Lecţia 2 – Design-ul bazei de date

2.1 Proiectarea bazei de date relaţionale

Proiectarea bazelor de date se referă la fixarea structurii bazei de date şi a metodelor de prelucrarea datelor, spre deosebire de utilizarea bazei de date, care se referă la informaţiile stocate în baza de date.

Dacă o bază de date îşi schimbă frecvent conţinutul, structura ei rămâne neschimbată pentru o perioadă lungă de timp.

Proiectarea unei baze de date urmăreşte obţinerea următoarelor calităţi:

- Corectitudine – reprezentarea cât mai fidelă în baza de date a modului obişnuit de lucru cu datele în sistemul real;
- Consistenţă – informaţiile corespunzătoare obiectelor cu care se lucrează în baza de date (nume, definire, relaţii) să nu conţină contradicţii;
- Distribuire – informaţiile să poată fi utilizate de aplicaţii multiple şi să poată fi accesate de mai mulţi utilizatori, aflaţi în diferite locuri, utilizând medii diverse
- Flexibilitate – facilităţi de adăugare de componente care să reflecte cereri noi de informaţii, să îmbunătăţească performanţele sau să adapteze datele pentru eventuale modificări.

Un model relaţional de baze de date cuprinde trei componente principale:

- Structura datelor prin definirea unor domenii şi a relaţiilor (atribute, tupluri, chei primare);
- Integritatea datelor prin impunerea unor restricţii;
- Prelucrarea datelor prin operaţii din algebra relaţională sau calculul relaţional.

Modelul relaţional se bazează pe noţiunea matematică de relaţie aşa cum este ea definită în teoria mulţimilor, şi anume ca o submulţime a produsului cartezian a unei liste finite de mulţimi numite domenii.

Algebra relaţională constă dintr-o colecţie de operatori ce au ca operanzi relaţii. Rezultatul aplicării unui operator la una sau două relaţii este tot o relaţie.

Noţiunile de model relaţional şi algebră relaţională au fost discutate în primul capitol al acestui curs.

Proiectarea structurii bazei de date relaţionale (**BDR**) se face pe baza modelelor realizate în activitatea de analiză. Înainte de proiectarea bazei de date se alege tipul de sistem de gestiune a bazei de date (**SGBD**). Alegerea **SGBD**-ului se face ţinând cont de două aspecte: cerinţele aplicaţiei şi performanţele tehnice ale **SGBD**-ului.

Cerinţele aplicaţiei se referă la: volumul de date estimat a fi memorat şi prelucrat în **BDR**; complexitatea problemei de rezolvat; ponderea şi frecvenţa operaţiilor de intrare/ieşire; condiţiile

privind protecţia datelor; operaţiile necesare (încărcare/validare, actualizare, regăsire etc.); particularităţile activităţii pentru care se realizează baza de date.

Performanţele tehnice ale **SGBD**-ului se referă la: modelul de date pe care-l implementează; ponderea utilizării **SGBD**-ului pe piaţă şi tendinţa; configuraţia de calcul minimă cerută; limbajele de programare din **SGBD**; facilităţile de utilizare oferite pentru diferite categorii de utilizatori; limitele **SGBD**-ului; optimizările realizate de **SGBD**; facilităţile tehnice; lucrul cu mediul distribuit şi concurenţa de date; elementele multimedia; posibilitatea de autodocumentare; instrumentele specifice oferite.

Proiectarea **BDR** se realizează prin proiectarea schemelor **BDR** şi proiectarea modulelor funcţionale specializate.

Schemele bazei de date sunt: *conceptuală*, *externă* şi *internă*.

a) *Proiectarea schemei conceptuale* porneşte de la identificarea setului de date necesar sistemului. Aceste date sunt apoi integrate şi structurate într-o schemă a bazei de date. Pentru acest lucru se parcurg paşii:

- Stabilirea schemei conceptuale iniţiale rezultă din schema entitate-relaţii **ERD**. Pentru acest lucru, fiecare entitate din modelul conceptual este transformată (mapată) într-o colecţie de date (tabel memorat în fişier), iar pentru fiecare relaţie se definesc cheile aferente.
- Ameliorarea progresivă a schemei conceptuale prin eventuale adăugări de tabele suport suplimentare, prin eliminarea unor anomalii

b) *Proiectare schemei externe* are rolul de a specifica vederile fiecărui utilizator asupra **BDR**. Pentru acest lucru, din schema conceptuală se identifică datele necesare fiecărei vederi. Datele obţinute se structurează logic în subscheme ţinând cont de facilităţile de utilizare şi de cerinţele utilizator. Schema externă devine operaţională prin definirea unor vederi (view-uri) în **SGBD**-ul ales şi acordarea drepturilor de acces. Datele dintr-o vedere pot proveni din una sau mai multe colecţii şi nu ocupă spaţiul fizic.

c) *Proiectarea schemei interne* presupune stabilirea structurilor de memorare fizică a datelor şi definirea căilor de acces la date. Acestea sunt specifice fie **SGBD**-ului (scheme de alocare), fie sistemului de operare. Proiectarea schemei interne înseamnă estimarea spaţiului fizic pentru **BDR**, definirea unui model fizic de alocare (a se vedea dacă **SGBD**-ul permite explicit acest lucru) şi definirea unor indecşi pentru accesul direct, după cheie, la date.

Proiectarea modulelor funcţionale ţine cont de concepţia generală a **BDR**, precum şi de schemele proiectate anterior. În acest sens, se proiectează fluxul informaţional, modulele de încărcare şi manipulare a datelor, interfeţele specializate, integrarea elementelor proiectate cu organizarea şi funcţionarea **BDR.**

2.2 Realizarea componentelor logice

Componentele logice ale unei baze de date (**BD**) sunt programele de aplicaţie dezvoltate, în cea mai mare parte, în **SGBD**-ul ales. Programele se realizează conform modulelor funcţionale proiectate în etapa anterioară. Componentele logice ţin cont de ieşiri, intrări, prelucrări şi colecţiile de date. În paralel cu dezvoltarea programelor de aplicaţii se întocmesc şi documentaţiile diferite (tehnică, de exploatare, de prezentare).

2.3 Punerea în funcţiune şi exploatarea

Se testează funcţiile **BDR** mai întâi cu date de test, apoi cu date reale. Se încarcă datele în **BDR** şi se efectuează procedurile de manipulare, de către beneficiar cu asistenţa proiectantului. Se definitivează documentaţiile aplicaţiei. Se intră în exploatare curentă de către beneficiar conform documentaţiei.

2.4 Întreţinerea şi dezvoltarea sistemului

Ulterior punerii în exploatare a **BDR**, în mod continuu, pot exista factori perturbatori care generează schimbări în **BDR**. Factorii pot fi: organizatorici, datoraţi progresului tehnic, rezultaţi din cerinţele noi ale beneficiarului, din schimbarea metodologiilor etc.

2.5 Relaţii. Tipuri de relaţii între tabele

În bazele de date relaţionale una dintre cele mai importante noţiuni este cea de relaţie. Practic, tabelele unei baze de date sunt relaţionate între ele. Într-o bază de date relaţională nu este indicat să avem tabele izolate.

Există trei tipuri de relaţii posibile între tabelele unei baze de date:

- *unu-la-unu* sau *one-to-one* (*1:1*) – unei înregistrări din prima tabelă îi corespunde o singură înregistrare în cealaltă tabelă;

- *unu-la-mai-mulţi* sau *one-to-many* (*1:n*) – unei înregistrări din prima tabelă îi corespund mai multe înregistrări în cealaltă tabelă;

- *mai-mulţi-la-mai-mulţi* sau *many-to-many* (*m:n*) – unei înregistrări din prima tabelă îi corespund una sau mai multe înregistrări din cealaltă tabelă şi reciproc.

Primul caz, relaţia *one-to-one* este mai puţin utilizată în cazuri concrete. Un exemplu ar fi o tabelă în care avem persoane şi o tabelă cu acte de identitate (o persoană are un singut act de identitate sau o persoană are o singură adresă de domiciliu).

Relaţia *one-to-many* este foarte răspândită (de exemplu, dacă avem o tabelă de clienţi şi una de facturi – un client poate să aibă mai multe facturi sau dacă avem o tabelă cu elevi şi una cu note atunci un elev poate să aibă mai multe note).

Relaţia *many-to-many* este o relaţie în care avem nevoie de o tabelă intermediară de legătură între cele două tabele (practic relaţia *many-to-many* se descompune în două relaţii *one-to-many*).

Un exemplu ar putea fi următorul: dacă într-o tabelă avem informaţii despre studenţii unei facultăţi iar într-o altă tabelă informaţii despre materiile (cursurile) disponibile în cadrul acelei facultăţi avem următorul tip de relaţie între aceste 2 tabele: un student participă la mai multe cursuri iar un curs este frecventat de mai mulţi studenţi, rezultă că avem de-a face cu o relaţie *many-to-many* între cele 2 tabele.

Această relaţie se descompune în 2 relaţii *one-to-many* prin introducerea unei tabele suplimentare care păstrează informaţii de identificare pentru studenţi şi pentru cursurile pe care ei le frecventează.

Înţelegerea modului de relaţionare al tabelelor dintr-o bază de date reprezintă un pas fundamental pentru a putea construi o bază de date optimă atunci când proiectăm o aplicaţie.

2.6 Exemplu

Prezentăm în continuare diagrama **ERD** corespunzătoare unei baze de date în care avem stocate informaţii despre candidaţii la un examen.

Sunt reprezentate în această diagramă entităţile, atributele fiecărei entităţi, precum şi relaţiile existente între aceste entităţi.

Urmează apoi o prezentare detaliată a entităţilor modelate în această diagramă.

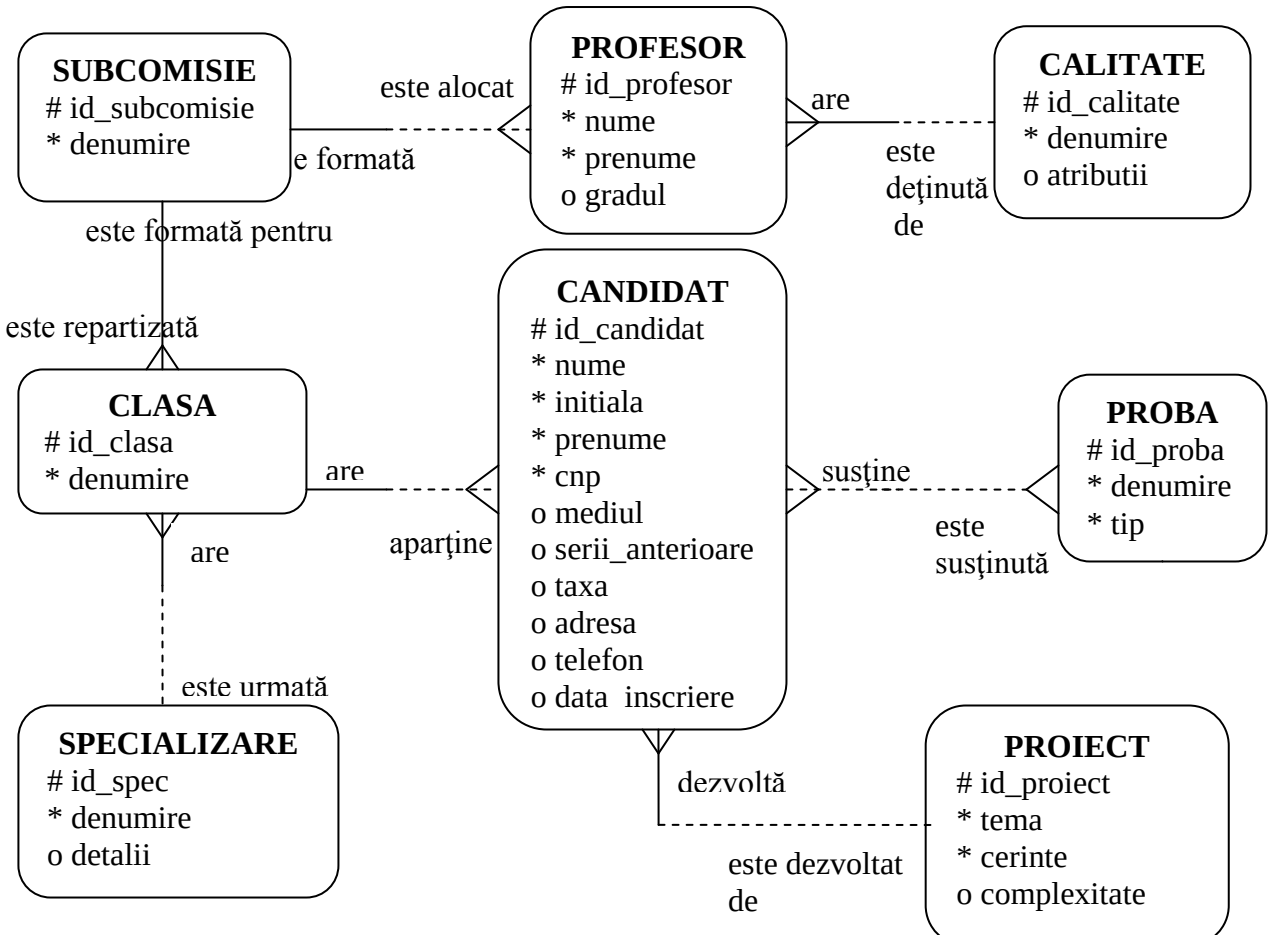


Figura 2.1. Diagrama ERD iniţială

Tabelele de mai jos prezintă detaliat entităţile modelate în diagramă:

Entitatea: **CALITATE**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_calitate	Numeric	Da	Da	ID-ul ce este asociat calităţii
denumire	Şir de caractere	Da	-	Denumirea calităţii deţinută de profesor în comisie: evaluator, secretar, preşedinte, etc
atributii	Şir de caractere	-	-	Atribuţiile ce decurg din calitatea deţinută: ex: evaluarea elevilor, organizarea examenului etc.

Entitatea: **PROFESOR**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_profesor	Numeric	Da	Da	Id-ul asociat profesorului (ex. 100)
nume	Şir de caractere	Da	-	Numele profesorului
prenume	Şir de caractere	Da	-	Prenumele profesorului
gradul	Şir de caractere	-		Gradul didactic al profesorului: definitiv, gradul II, gradul I

Entitatea: **SUBCOMISIE**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_subcomisie	Numeric	Da	Da	Id-ul asociat comisiei (ex. 100)
denumire	Şir de caractere	Da	-	Denumirea comisiei (ex. Comisia 1, Comisia 2 etc.)

Entitatea: **CLASA**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_clasa	Numeric	Da	Da	Id-ul asociat clasei (ex. 100)
denumire	Şir de caractere	Da	-	Denumirea clasei (ex. XII A, XII B etc.)

Entitatea: **SPECIALIZARE**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_specializare	Numeric	Da	Da	Id-ul asociat specializării (ex. 100)
denumire	Şir de caractere	Da	-	Denumirea specializării (ex. Matematică informatică etc.)
detalii	Şir de caractere	-	-	Detaliile specializării

Entitatea: **CANDIDAT**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_candidat	Numeric	Da	Da	Id-ul asociat candidatului (ex. 100)
nume	Şir de caractere	Da	-	Numele candidatului
initiala	Şir de caractere	Da	-	Iniţiala prenumelui tatălui
prenume	Şir de caractere	Da	-	Prenumele candidatului
cnp	Şir de caractere	Da	-	Codul numeric personal
mediul	Numeric	-	-	Mediul de proveniență al candidatului; se foloseşte codificarea numerică 1 – mediul urban, 2 – mediul rural
serii_anterioare	Numeric	-	-	Candidaţii din seria curentă (nu au absolvit cls. XII) nu au completată valoarea atributului, cei din seriile anterioare au valoarea atributului 1
taxa	Numeric	-	-	Candidaţii ce nu au mai susţinut examenul nu au completată valoarea atributului, cei din învăţământul particular şi cei ce au mai susţinut examenul de minimum 2 ori au valoarea taxei
adresa	Şir de caractere	-	-	Adresa candidatului
telefon	Şir de caractere	-	-	Telefonul candidatului
data_inscriere	Dată calendaristică	-	-	Data depunerii cererii de înscriere la examenul de atestat

Entitatea: **PROIECT**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_proiect	Numeric	Da	Da	Id-ul asociat proiectului (ex. 100)
tema	Şir de caractere	Da	-	Denumirea temei (ex. Agenţie de voiaj)
cerinte	Şir de caractere	Da	-	Cerinţele detaliate ale proiectului
complexitate	Şir de caractere	-	-	Complexitatea proiectului realizat : redusă, medie, etc

Entitatea: **PROBA**				
Atribut	*Tipul de date folosit*	*Atribut obligatoriu*	*Identificator unic*	*Semnificaţia*
id_proba	Numeric	Da	Da	Id-ul asociat probei (ex. 1)
denumire	Şir de caractere	Da	-	Denumirea probei (ex. Programare, Sisteme de gestiune a bazelor de date, etc)
tip	Şir de caractere	Da		Tipul probei: probă practică sau proiect

2.7 Relaţii între entităţi

Diagrama **ERD** evidenţiază şi relaţiile existente între entităţile modelate. O relaţie între două entităţi arată că există o dependenţă, o legătură naturală între conceptele reprezentate de aceste entităţi. Este evidentă relaţia dintre entităţile CANDIDAT şi CLASĂ : un candidat este un elev ce este înregistrat într-o anumită clasă, iar o clasă este formată din mai mulţi elevi.

Relaţia dintre entitatea A şi entitatea B se defineşte prin:

- denumirea relaţiei: un verb ce sugerează dependenţa dintre cele două entităţi
- opţionalitatea relaţiei: este necesar să stabilim dacă *trebuie* sau *poate* să existe corespondenţă între cele două entităţi
- cardinalitatea relaţiei: precizează numărul de instanţe ale entităţii B ce sunt puse în corespondenţă cu o instanţă a entităţii A.

Relaţia dintre două entităţi este bidirecţională, dar nu simetrică: dacă există o relaţie între A şi B, există şi o relaţie între B şi A, dar nu aceeaşi.

2.8 Convenţii de reprezentare a relaţiilor

1. Linia ce uneşte entităţile relaţionate e formată din două segmente distincte. Tipul liniei ce pleacă de la entitatea A către entitatea B relevă opţionalitatea relaţiei A→B: dacă linia este continuă, relaţia este obligatorie – „trebuie”, iar dacă este discontinuă, relaţia este opţională – „poate”.
2. Denumirea relaţiei A→B este poziţionată lângă entitatea A, deasupra sau dedesubtul liniei de opţionalitate.
3. Cardinalitatea relaţiei A→B se reprezintă astfel: linia de A la B se termină cu o linie simplă, în cazul în care o instanţă a entităţii A este pusă în corespondenţă cu o singură instanţă a entităţii B, şi are forma unui „picior de cioară” ⋞ în cazul în care o instanţă a entităţii A este pusă în corespondenţă cu mai multe instanţe ale entităţii B.

Exemplu:

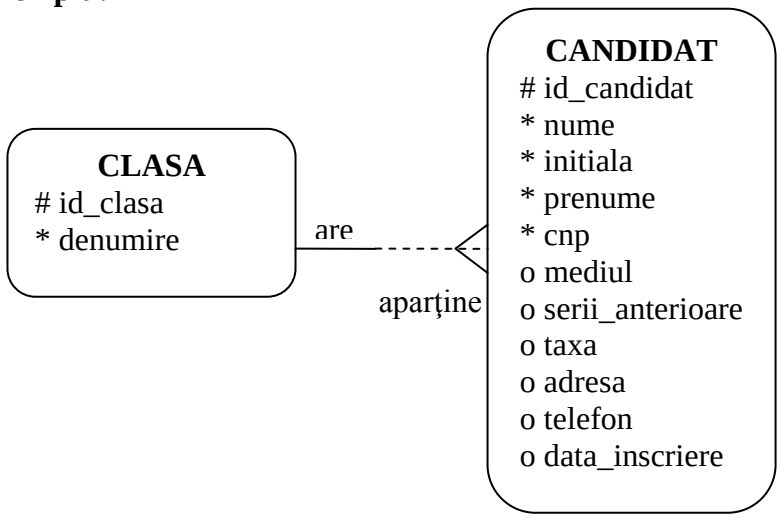


Figura 2.2 Relaţia dintre entităţile CLASA şi CANDIDAT

Relaţia CLASA→ CANDIDAT

- denumirea: **are**
- opţionalitatea: **trebuie** (segmentul ce pleacă dinspre CLASA este continuu)
- cardinalitatea: **una sau mai mulţi** (linia relaţiei se termină cu „picior de cioară”)

Relaţia CLASA→ CANDIDAT se citeşte: „O CLASA trebuie *să aibă* unul sau mai mulţi **CANDIDATI**”.

Relaţia CANDIDAT→ CLASA

- denumirea: **aparţine**
- opţionalitatea: **poate** (e posibil să existe candidaţi din seriile anterioare care să nu mai aparţină vreunei clase)
- cardinalitatea: **una şi numai una**

Relaţia CANDIDAT →CLASA se citeşte: „Un CANDIDAT poate *aparţine* unei singure **CLASE**".

O clasificare a relaţiilor dintre entităţile A şi B foloseşte cardinalitatea relaţiilor A →B şi B →A.

Cardinalitate A →B	*Cardinalitatea B →A*	*Tipul relaţiei*
una şi numai una	una şi numai una	„one to one" (1:1)
una şi numai una	una sau mai multe	„one to many" (1:M)
una sau mai multe	una şi numai una	„one to many" (1:M)
una sau mai multe	una sau mai multe	„many to many" (M:M)

Relaţia dintre entităţile CLASA şi CANDIDAT este de tipul „one-to-many".

Entităţile relaţionate	*Citirea relaţiei*	*Tipul relaţiei*
PROFESOR, CALITATE	Un PROFESOR trebuie să aibă o singură CALITATE. O CALITATE poate fi deţinută de unul sau mai mulţi PROFESORI.	One to many
SUBCOMISIE, PROFESOR	O SUBCOMISIE trebuie să fie formată din unul sau mai mulţi PROFESORI. Un PROFESOR poate fi alocat unei singure SUBCOMISII	One to many
SUBCOMISIE, CLASA	O SUBCOMISIE trebuie să fie formată pentru una sau mai multe CLASE. O CLASA trebuie să fie repartizată unei singure SUBCOMISII.	One to many
CLASA, CANDIDAT	O CLASA trebuie să aibă unul sau mai mulţi CANDIDATI. Un CANDIDAT poate aparţine unei singure CLASE.	One to many
CLASA, SPECIALIZARE	O CLASA trebuie să aibă o singură SPECIALIZARE. O SPECIALIZARE poate fi urmată de una sau mai multe CLASE.	One to many
CANDIDAT, PROIECT	Un CANDIDAT trebuie să dezvolte un singur PROIECT. Un PROIECT poate fi dezvoltat de unul sau mai mulţi CANDIDAŢI.	One to many
CANDIDAT, PROBA	Un CANDIDAT poate susţine una sau mai multe	Many to many

	PROBE. O PROBĂ poate fi susţinută de unul sau mai mulţi CANDIDAŢI.	

Se observă relaţia de tip *many-to-many* dintre entităţile CANDIDAT şi PROBA. Relaţiile de acest tip nu pot fi implementate în practică în niciun sistem de gestiune a bazelor de date.

O relaţie *many to many* dintre entitatea A şi entitatea B este descompusă prin adăugarea unei noi entităţi C denumită *entitate de intersecţie* şi construirea de relaţii *one to many* între noua entitate C şi entităţile iniţiale A şi B.

Entităţii de intersecţie i se poate atribui o denumire naturală, ce are o semnificaţie concretă în legătură cu entităţile iniţiale, sau, în lipsa acesteia, o denumire artificială care să combine denumirile entităţilor iniţiale.

În cazul nostru, entitatea de intersecţie este denumită NOTA, deoarece legătura între un candidat şi o probă a examenului este nota obţinută de acesta la acea probă. O denumire artificială ce putea fi folosită este PROBA_CANDIDAT, însă putem recunoaşte că nu este cea mai fericită alegere.

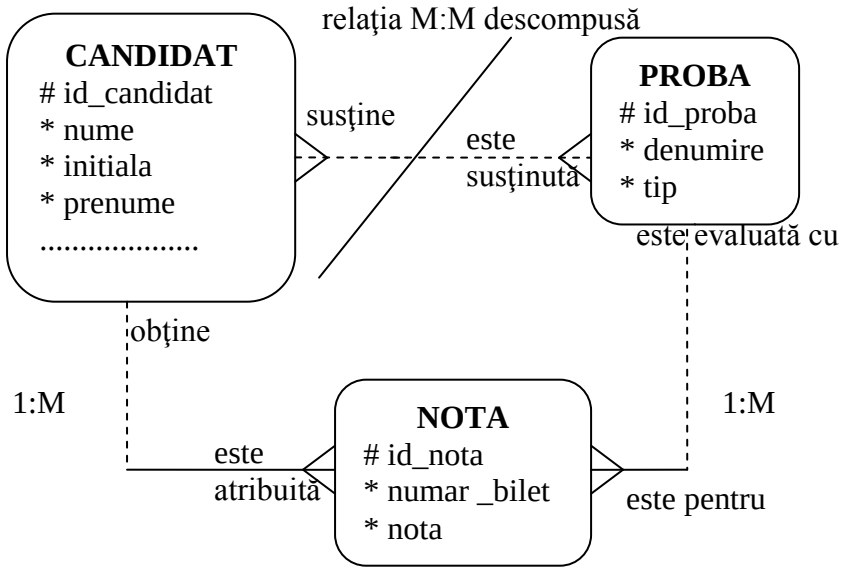


Figura 2.3. Descompunerea relaţiilor „many to many”

Se observă că se păstrează vechile opţionalităţi în partea dinspre entităţile originale, iar relaţiile care pleacă din entitatea de intersecţie sunt întotdeauna obligatorii în această parte.

Noile relaţii sunt de tip *one-to-many*, partea cu many - „piciorul de cioară” fiind întotdeauna înspre entitatea de intersecţie.

Se pune problema alegerii identificatorului unic (**UID**) pentru entitatea de intersecţie. Există două posibilităţi:

Să se creeze un nou **identificator unic artificial**, de ex: *id_nota*, atribut numeric Valorile acestui identificator unic nu au o semnificaţie concretă pentru o instanţă a acestei entităţi (o notă), singura preocupare fiind asigurarea unicităţii (evitarea duplicatelor). Această metodă este preferată deseori pentru simplitatea implementării, în condiţiile utilizării secvenţelor de numere generate automat (*sequences*).

Să se construiască un **identificator unic compus** care să includă identificatorii unici ai entităţilor relaţionate, la care eventual să se adauge atribute proprii entităţii de intersecţie. În acest caz relaţiile cu entităţile de la care s-au utilizat identificatori unici în construcţia UID-ului entităţii de intersecţie trebuie barate către entitatea de intersecţie, iar atributul propriu ce a fost inclus în **UID**-ul compus trebuie precedat de caracterul „#”.

Se observă că **UID**-ul entităţii NOTA este unul compus, format din **UID**-ul *id_candidat* preluat de la entitatea CANDIDAT, **UID**-ul *id_proba* preluat de la entitatea PROBA şi atributul propriu *numar_bilet.* Includerea atributului propriu *numar_bilet* în **UID** ar fi necesară doar în eventualitatea în care un candidat ar putea schimba biletul cu subiecte primit la o anumită probă şi se doreşte înregistrarea acestei situaţii, altfel atributele împrumutate în **UID** de la entităţile relaţionate ar fi suficiente pentru unicitate.

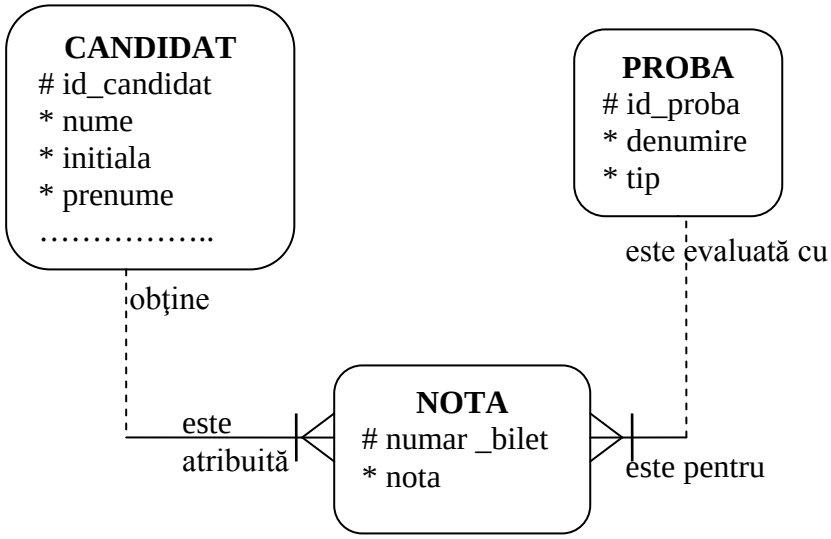


Figura 2.4. Definirea UID-ului unic compus al entităţii NOTA

Diagrama **ERD** în forma actuală permite înregistrarea unei note sau a mai multor note obţinute de un candidat la o anumită probă, însă implică discuţii: dacă se înregistrează o singură notă obţinută de un candidat la o anumită probă, atunci această notă trebuie să fie media aritmetică a notelor acordate de cei doi profesori evaluatori. Atunci nu am ştii ce notă a acordat fiecare profesor unui candidat la proba respectivă.

Rezultă necesitatea de a relaţiona entitatea NOTA cu entitatea PROFESOR, pentru a întregii informaţiile privitoare la notă. Vom cunoaşte cu exactitate cui i s-a acordat nota (prin relaţia cu entitatea CANDIDAT), la ce probă (prin relaţia cu entitatea PROBA) şi de către cine (prin relaţia cu entitatea PROFESOR).

Adăugarea acestei relaţii creează în diagramă o buclă ce creează posibilitatea apariţiei unor relaţii ciclice, *redundante*, situaţie ce trebuie evitată. O relaţie între două entităţi A şi B este considerată redundantă dacă relaţia se poate deduce din două relaţii A → C şi C → B create anterior. Un exemplu de relaţie redundantă este prezentat în figura următoare:

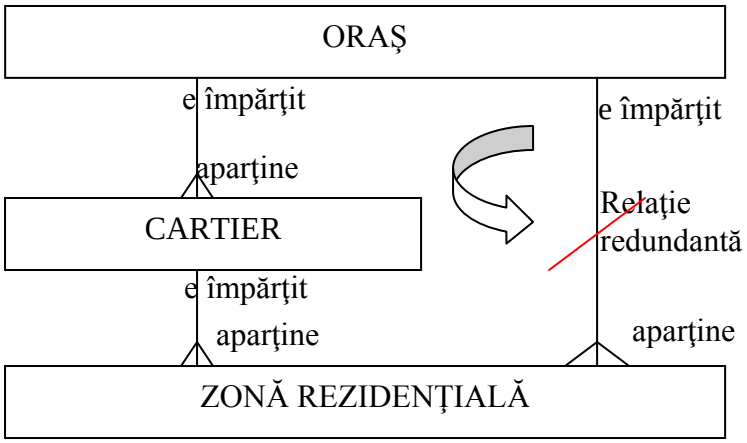


Figura 2.5. Relaţie redundantă

Dacă un oraş e împărţit în unul sau mai multe cartiere, şi un cartier e la rândului împărţit în mai multe zone rezidenţiale, se poate deduce că oraşul este împărţit în zone rezidenţiale, fără a fi necesar să marcăm acest lucru în diagramă. Ar apărea o relaţie ciclică, redundantă.

Relaţia PROFESOR → NOTA ar putea fi considerată redundantă, deoarece din diagramă rezultă faptul că un candidat aparţine unei clase ce este repartizată unei

subcomisii de examinare, formată din doi profesori evaluatori. Se poate deduce deci ce profesori au evaluat un elev, nefiind necesară o relaţie directă între CANDIDAT şi NOTA. Pentru a destrăma bucla ar trebui eliminată o relaţie.

Dacă am elimina relaţia PROFESOR → NOTA, neexistând o ierarhie între relaţiile PROFESOR→SUBCOMISIE→CLASA→CANDIDAT→NOTA, nu am ştii ce profesor a acordat nota.

Dacă eliminăm relaţia PROFESOR→SUBCOMISIE, nu am cunoaşte (sau ar fi dificil de aflat) fiecare profesor cărei subcomisii aparţine.

Soluţia este păstrarea diagramei în forma prezentă, nefiind de fapt o situaţie de redundanţă, deoarece numele relaţiilor nu sugerează că dacă un profesor este alocat unei subcomisii, el evaluează neapărat. În cazul existenţei unei singure subcomisii, toată componenţa comisiei (inclusiv preşedintele) este asociată cu această subcomisie.

Rombul existent pe linia relaţiei NOTA → CANDIDAT precizează că relaţia este *non-transferabilă*: o notă atribuită unui candidat nu poate fi transferată altui candidat.

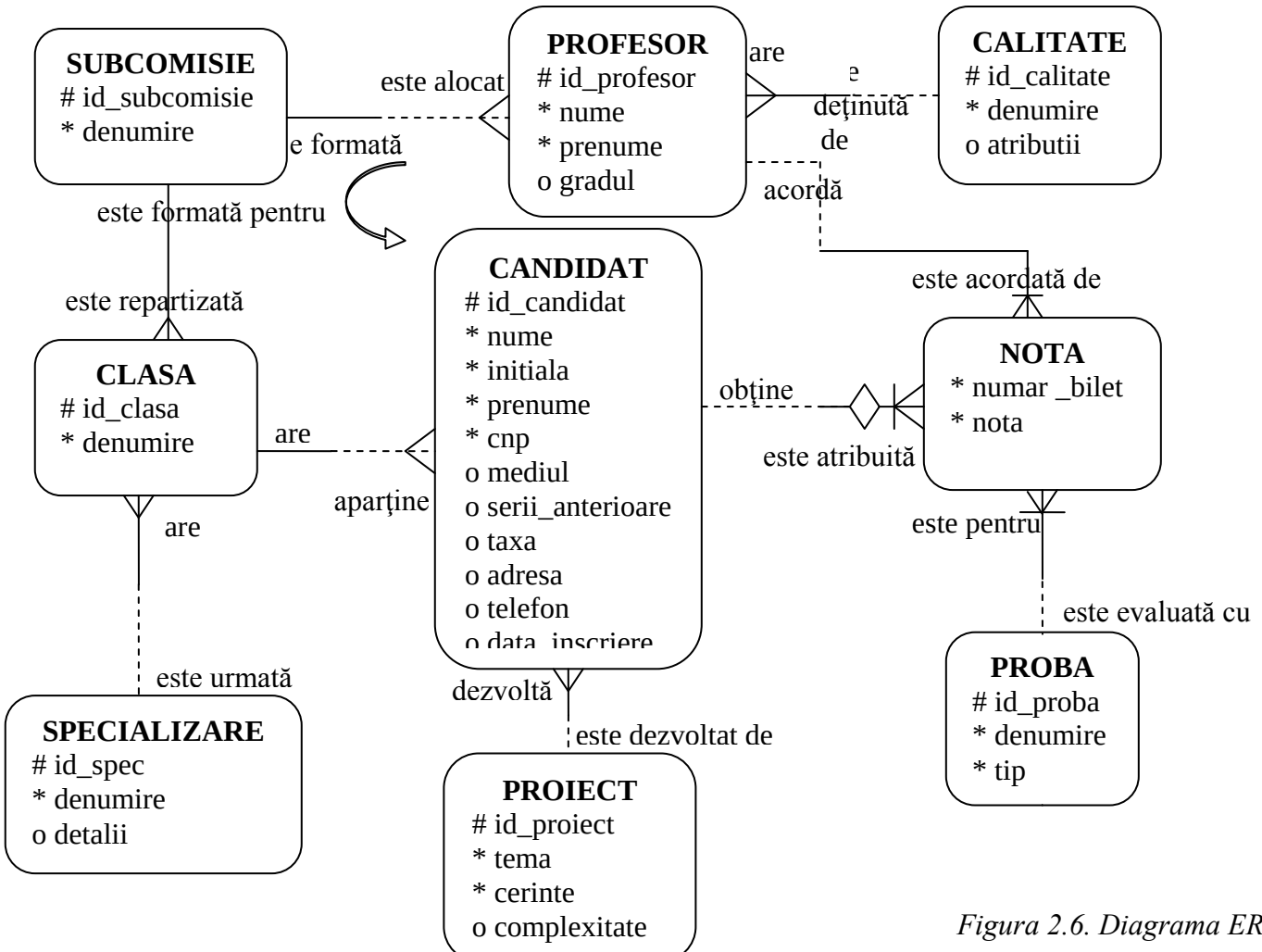


Figura 2.6. Diagrama ERD finală

2.9 Limbajul SQL. Introducere

Primele sisteme de baze de date relaţionale au apărut în 1970. Cele mai populare SGBD-uri relaţionale sunt: Oracle, Microsoft SQL Server, MySQL. Toate aceste sisteme de baze de date relaţionale au în comun limbajul standard de interogare a bazei de date numit **SQL**.

SQL - **S**tructured **Q**uery **L**anguage este un limbaj de baze de date realizat pentru a extrage informaţii şi a administra bazele de date relaţionale. Limbajul **SQL** a devenit standard **ANSI** (**A**merican **N**ational **S**tandards **I**nstitute) în 1986. Fiecare sistem de management al bazei de date (**RDBMS** - **R**elational **D**atabase **M**anagement **S**ystem) are propria versiune de limbaj **SQL**, bazată pe standardul **SQL**. Astfel, limbajul **SQL** folosit în **MySQL**, fată de limbajul **SQL** folosit în PostgreSQL sau Oracle, deşi asemănătoare, au elemente distincte, specifice acelui **RDBMS**.

MySQL este o aplicaţie comercială pentru managementul bazelor de date relaţionale (pe scurt un **RDBMS**) foarte populară, mai ales în dezvoltarea aplicaţiilor web. **MySQL** este dezvoltată de firma suedeză MySQL AB ce a fost între timp cumpărata de Sun Microsystems.

Echipele ce au dezvoltat limbajul PHP şi baza de date MySQL au colaborat cu succes de-a lungul timpului pentru a oferi o interoperabilitate ridicată între cele două programe, astfel încât prima preferinţă a programatorilor dezvoltatori în PHP pentru baze de date este MySQL.

În plus, PHP are extensii (set de funcţii) pentru a lucra şi cu alte baze de date: PostgreSQL, Oracle, SQL Server, etc.

2.10 Clienţi MySQL

Sistemele de baze de date sunt concepute într-o arhitectura client-server. Astfel, serverul de baze de date este programul principal ce stochează şi manipulează datele, şi răspunde clienţilor ce se conectează la acesta pentru a cere informaţii sau pentru a trimite cereri de altă natură (adăugări, modificări, etc). Serverul **MySQL** şi clientul **MySQL** folosit pentru interogare pot fi instalate pe acelaşi calculator, dar nu neapărat. Dacă lucrăm local (pe calculatorul propriu) şi folosim un program ca **WAMP** server, atât serverul **MySQL** cât şi clientul **MySQL** pe care-l alegem, vor fi instalate pe calculatorul nostru.

În momentul când mutăm baza de date pe un server de hosting, serverul **MySQL** va fi pe acel server de hosting iar clientul MySQL poate fi tot pe acel server (de exemplu phpMyAdmin) sau ne putem conecta cu un client MySQL instalat pe calculatorul nostru.

Aşadar, **SQL** este un limbaj special conceput pentru comunicarea cu bazele de date.

2.11 Medii de lucru

În continuare vom prezenta programele pe care le vom folosi pentru a testa noţiunile pe care le vom învăţa. În primul rând avem nevoie de instalarea pe calculator a unui server de baze de date **MySQL**. În acest sens vom instala un program numit **WAMP** care instalează local, pe calculator, un server de Apache şi unul de baze de date **MySQL**.

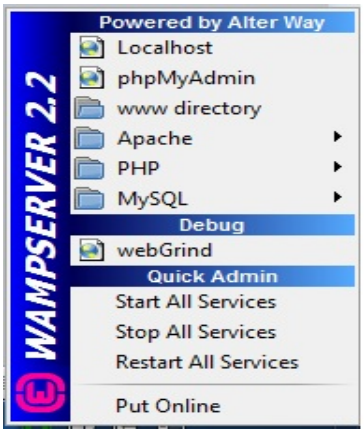
Adresa de la care se poate descărca acest program este următoarea: http://www.wampserver.com/en/

Acest program poate fi folosit şi atunci când realizăm aplicaţii web pe calculatorul propriu în limbajul PHP şi avem nevoie de un server Apache pentru a le testa.

Există şi alte programe care odată instalate pe calculator şi lansate în execuţie ne oferă un server de baze de date. De exemplu: XAMPP sau EasyPHP.

După instalarea **WAMP** se lansează în execuţie acest program. La pornire pictograma acestei aplicaţii se plasează în partea dreaptă a barei de start şi atunci când toate serviciile oferite sunt pornite are culoarea verde.

În continuare iată şi fereastra **WAMP** care se deschide la clic pe această pictogramă:



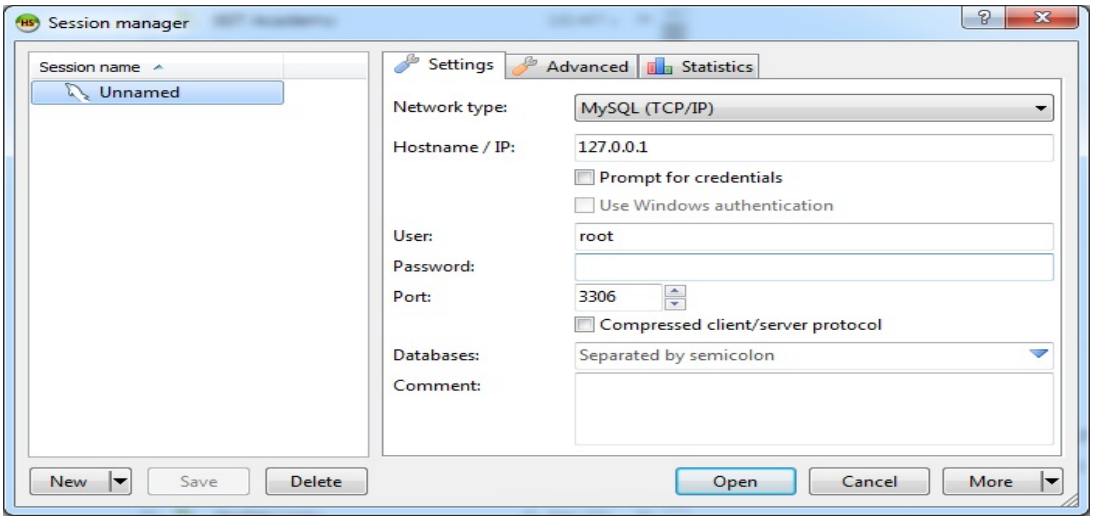
În această imagine se observă printre serviciile oferite de **WAMP** şi **MySQL**.

În cazul în care serviciul este oprit se apasă opţiunea *Start All Services*. De asemenea, pot fi oprite serviciile prin opţiunea *Stop All Services* sau restartate prin opţiunea *Restart All Services*.

De asemeneam vom instala şi un program care ne permite să lucrăm efectiv cu baze de date în **MySQL**. Este vorba de programul **HeidiSQL**.

Acest program poate fi descărcat de la următoarea adresă: http://www.heidisql.com/download.php

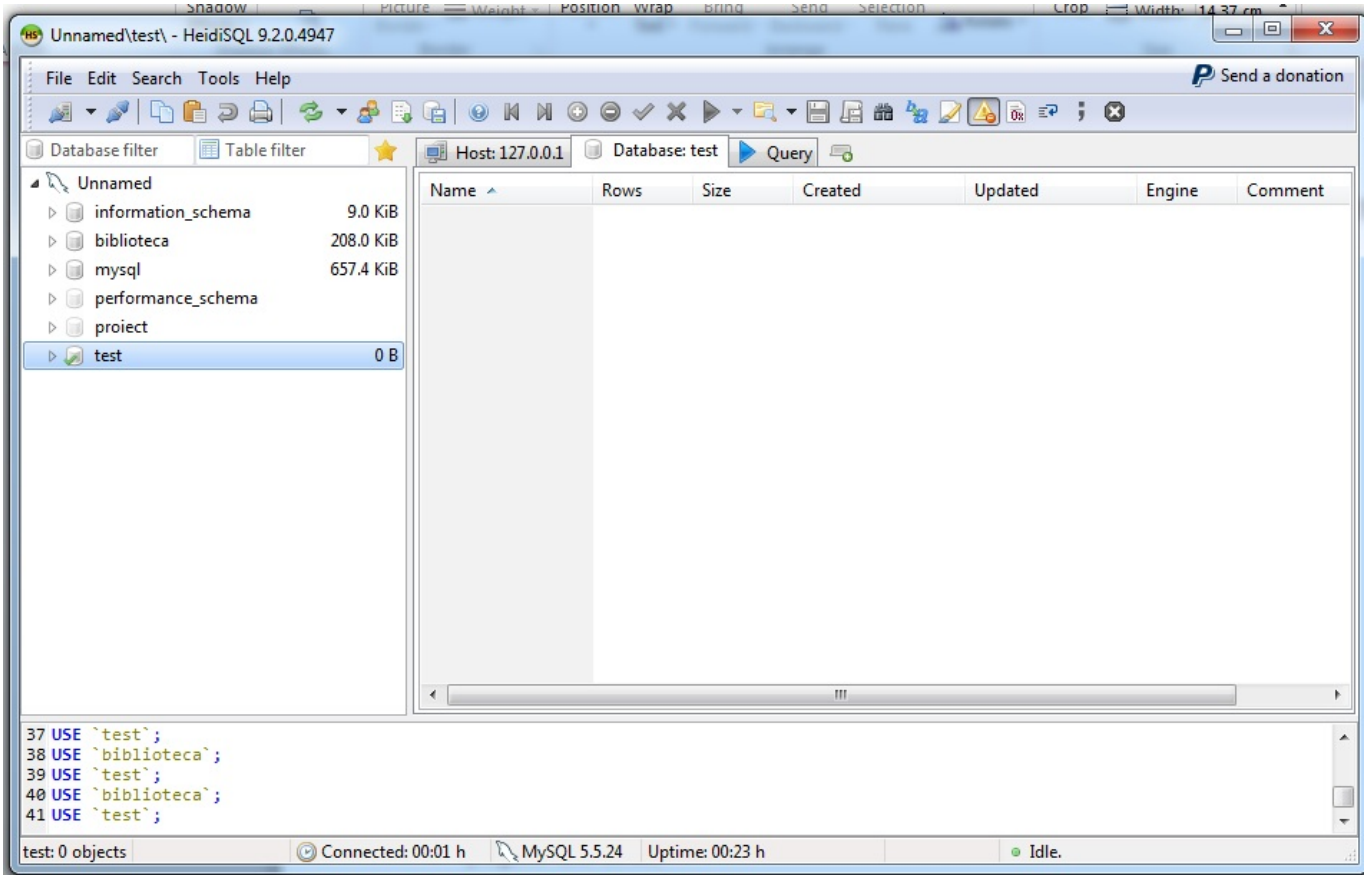
După lansarea în execuţie se va deschide următoarea adresă:



La Network Type opţiunea este MySQL, întrucât ne conectăm la un server local, la Hostname/IP este trecută adresa IP corespunzătoare localhost (127.0.0.1).

Conectarea la baza de date se face cu userul root.

Se apasă butonul Open şi se deschide fereastra următoare:



În această fereastră, în partea stângă se observă bazele de date disponbile, iar în partea dreaptă vedem tabelele bazei de date selectate, dacă există sau avem tab-ul *Query* care permite scrierea de instrucţiuni *MySQL* şi rularea acestora prin acţionarea butonului *Execute SQL*.

Această lecţie a dezvoltat conceptul de proiectare a unei baze de date relaţionale. Am prezentat în cadrul ei exemple concrete de realizare a design-ului unei baze de date. Conceptul a fost prezentat şi explicat pe larg, cu exemple concrete. Tot în cadrul acestei lecţii s-a realizat şi o introducere în limbajul **SQL**.

De asemenea, s-a făcut şi o prezentare a aplicaţiilor pe care le vom folosi mai departe pentru conectarea la o bază de date **MySQL** şi pentru realizarea de operaţii pe baza de date.

În următoarea lecţie se va trece la prezentarea sintaxei **SQL**. Vom discuta pe larg despre **L**imbajul de **D**escriere a **D**atelor (**LDD**) şi despre comenzile (instrucţiunile) acestui limbaj care se referă la structura bazei de date şi a tabelelor componente. Va fi prezentată sintaxa precum şi exemple concrete de utilizare a acestor comenzi. Vor fi prezentate, de asemenea, tipurile de date existente în **MySQL**.