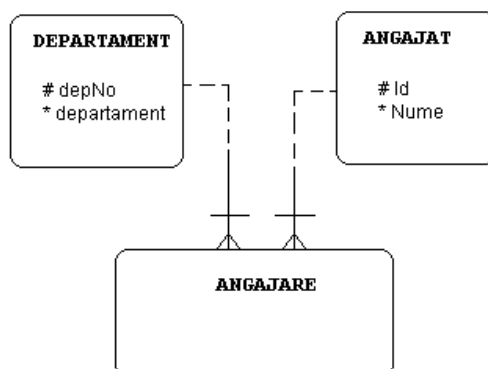
**Total 10 p**

### PARTEA I.3.

**Barem de corectare și notare pentru testul de autoevaluare**

**Răspunsurile corecte sunt:** 1-c, 2-b, 3-b, 4-a

**Răspunsul corect pentru punctul 5:**



Se acordă: **1.5 p x 4** (pentru întrebările 1, 2, 3, 4)

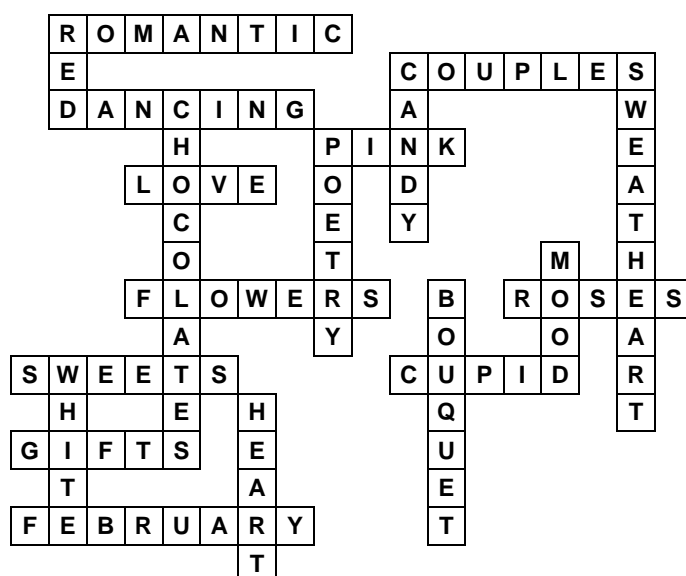
**2 p** (pentru întrebarea 5)

**Oficiu 2 p**

**Total 10 p**

### **PARTEA II.1.**

Rezolvarea jocului de cuvinte (rebusului) de la pag. 131:



### **PARTEA II.3.**

#### **Barem de corectare și notare pentru testul de evaluare**

Se acordă 1 punct pentru fiecare răspuns corect, 1 punct se acordă din oficiu.

Răspunsurile corecte sunt:

1-a, 2-c, 3-a, 4-b, 5-d,

6-d, 7-c, 8-b, 9-a.

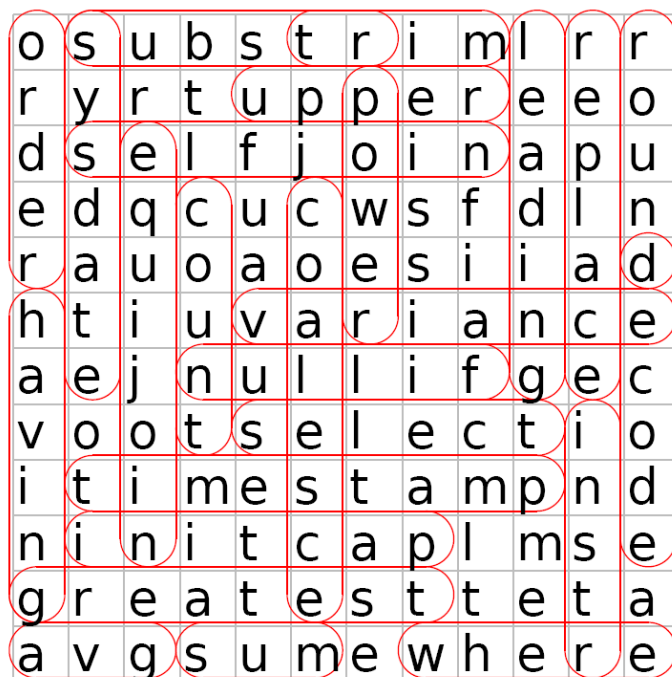


#### **PARTEA II.4.**

Rezolvările celor două jocuri. Cuvintele pe care le puteți găsi în grilă sunt:

AVG	COALESCE	COUNT
DECODE	EQUIJOIN	GREATEST
HAVING	INITCAP	INSTR
LEADING	NULLIF	ORDER
POWER	REPLACE	ROUND
SELECT	SELFJOIN	SUBSTR
SUM	SYSDATE	TIMESTAMP
TRIM	UPPER	VARIANCE
WHERE		

Iată și cum sunt ele poziționate pe grilă:



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1					G	R	O	U	P	B	Y								H	
2		L				P													A	
3		P				A												A	V	G
4	L	A	S	T	_	D	A	Y				M							I	
5		D		O					A	L	I	A	S			C	O	U	N	T
6							S			O	X				O				G	
7		S	I	N	G	L	E	R	O	W				M	I	N				T
8		Y		U			L			E				O		C				O
9		S	U	M			F			R	O	U	N	D		A				_
10		D		B												T	R	U	N	C
11		A		E	Q	U	I	J	O	I	N									H
12		T		R							E									A
13		E			T				L	X					S	U	B	S	T	R
14					A	R	G	U	M	E	N	T				P			O	
15					I					N						P				
16					M					G	D					D	E	C	O	D
17							I	N	I	T	C	A	P			R				A
18							V		H		Y									T
19							L													E

## **PARTEA II.5**

### **Barem de corectare și notare pentru testul de evaluare**

1-e, 2-a, 3-c, 4-b, 5-d, 6-b, 7-a

Se acordă 2 puncte pentru întrebările 2 și 4 și 1 punct pentru celelalte întrebări. Se acordă 1 punct din oficiu.

## **PARTEA II.6**

### **Barem de corectare și notare pentru testul de evaluare**

Se acordă 1 punct pentru fiecare răspuns corect, 2 puncte se acordă din oficiu. Răspunsurile corecte sunt:

1-a, 2-c, 3-d, 4-e, 5-b,d,e,f,

6-g, 7-c, 8-a.

## **PARTEA II.9.**

### **Barem de corectare și notare pentru testul de evaluare**

1-e, 2-d, 3-c, 4-a, 5-b, 6-b, 7-c, 8-a, 9-c

Se acordă câte 1 punct pentru fiecare răspuns corect și 1 punct din oficiu.