UNIVERSITATEA DANUBIUS GALAȚI
Facultatea de Ştiințe economice

Note de curs la disciplina
Baze de date

Anul II

Titular curs,
Lector inf. drd. Gheorghe PANFILOIU

Galați - 2009

Noțiuni fundamentale despre baze de date

Nu există o singură definiție pentru noțiunea de bază de date. În continuare voi prezenta două definiții:

Definiția 1: O bază de date este o colecție de date interrelaționate gestionate ca o singură entitate[1].

Definiția 2: O bază de date conține toate informațiile necesare despre obiectele (datele) ce intervin într-o aplicație, relațiile logice între aceste informații şi tehnicile de prelucrare corespunzătoare.

Într-o bază de date are loc o „integrare a datelor", în sensul că mai multe fișiere sunt privite în ansamblu, eliminându-se pe cât posibil informațiile redondante. De asemenea se permite accesul simultan şi în viziune proprie la aceleaşi date, în acelaşi loc sau distribuite spațial, a mai multor utilizatori.

Sistemul de programe care permite construirea unor baze de date, introducerea datelor în bazele de date şi dezvoltarea de aplicații se numeşte sistem de gestiune a bazelor de date(SGBD - Sisteme De Gestion de Base de Donne; DBMS - Data Base Management System). Un SGBD dă posibilitatea utilizatorului să aibă acces la date folosind un limbaj de nivel înalt, apropiat de limbajul natural, pentru a obține informații, utilizatorul făcând abstracție de algoritmii aplicați pentru selecționarea datelor implicate şi a modului de memorare a lor. SGBD-ul poate fi privit şi ca o interfață între utilizatori şi sistemul de operare. Din punct de vedere al complexității un SGBD este la nivelul unui sistem de operare.

Componente principale ale unui SGBD:

- un limbaj de descriere a datelor (LDD) care permite descrierea structurii unei baze de date, a fiecărei componente a ei, a relațiilor dintre componente, a drepturilor de acces ale utilizatorilor la baza de date, a restricțiilor în reprezentarea datelor, etc;
- un limbaj de cereri (LC) sau limbajul de prelucrare a datelor (LPD), ce permite operații asupra datelor aflate în baza de date, cum ar fi: încărcarea bazei de date, inserarea unui nou element, ştergerea unui element, modificarea unui element, căutarea unor elemente, realizarea a diferite statistici asupra datelor, etc.

Limbajele LDD şi LC sunt, cel mai adesea, extensii ale unor limbaje de programare numite limbaje gazdă. Compilarea comenzilor pentru descrierea datelor sau pentru operarea cu date se reduce la o precompilare, adică la transformarea comenzilor într-o succesiune de instrucțiuni ale limbajului gazdă, care prin execuție, dau rezultatul dorit. O altă modalitate de lucru este aceea a transformării comenzilor în programe executabile, care sunt puse în lucru de către utilizatori. Comenzile sunt descrise prin sintaxe specifice fiecărui tip de SGBD, iar activarea lor se face automat, prin rutine scrise, cel mai adesea, în limbajul gazdă. Iată de ce calitățile unui SGBD depind de calitățile limbajului gazdă, în cea mai mare măsură.

O structură posibilă pentru un SGBD poate fi cea din figura 1. Procesorul de cereri prelucrează cererile curente ale utilizatorilor, cereri ad-hoc, emise la un terminal, sau sub formă de programe de aplicații scrise în LC, transformându-le în comenzi

[1] Andy Oppel, SQL fără mistere, Editura Rosetti Educațional, Bucureşti, 2006

executabile de către gestionarul bazei de date; compilatorul LDD interpretează şi transformă descrierile utilizatorului în comenzi pentru inițierea sau modificarea bazei de date; gestionarul bazei de date transformă comenzile de cereri şi descrieri de baze de date în operații executabile de către sistemul de gestiune a fişierelor, care operează asupra datelor aflate în diferitele fişiere.

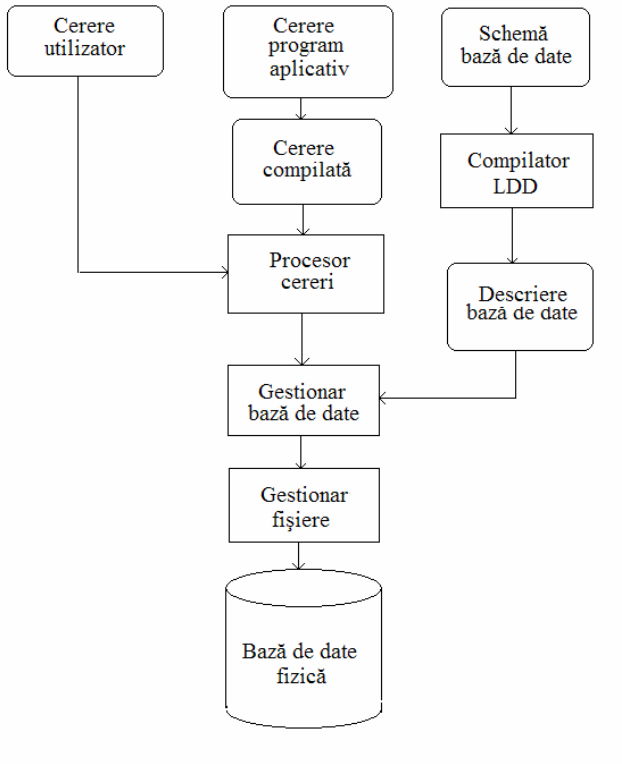


Figura 1

Dintre sarcinile pe care le îndeplineşte gestionarul bazelor de date putem enumera următoarele:

- reducerea redondanțelor prin identificarea informațiilor comune şi alcătuirea corespunzătoare a aplicațiilor;
- eliminarea inconsistențelor ce rezultă din reducerea redondanțelor;
- utilizarea simultană a datelor de mai mulți utilizatori;
- standardizarea informațiilor;
- asigurarea securității bazelor de date, în sensul acordării şi urmăririi modului de acces al utilizatorilor la diferitele părți componente ale bazelor de date;
- asigurarea integrității bazelor de date, în sensul păstrării corectitudinii înformațiilor conținute în baza de date prin testele aplicate datelor introduse în aceasta;
- asigurarea sincronizării în cazul utilizării bazei de date simultan de mai mulți utilizatori sau a distribuirii informației pe mai multe sisteme.

În ceea ce priveşte utilizatorii bazelor de date, aceştia pot fi împărțiți în următoarele clase:

- *administratorul sistemului de baze de date*, care stabilişte bazele de date de pe un sistem de calcul, alocă spaţii de memorare şi asigură drepturi de acces;
- *administratorul bazei de date*, care stabilişte structura iniţială a bazei de date şi modul de memorare a datelor la nivel fizic, acordă utilizatorilor drepturi de acces la baza de date sau părţi ale acesteia, stabileşte condiţiile pentru asigurarea securităţii şi integrităţii datelor, modifică structura bazei de date dacă este nevoie, asigură întreţinerea bazei de date făcând copii şi reconstituind eventual baza de date în cazul în care au apărut erori datorate componentelor soft, hard şi răspunde în general, de modul de utilizare al bazei de date;
- *programatorii de aplicaţii*, care pot scrie programe în LC, acestea fiind apoi compilate şi memorate în fişiere, putând fi lansate în execuţie de utilizatori, prin invocarea numelui asociat lor;
- *utilizatorii obişnuiţi*, care pot să obţină informaţii fără a avea cunoştinţe de programare. Informaţiile pot fi obţinute prin lansarea în lucru a programelor scrise de programatorii de aplicaţii, sau în mod interactiv, dacă cunosc comenzile LC.

Cele mai multe SGBD-uri conţin şi o colecţie de programe utilitare ca: procesoare pentru limbaje de cereri, editoare de rapoarte, subsisteme de reprezentări grafice, lucru în format tabelat, procesoare de limbaje naturale, programe statistice, posibilităţi de copiere, generatoare de aplicaţii.

O bază de date poate fi privită din mai multe puncte de vedere, astfel:

- punctul de vedere al administratorului bazei de date, care integrează toate vederile referitoare la baza de date într-un singur model numit *schemă conceptuală*. Schema conceptuală constituie *nivelul logic al bazei de date*;

- punctul de vedere al implementatorului bazei de date; de cele mai multe ori, el coincide cu al administratorului bazei de date, care priveşte baza de date ca o colecţie de fişiere memorate pe diferite medii externe. Acesta constituie *nivelul fizic al bazei date*, fiind de fapt singurul nivel existent efectiv;

- punctul de vedere al utilizatorilor, care lucrează cu anumite părţi componente ale bazei de date numite vederi. Vederile sunt descrise prin *subscheme în sublimbaje ale limbajului de descriere a datelor (SLDD)*. De asemenea, utilizatorii pot să primească răspunsuri la diferitele cereri formulate prin intermediul limbajului de prelucrare a datelor ce sunt specifice structurilor virtuale date de vederi.

Cele trei nivele enumerate mai sus pot conţine subnivele. Nivelul fizic poate, de exemplu, să conţină un subnivel logic, în care contează semnificaţia diferitelor câmpuri din înregistrările fişierelor şi structurile de date asociate, şi un subnivel fizic, în care contează numai modul de organizare şi gestionare a blocurilor pe memoria externă.

Nivelele logic şi cel fizic al bazei de date sunt descrise prin planuri, ce constau în enumerarea tipurilor de entităţi ce apar, relaţiile dintre aceste tipuri de entităţi şi modul de trecere de la noţiunile acestui nivel la nivelul imediat următor.

Scheme externe

Datele ce apar în schemele externe pot fi luate ca atare nivelele logic şi fizic sau pot fi deduse din aceste pe baza unor calcule. Un exemplu, clasic să spunem, îl constituie

vârsta unei persoane, care ca atribut într-o vedere nu trebuie să existe ca atare la nivelul logic sau fizic, din cauza schimbării conținutului său. La nivelul logic, atributul data_nașterii este cel indicat a fi reținut. Printr-un calcul simplu, scăderea acestei date din data_curentă, furnizată de sistemul de calcul, se poate afla vârsta persoanei în orice moment.

Noțiunile cele mai des utilizate ce intervin în vederi sunt: entitate, relație, atribut, cheie, funcționalitate, diagramă.

Entitatea este o noțiune primară, care nu se poate defini formal. Totuşi, o definiție a acestei noțiuni ar putea fi: mai multe elemente de acelaşi tip. Câteva exemple pot să conducă la înțelegerea acestei noțiuni: student, persoană, automobil, conturi, etc.

Fiecare entitate este descrisă de o mulțime de proprietăți de interes pentru problema ce se dorește a fi rezolvată, numite *atribute,* care, pentru diferitele elemente ale entității, pot să primească valori din anumite mulțimi numite *domeniul atributului* respectiv. Aceste domenii pot fi submulțimi ale diferitelor mulțimi de numere: reale, naturale, etc., sau submulțimi ale mulțimii şirurilor de caractere, de exemplu culori, mărci sau modele(de exemplu, dacă ne referim la automobile).

Un atribut sau o mulțime de atribute pentru care valorile asociate determină în mod unic orice element al entității se numeşte *cheie*. De obicei, orice entitate admite cel puțin o cheie, implicit se consideră că toate elementele unui entități sunt distincte. În cazul în care există elemente care să aibă aceleaşi valori pentru toate atributele, se ia drept cheie un atribut suplimentar reprezentat de numărul asociat elementului în entitatea respectivă(numărul de ordine), care defineşte în mod unic elementul.

Se numeşte *relație* între entitățile E_1, E_2, ..., E_n orice submulțime a produsului cartezian al mulțimilor elementelor celor **n** entități, adică mulțimi de elemente de forma (e_1, e_2, ..., e_n), unde e_1 este un element din E_1, unde e_2 este un element din E_2 ș.a.m.d. O astfel de relație se notează cu REL(E_1, E_2, ..., E_n), unde REL este numele asociat relației, iar **n** reprezintă *aritatea* acesteia. Pentru n = 2 se poate vorbi de relații binare. În cadrul acestora se poate face o clasificare a lor în funcție de câte elemente corespund fiecărui element dintr-o entitate în cealaltă entitate, astfel:

- relație unu-la-unu (notată 1:1), în cazul în care fiecărui element din prima entitate îi corespunde cel mult un element din a doua entitate şi invers;
- relație unu-la-mai-mulți (notată cu 1:N), în care fiecărui element al primei entități îi pot corespunde mai multe elemente din a cea de-a doua entitate, dar fiecărui element din a doua entitate îi corespunde cel mult un element din prima entitate;
- relație mai-mulți-la-mai-mulți (notată M:N), în cazul în care fiecărui element al primei entități îi pot corespunde mai multe elemente din cea de-a doua entitate şi invers.

Informațiile privind structura unei vederi pot fi sintetizate grafic în ceea ce se numeşte *diagramă entitate-relație*, care pune în evidență entitățile ce intervin reprezentate prin dreptunghiuri, atributele asociate lor reprezentate prin elipse şi diferitele relații dintre entități reprezentate prin săgeți(cu vârf dublu către entitatea pentru care pot apărea mai multe elemente în relație cu un element din cealaltă entitate).

Scheme conceptuale

Schema conceptuală a unei baze de date combină subschemele vederilor ce privesc o anumită aplicație într-un model unitar. Modul de descriere şi de reprezentare este acelaşi cu modul de descriere şi de reprezentare al vederilor.

O schemă conceptuală trebuie să se bazeze pe un model teoretic şi să fie simplă, adică să fie uşor de înțeles şi de prelucrat. Câteva principii care se aplică în acest sens sunt: numărul de elemente ce o constitutie să nu fie prea mare, diferitele concepte folosite să fie separate clar, să se țină sub control redondanțele.

Sistemele de gestiune a bazelor de date au fost clasificate în trei grupe mari, în funcție de tipul elementelor cu care se lucrează şi structurile obținute, astfel:

- **modelul rețea**, care permite lucrul cu entități şi relații binare de tipul unu-la-unu şi unu-la-mai-mulți, diagrama rezultată fiind un graf oarecare;
- **modelul arborescent**(ierarhic), care permite lucrul cu entități şi relații binare de tipul unu-la-unu şi unu-la-mai-mulți şi diagrama alcătuită dintr-o mulțime de arbori;
- **modelul relațional**, în care intervin numai relații şi operații cu aceste relații.

Modelul relațional este cel mai răspândit, fiind eficient în special pentru baze de date de dimensiuni mici, fiind mai mult studiat din punct de vedere teoretic şi, nu în ultimul rând, mai uşor de mânuit de utilizatori.

Scheme interne

Schemele interne descriu diferitele fişiere utilizate pentru memorarea bazei de date şi modul de operare cu acestea.

Traducerea schemelor conceptuale în scheme interne se face de obicei automat de către SGBD. Pe lângă stabilirea diferitelor tipuri de înregistrări utilizate în reprezentarea fizică a datelor, se specifică şi existența indexilor asociați unor fişiere, semnificația câmpurilor înregistrărilor, ordinea de apariție a înregistrărilor şi modul de acces.

Scurtă introducere în proiectarea bazei de date

Proiectarea unei baze de date înseamnă fixarea structurii bazei de date şi a metodelor de prelucrare a datelor, spre deosebire de utilizarea bazei de date, care priveşte informațiile stocate în acea bază la un moment dat. În mod obişnuit baza de date îşi schimbă frecvent conținutul, dar în ceea ce priveşte structura ei, aceasta rămâne nemodificată pentru perioade mai mari de timp.

Proiectarea înseamnă determinarea unui model semantic, care să reflecte cât mai fidel lumea reală, construit astfel:

1. Se identifică o mulțime de concepte semantice: entități, tipuri de entități, proprietăți ale entităților, identificatorii entităților, relații între entități, etc., care dau informații despre lumea reală.
2. Se asociază obiecte simbolice formale prin care sunt reprezentate conceptele semantice.
3. Se definesc reguli de integritate formale ce se aplică obiectelor simbolice.
4. Se defineşte o mulțime de operatori formali ce pot să transforme obiectele formale.

În proiectarea bazelor de date se ține seama de independența datelor pe diferite nivele. Spre exemplu, reprezentarea fizică a datelor se poate schimba în timp, pentru îmbunătățirea performanțelor în ceea ce privește timpul de acces la date şi spațiul ocupat, fără ca acestea să afecteze modul de reprezentare a datelor în schema conceptuală. Aceasta se numește *independența fizică* a datelor. Între vederi şi schema conceptuală există, de asemenea, o independență numită *independența logică* a datelor. În timpul existenței unei baze de date pot să apară modificări în schema conceptuală prin adăugarea unor noi entități, sau prin adăugarea unor noi atribute unor entități existente. Vederile care nu fac referiri la câmpurile modificate rămân neschimbate, fiind rescrise numai acele aplicații pentru care s-au modificat unele atribute, sau pot fi construite vederi noi, fără a fi necesară redefinirea schemei conceptuale asociate bazei de date.

Modelarea bazelor de date reprezintă un proces de definire amănunțită, pas cu pas, a tuturor elementelor ce permit o documentare completă privind cererea de informații. La proiectarea bazelor de date trebuie să existe o comunicare efectivă între cel care proiectează baza de date şi cei care o implementează, cei care o utilizează, pentru a găsi şi valida soluțiile alese la proiectare.

În urma fazei de proiectare se dorește obținerea unei baze de date care să respecte următoarele calități:

- *corectitudine*: care înseamnă reprezentarea cât mai fidelă în baza de date a modului obişnuit de lucru cu datele în sistemul real;
- *consistență*: adică informațiile corespunzătoare obiectelor cu care se lucrează în baza de date (nume, definire, relații, documentare, etc.) să nu conțină contradicții;
- *distribuire*: informațiile să poată fi utilizate de aplicații multiple şi să poată fi accesate de mai mulți utilizatori, aflați în diferite locuri, utilizând medii de calcul diverse şi acoperind un număr mare de cereri posibile;
- *flexibilitate*: facilități de adăugare componente care să reflecte cereri noi de informații, să îmbunătățească peformanțele sau să adapteze datele pentru eventuale schimbări din lumea reală.

Modelul logic al datelor

Cea mai importantă parte în construirea unei baze de date o constituie studiul sistemului ce urmează a fi reflectat în aceasta. Stabilirea informațiilor relevante pentru sistem şi a relațiilor dintre ele este un lucru esențial pentru etapele de construire a bazei de date şi a aplicației pe care aceasta se bazează, de aceea prima etapă în construirea unei baze de date o constituie o apreciere generală a sistemului. Se va alcătui astfel o schiță preliminară, care cuprinde, printre alte informații, şi modul în care sistemul este văzut de diferitele persoane implicate în sistemul respectiv. Se creează, astfel, un model informațional în care sunt cuprinse principalele funcțiuni şi fluxul de informații din sistem. Sistemul trebuie privit unitar şi nu ca o alăturare a componentelor sale. În acest sens trebuie avut în vedere că multe părți sunt folosite în comun de diferitele componente ale sistemului.

Modelul cel mai frecvent utilizat, aşa cum s-a specificat şi mai sus, este modelul entitate-relație (E-R), descris de Chen în 1976. Modelul are drept obiecte semantice

entitatea, proprietatea sau atributul, relația şi subtipul unei entități, elemente descrise anterior.

Modelul logic al bazei de date este reprezentat grafic prin diagrame entitate-relație. În figura 2 se poate vedea modelul logic al bazei de date considerată ca exemplu practic.

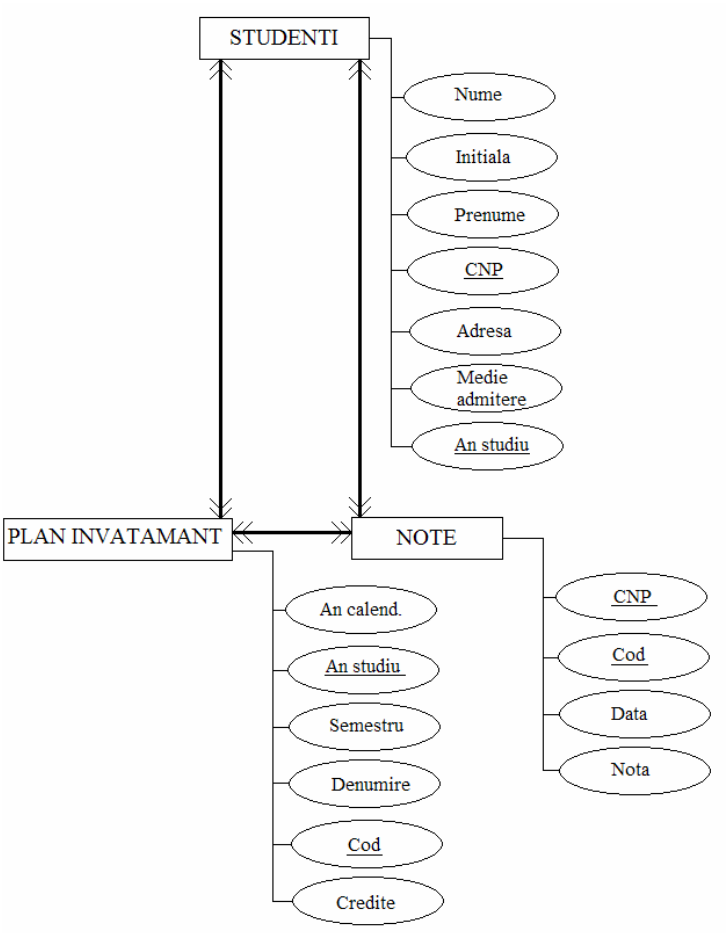


Figura 2

Entitățile sunt reprezentate sub formă de dreptunghi. Proprietățile(atributele) sunt reprezentate prin elipse ce conțin numele proprietății respective, unite prin linii de entitățile la care sunt asociate. Cheile sunt subliniate. Relațiile dintre entități sunt reprezentate prin săgeți. Săgeata se dublează, dacă relația este la-mai-mulți.

Vederile utilizatorilor

Orice aplicație presupune utilizarea unei părți din baza de date, folosind informațiile într-un mod determinat. Această parte de informație se numeşte *vedere*. O bază de date poate să aibă una sau mai multe vederi. Fiecărei vederi îi corespunde un grup de utilizatori. Pentru un grup particular se definesc tipurile de cereri de informații şi modul de determinare a datelor din baza de date care formează răspunsuri la aceste cereri.

Modelul relațional de baze de date

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

- structura datelor, prin definirea unor domenii şi a relațiilor n-are (atribute, tupluri, chei primare, etc.)
- integritatea datelor prin impunerea unor restricții
- prelucrarea datelor prin operații din algebra relațională sau calculul relațional.

Modelul relațional se bazează pe noțiunea matematică de *relație* şi anume ca o submulțime a produsului cartezian a unei liste finite de mulțimi numite *domenii*. Elementele unei relații se numesc *tupluri*, iar numărul de domenii (nu toate distincte) din produsul cartezian se numeşte *arietatea relației*.

Cel mai adesea relațiile sunt reprezentate sub forma unor tabele în care fiecare rând reprezintă un tuplu şi fiecare coloană reprezintă valorile tuplului dintr-un domeniu dat al produsului cartezian. În reprezentarea sub formă de tabel a unei relații, coloanelor şi respectiv, domeniilor corespunzătoare lor li se asociază nume intitulate *atribute*. Mulțimea numelor atributelor unei relații se numeşte *schemă relațională*. Astfel, dacă, relația *R* are atributele A_1, A_2, ..., A_n, atunci schema relațională se notează $R(A_1, A_2, ..., A_n)$.

Pentru relațiile ce constituie o bază de date se fac o serie de presupuneri inițiale, cele mai importante fiind:

- neexistența unor tupluri duplicate;
- neapariția într-o ordine dată a tuplurilor;
- neapariția într-o ordine dată a atributelor;
- posibilitatea ca toate atributele să aibă numai valori atomice (nedecompozabile).

Se numeşte candidat de cheie a unei relații *R* coloana sau mulțimea de coloane din *R* pentru care valorile corespunzătoare din oricare două tupluri nu coincid. Pentru fiecare relație se alege un candidat cheie care se numeşte *cheie primară* a relației. Tuplurile unei relații nu pot să conțină valoarea nulă în coloane ce aparțin cheii primare. Eventualii candidați de chei, diferiți de cheia primară se numesc *chei alternante*. Se numeşte *cheie străină* o coloană sau o mulțime de coloane a unei relații R_i ale cărei (căror) valori, dacă nu sunt nule, coincid cu valori ale unei chei primare dintr-o relație R nu neaparat distinctă de R_i.

Mulțimea tuturor schemelor relaționale corespunzătoare unei aplicații se numeşte *schema bazei de date relaționale*, iar conținutul curent al relațiilor la un moment dat se numeşte *bază de date relațională*.

În modelul relațional, entitățile sunt reprezentate sub formă de relații în care schema relațională conține toate atributele entității şi fiecare tuplu al relației corespunde unui element al entității.

Limbajele de prelucrare a datelor sau limbajele de cereri, cum se mai numesc, se împart în două mari categorii, pentru modelul relațional:

- limbaje algebrice, în care cererile sunt exprimate prin operatorii pe care trebuie să-i aplicăm relațiilor existente în baza de date pentru a obține răspunsul;
- limbaje de calculul predicatelor, în care cererile sunt exprimate prin mulțimea tuplurilor sau valorilor pentru care se specifică, sub forma unor predicate, proprietățile pe care trebuie să le îndeplinească.

Calculul predicatelor se împarte în funcție de obiectele cu care operează predicatele în următoarele subclase:

- limbaje cu calcul pe tupluri, dacă obiectele primare sunt tupluri;
- limbaje cu calcul pe domenii, dacă obiectele primare sunt domeniile diferitelor atribute ale relațiilor.

SQL (Structured Query Language)

SQL este unul din limbajele relaționale de cereri, şi formează nucleul multor sisteme de gestiune a bazelor de date. În luna mai, anul 1986, ANSI a recunoscut standardizarea limbajului SQL.

SQL, ca orice limbaj, posedă o listă de instrucțiuni. Majoritatea acestor instrucțiuni sunt executabile, ele putând fi interpretate şi executate imediat în mod interactiv sau putând fi incluse în diferite aplicații scrise în alte limbaje de programare ca APL, BASIC, C, COBOL şi altele, executându-se în momentul rulării programului respectiv. Între cele două moduri de utilizare sunt mici deosebiri. Astfel, la apelul într-un alt limbaj de programare se utilizează construcția EXEC SQL, urmată de definirea unor variabile de transfer de informații prin INTO: variabilă, care permite comunicarea între programe şi sistemul de gestiune a bazelor de date.

Forma generală a unei instrucțiuni SQL este:

SELECT [DISTINCT] **elemente**
FROM **tabele**
[WHERE **condiție**]
[GROUP BY **câmpuri**[HAVING **condiție**]]
[ORDER BY **câmpuri**]

În mod implicit la poiecție nu sunt eliminate duplicatele. Pentru a elimina duplicatele, se pune opțiunea DISTINCT după SELECT. Drept elemente se pot enumera câmpuri şi expresii cu câmpuri (despărțite ca la orice enumerare de semnul virgulă). FROM tabele precizează din ce tabele vor fi selectate câmpurile sau expresiile precizate. WHERE permite specificarea unei condiții simple sau compuse de filtrare a înregistrărilor din tabele. În rezultatul execuției acestei instrucțiuni vor fi selectate doar acele înregistrări pentru care din evaluarea condiției rezultă valoarea adevărat(true).

Condiția poate conține operatorii relaționali =, <>, >=, <=, < şi >, operatorii booleeni AND, OR sau NOT, precum şi eventual paranteze, pentru a impune o altă ordine de evaluare. Este permisă folosirea funcțiilor agregate COUNT, SUM, AVG, MAX şi MIN, care se pot aplica relațiilor unare având ca rezultat numărul(contorul), suma, valoarea medie aritmetică, cel mai mare şi respectiv cel mai mic element din cele cărora li s-a aplicat funcția respectivă. Aceste funcții agregate se pot aplica unor grupe de elemente precizate prin GROUP BY şi eventual selectate prin condiția HAVING.

În condițiile din instrucțiunea SELECT pot fi utilizate şi construcții de forma:

câmp LIKE **cuvânt**
câmp NOT LIKE **cuvânt**
câmp IS NULL
câmp IS NOT NULL

unde **câmp** este de tip cuvânt şi **cuvânt** poate conține caracterul "_", care înlocuieşte orice caracter, sau "%", care înlocuieşte orice cuvânt, orice alt caracter fiind luat în considerare. Rezultatul este o valoare de adevăr, după cum **câmp** conține o valoare şir de caractere ce se conformează modului de alcătuire din cuvânt sau nu, respectiv conține sau nu valoarea NULL. Se pot folosi, de asemenea, construcțiile IN sau NOT IN, pentru a indica apartenența sau neapartenența la o mulțime, EXISTS sau NOT EXISTS pentru a verifica existența sau inexistența unui element într-o mulțime şi se poate aplica UNION între două selecții pentru a realiza reuniunea celor două relații ce rezultă în urma operațiilor de selecție.

Se pot atribui nume unor tupluri ale unor relații folosite ca variabile libere ce pot fi utilizate în celelalte părți componente ale cererii, spre exemplu:

SELECT … FROM tabela nume1 WHERE …,

unde nume1 poate fi folosit, nume1.componentă semnificând apartenența componentei la tabela specificată(calificare).

Definirea unei tabele se face cu instrucțiunea CREATE TABLE, în care se specifică numele tabelului care se creează, numele şi tipurile de date ale atributelor sale, câmpurile care sunt componente ale cheii primare şi ale altor chei, împreună cu componentele lor, având forma generală:

CREATE TABLE **tabel**
(**definire_câmp1**[, **definire_câmp2**]…
[, PRIMARY KEY (**câmpuri**)]
[, FOREIGN KEY (**câmp**) REFERENCES **tabel**
[, FOREIGN KEY (**câmp**) REFERENCES **tabel**]…]

unde **definire_câmp** este de forma: **nume_câmp tip_dată** [NOT NULL], iar tip_dată poate fi:

- INTEGER, pentru o valoare întreagă pe un cuvânt;
- SMALLINT pentru o valoare întreagă pe un semicuvânt;
- DECIMAL(lungime, zecimale), unde lungime reprezintă numărul total de cifre + 1 (poziția pentru semn) + 1 (poziția mărcii zecimale), iar zecimale, numărul acestora;
- FLOAT(p) pentru un număr real cu p cifre binare precizie;
- CHARACTER(n) pentru cuvinte de n octeți;
- VARCHAR(n) pentru cuvinte de cel mult n octeți;
- VARGRAPHIC(n) pentru cuvinte de cel mult n caractere de câte 16 biți;

- DATE pentru date calendaristice exprimate sub forma *aaaallzz*;
- TIME pentru momente de timp exprimate sub forma *oommss*;
- TIMESTAMP pentru exprimarea în microsecunde a unei combinații de dată şi oră.

Se pot utiliza prescurtările INT, DEC sau CHAR pentru INTEGER, DECIMAL şi respectiv CHARACTER. Câmpurile pentru care s-a specificat NOT NULL trebuie să conțină o valoare definită din domeniul indicat, iar pentru acelea pentru care nu s-a precizat acest lucru pot să nu fie definite.

Pentru crearea tabelei studenți se poate da următoarea secvență:

```
CREATE TABLE studenti
(Nume CHAR(20) NOT NULL,
 Inițiala CHAR(2) NOT NULL,
 Prenume CHAR(25) NOT NULL,
 Cnp CHAR(13) NOT NULL,
 Adresa CHAR(50) NOT NULL,
 Medie_admitere DECIMAL(6,2) NOT NULL,
 An_studiu SMALLINT NOT NULL,
 PRIMARY KEY (cnp) );
```

Instrucțiunea CREATE TABLE produce la nivel fizic fişiere ce corespund tabelelor definite şi care imediat după terminarea execuției instrucțiunii sunt vide. Definirea unor tabele virtuale utilizate ca vederi se face prin instrucțiunea CREATE VIEW, forma generală a instrucțiunii fiind:

```
CREATE VIEW vedere [(coloană1[,coloană2]…)]
  AS subcerere
  [WITH CHECK OPTION];
```

în care WITH CHECK OPTION indică verificarea condițiilor din definirea vederii în momentul executării operațiilor UPDATE şi INSERT.

Crearea unui index se face cu instrucțiunea CREATE INDEX, a cărei sintaxă este:

```
CREATE [UNIQUE] INDEX index
   ON tabel (câmp1[ordine1] [,câmp2[ordine2]]…) [CLUSTER];
```

în care parametrul UNIQUE specifică necesitatea de a avea un singur tuplu în tabel pentru valorile câmpurilor specificate. Ordine1, ordine2,… poate lua una din valorile ASC, valoare implicită, care specifică o ordine crescătoare, sau DESC pentru ordine descrescătoare. CLUSTER specifică gruparea tuplurilor după adresele date de index, putând să apară în cel mult un index pentru fiecare tabel în parte.

Instrucțiunile DROP TABLE, DROP VIEW şi DROP INDEX sunt folosite pentru eliminarea unor elemente din baza de date: tabele, vederi sau fişiere index. Forma generală a acestor instrucțiuni este:

```
DROP TABLE tabel;
DROP VIEW vedere;
DROP INDEX index;
```

Eliminarea unei tabele produce eliminarea automată a tuturor vederilor în care este implicată.

Într-un tabel se poate adăuga un nou câmp prin utilizarea instrucțiunii ALTER TABLE, cu sintaxa:

```
ALTER TABLE tabel ADD câmp_nou tip_dată;
```

Sistemul adaugă în mod automat valoarea null pentru toate tuplurile(înregistrările) din tabel.

Alte instrucțiuni de prelucrare a datelor din SQL sunt: UPDATE pentru actualizări, DELETE pentru ştergerea unor tupluri şi INSERT pentru adăugarea de noi tupluri. Sintaxa acestor instrucțiuni este:

```
UPDATE tabel
SET câmp1 = expresie1 [, câmp2 = expresie2] …
[WHERE condiție];

DELETE
FROM tabel
[WHERE condiție];

INSERT
INTO tabel [(câmp1[, câmp2]…)]
VALUES (valoare1[, valoare2]…);
```

sau

```
INSERT
INTO tabel [(câmp1[, câmp2]…)]
subcondiție;
```

Forme normale ale bazelor de date relaționale

Definiție: se spune că o relație este într-o ***formă normală*** particulară dacă satisface o mulțime dată de constrângeri.

Anomalia de inserare, anomalia de ştergere(inversa anomaliei de inserare) şi anomalia de actualizare sunt cele trei tipuri de anomalii care au evidențiat necesitatea normalizării.

Anomalia de inserare reprezintă situația când nu aş putea adăuga o denumire, un cod, până nu aş avea o înregistrare de adăugat într-o tabelă.

Anomalia de ştergere reprezintă situația în care se poate pierde o informație prin ştergerea unei înregistrări.

Anomalia de actualizare reprezintă situația în care pentru a schimba o anumită valoare trebuie să se actualizeze mai multe înregistrări.

Transformarea unei relații într-o mulțime de relații de un anumit tip se numeşte ***normalizare***. Primele trei forme normale au fost definite de Codd, iar a patra şi a cincea formă normală au fost definite de Fagin.

Scopurile principale urmărite în procesul de normalizare sunt:
- eliminarea unor redondanțe;
- evitarea unor anomalii de reactualizare;
- realizarea unui proiect care să reprezinte cât mai fidel modelul real;
- stabilirea unor constrângeri de integritate simple, etc.

Prima formă normală (1NF)

Se spune că o relație este în prima formă normală (1NF) dacă fiecărui atribut îi corespunde o valoare indivizibilă (atom), deci orice valoare nu poate fi o mulțime sau un tuplu de valori în anumite domenii şi nu pot să apară grupuri repetitive.

Relațiile în prima formă normală pot fi vizualizate sub formă de tabele, permit o referire simplă a datelor prin indicarea numelui tabelului, a coloanei sau atributului şi a cheii tuplului din care face parte informația respectivă (adresare tip sistem de coordonate), operatorii pentru aceste relații sunt mai simpli şi în număr mai mic şi permit definirea unor tehnici de proiectare şi utilizare a bazelor de date.

A doua formă normală (2NF)

Fie R o schemă relațională şi A un atribut din R. Vom spune că A este un atribut prim al lui R dacă A apare în cel puțin o cheie a lui R; altfel, vom vom spune că A este un atribut neprim.

Spunem că schema relațională R este în a doua formă normală (2NF) dacă este în prima formă normală şi, pentru orice dependență X → A, cu A atribut neprim neconținut în X, care are loc în R, nu există nici o cheie care să-l conțină strict pe X.

Cu alte cuvinte, o relație R este în 2NF dacă este în 1NF şi orice atribut neprim este complet dependent de orice cheie. A doua formă normală înlătură redondanțele şi anomaliile la modificări. A doua formă normală poate să conțină aşa-numitele dependențe tranzitive, ce se deduc prin aplicarea axiomei tranzitivitățiidin alte două dependențe funcționale nebanale. Dependențele tranzitive pot să producă anomalii la inserție, ştergere şi modificare.

A treia formă normală (3NF)

Spunem că schema relațională R este în a treia formă normală (3NF) dacă este în a doua formă normală şi, pentru orice dependență X → A cu A neconținut în X care are loc în R, fie X conține o cheie, fie A este un atribut prim.

Cu alte cuvinte, relația R este în 3NF dacă şi numai dacă atributele care nu apar în chei sunt independente între ele şi sunt complet dependente de cheia primară (atributele prime nu sunt dependente tranzitiv de cheia primară).

Forma normală Boyce-Codd (BCNF)

Spunem că o relație R cu dependențele F este în forma normală Boyce-Codd (BCNF) dacă oricare ar fi dependența X → A cu A atribut neconținut în X, există o cheie

a lui R conținută în X. Orice relație normală Boyce-Codd este în a treia formă normală, dar reciproca nu este adevărată.

A patra formă normală (4NF)

Fie R o schemă relațională şi D o mulțime de dependențe funcționale şi multivaloare pe R. Spunem că R este în a patra formă normală (4NF) dacă, pentru orice orice dependență multivaloare $X \twoheadrightarrow Y$ cu Y neinclusă în X şi $XY \subset R$, există o cheie a lui R inclusă în X. Cu alte cuvinte, o relație R este în a patra formă normală dacă este în BCNF şi orice dependență multivaloare este, de fapt, o dependență funcțională.

A cincea formă normală (5NF)

Se spune că o relație R satisface dependența de uniune (join dependency sau, pe scurt, JD), şi se notează cu *(X,Y,...,Z), dacă şi numai dacă R este egală cu uniunea proiecțiilor lui R pe X, Y, ..., Z, acestea fiind submulțimi ale lui R.

O relație R este în a cincea formă normală, numită şi forma normală proiecție-uniune (pe scurt PJ/NF sau 5NF), dacă şi numai dacă orice dependență de uniune a lui R este o consecință a unui candidat de cheie a lui R.

Orice relație care este în 5NF este şi în 4NF, deoarece fiecare dependență multivaloare poate fi privită ca un caz particular de dependență de uniune. Orice relație poate fi descompusă fără pierderi la uniune într-o mulțime de relații care sunt în 5NF. Se garantează faptul că o relație 5NF nu conține anomalii ce pot fi eliminate luând proiecțiile pe diferite submulțimi ale ei.

Procesul de normalizare a relațiilor se poate descrie astfel:

1. Se proiectează relația inițială 1NF pe alte relații pentru a elimina dependențele funcționale care nu sunt complete. Se obține o mulțime de relații în 2NF.
2. Se proiectează relațiile obținute la pasul 1 pe alte relații pentru a elimina dependențele funcționale tranzitive. Se obține o mulțime de relații în 3NF.
3. Se proiectează relațiile obținute la pasul 2 pe alte relații pentru a elimina dependențele în care partea din stânga nu este o supracheie. Se obține o mulțime de relații în BCNF.
4. Se proiectează relațiile obținute în pasul 3 pe alte relații pentru a elimina toate dependențele multivaloare care nu sunt şi dependențe funcționale. Se obține o mulțime de relații în 4NF.
5. Se proiectează relațiile obținute la pasul 4 pe alte relații pentru a elimina orice dependență de uniune care nu este implicată de o cheie. Se obține o mulțime de relații în 5NF.

Integritate

Integritatea bazelor de date este în principiu dată de corectitudinea informațiilor conținute în ele şi presupune detectarea, corectarea şi prevenirea diferitelor erori neintenționate privind informațiile introduse ăn baze de date. Condițiile de integritate, numite şi reguli de integritate, nu permit introducerea în aza de date a unor date aberante şi sunt exprimate prin diferitele condiții puse asupra datelor. O serie de condiții sunt de

tip structural, legate de anumite egalități între valori, şi se exprimă prin dependențe funcționale sau prin declararea unor câmpuri cu valori unice (cel mai adesea aceste câmpuri sunt chei). O altă serie de condiții de integritate privesc valorile actuale memorate în baza de date, permitând numai introducerea unor valori dintr-un domeniu sau stabilind relații aritmetice între diferite câmpuri.

Parte practică

Pentru a putea lucra, va trebui să se instaleze serverul Apache şi sistemul de gestiune a bazelor de date MySql[2]. Pentru aceasta se vor parcurge următorii paşi:

1. Se va copia şi se va introduce adresa următoare:
   http://www.apachefriends.org/download.php?xampplite-win32-1.7.0.exe
   în browser-ul dorit. Prin această acțiune se va descărca de pe Internet programul: **xampplite-win32-1.7.0**

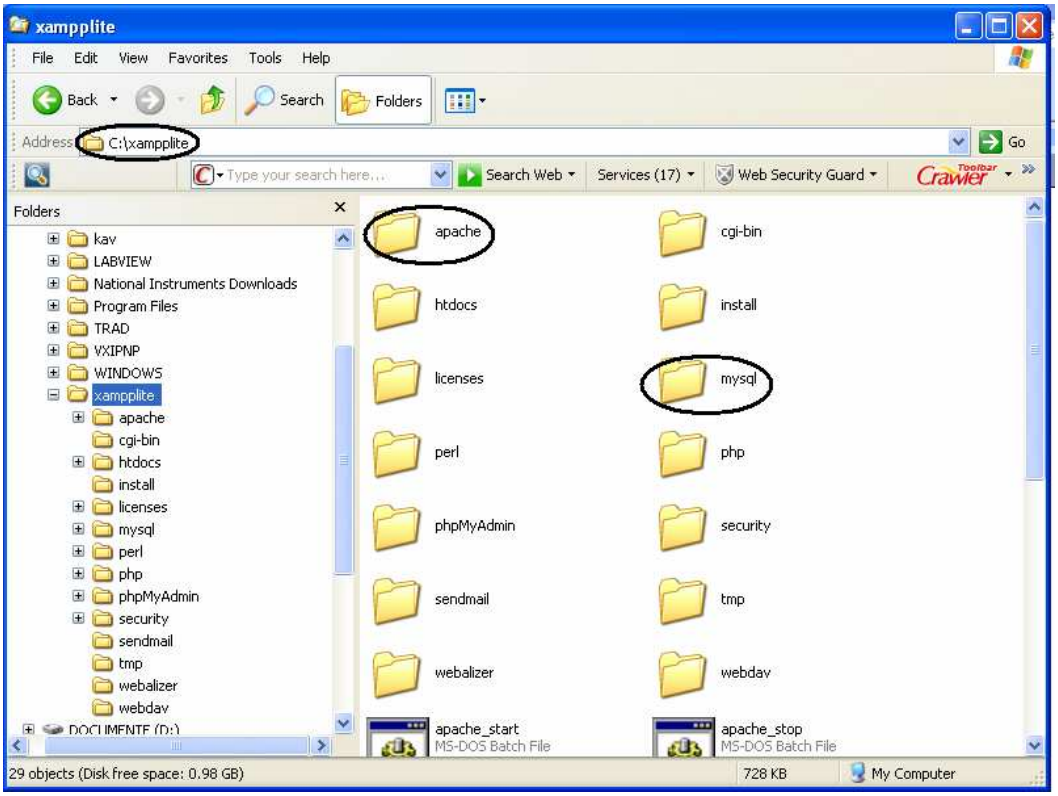
2. Prin execuția acestuia se va descărca directorul: **xampplite**, care va fi creat, sau mutat în rădăcina discului C:.
3. 

Figura 1. Instalarea directorului xampplite în sistemul de calcul

4. Se schimba directorul curent în apache.

[2] MySql funcționează doar sub serverul Apache.

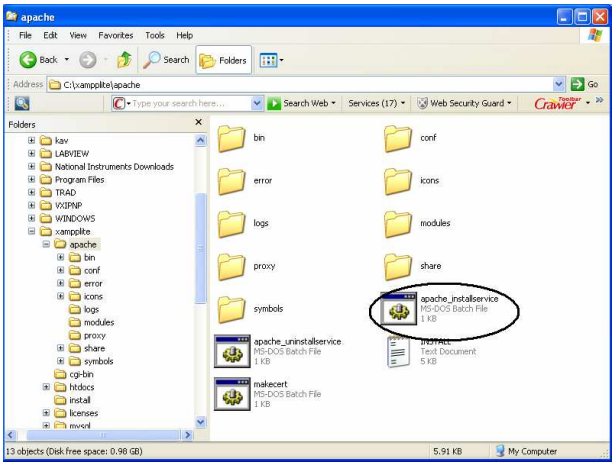


Figura 2. Instalarea serviciului de server Apache

Se va executa apache_installservice.

5. Se va schimba directorul curent în mysql.

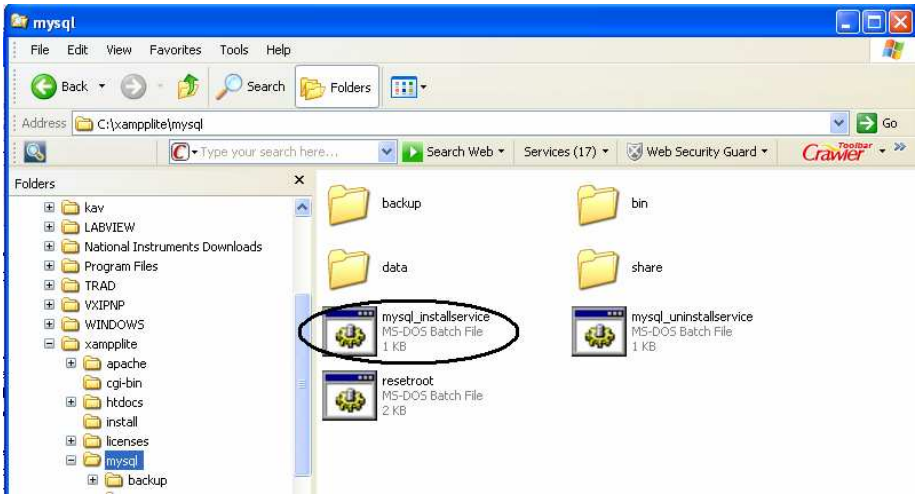


Figura 3. Instalarea serviciului MySql

Se va executa mysql_installservice.

6. Din directorul xampplite, prin manevra cunoscută se recomandă crearea unui shortcut pentru programul xampp-control.

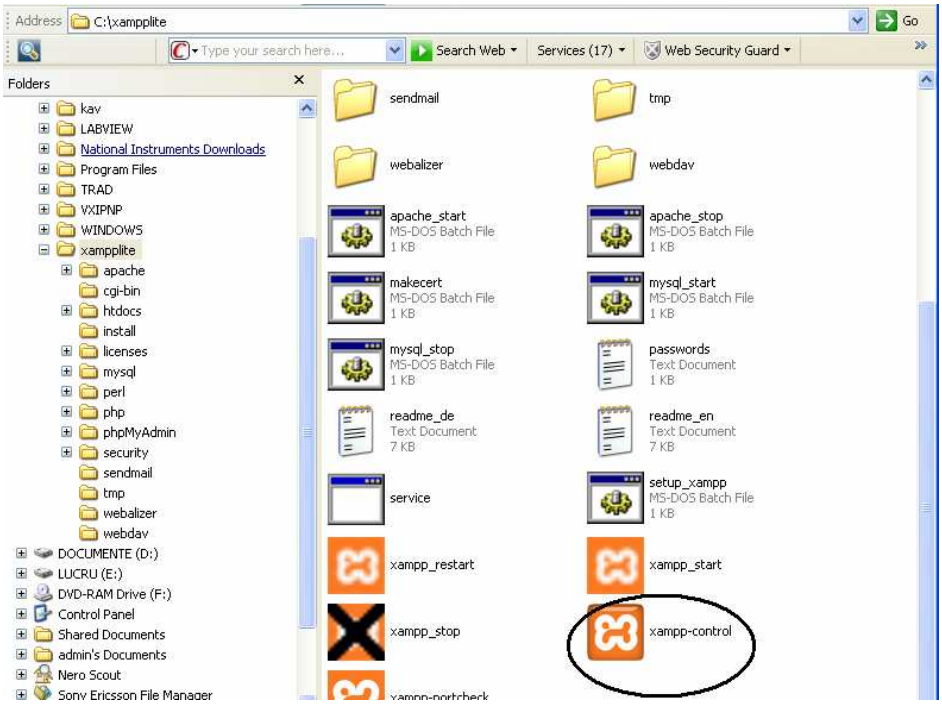


Figura 4. Crearea shortcut-ului pentru programul xampp-control

La pornirea calculatorului cele două servicii, apache şi mysql pornesc automat. Dacă nu se lucrează cu aceste servicii, se recomandă oprirea execuției lor. Prin intermediul shortcut-ului creat, conform sugestiilor de mai sus, se deschide o fereastră ca în figura 5 şi se solicită oprirea celor două servicii prin acționarea butoanelor Stop, apoi Exit.

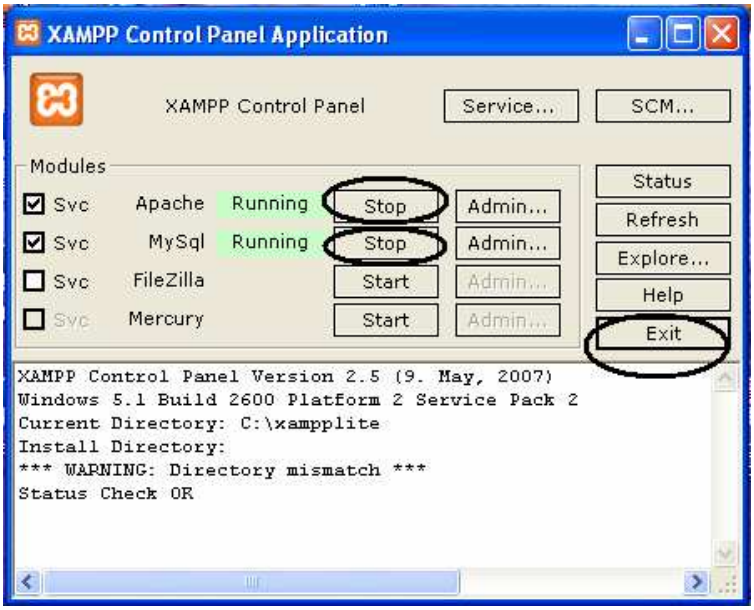


Figura 5. Oprirea serviciilor Apache şi MySql

Dacă se doreşte a se lucra cu cele două programe, se vor acționa butoanele Start pentru cele două servicii, prin intermediul aceluiaşi program xampp-control.

Exemplu: Evidența notelor studenților

Pentru realizarea unei astfel de evidențe se va crea o bază de date, cu numele Studenti, care va avea în structură trei tabele, ca în explicațiile următoare. De remarcat faptul că exemplul este simplificat, din dorința de a înțelege mecanismul de lucru.

Tabela: Studenți, care va avea următoarele câmpuri:
Nume x(20);
Inițiala x(2);
Prenume x(25);
Cnp x(13);
Adresa x(50);
Medie_admitere 9(2).9(2);
An_studiu 9(1).

Tabela: Plan_învățământ, care va avea următoarele câmpuri:
An_calendaristic 9(4);
An_studiu 9(1);
Semestru 9(2);
Denumire x(60);
Cod 9(3);
Credite 9(1).

Tabela: Note, care va avea câmpurile:
Cnp x(13);
Cod 9(3);
Data_c dată calendaristică;
Nota 9(2).

Pentru a avea acces la sistemul SGBD MySql, se va pune în lucru browser-ul dorit: Internet Explorer, Mozilla Firefox, Opera şi se va introduce la adresă şirul de caractere: **127.0.0.1** . Se va deschide fereastra ca în figura 6.

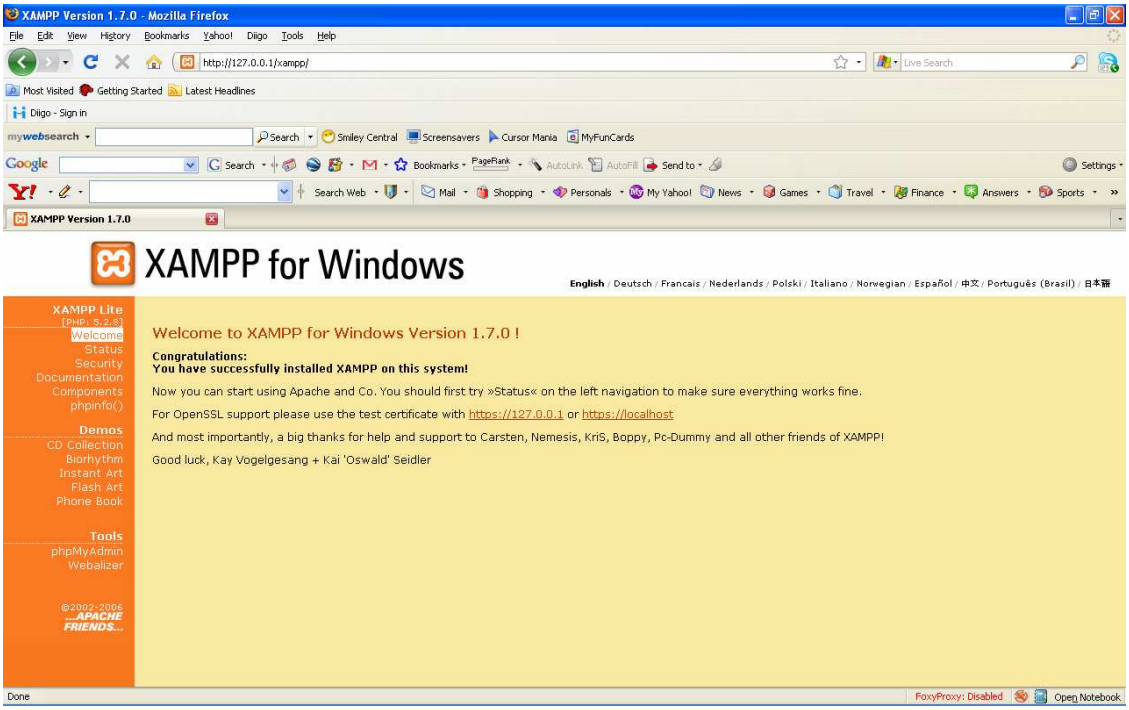


Figura 6. Pagina de pornire

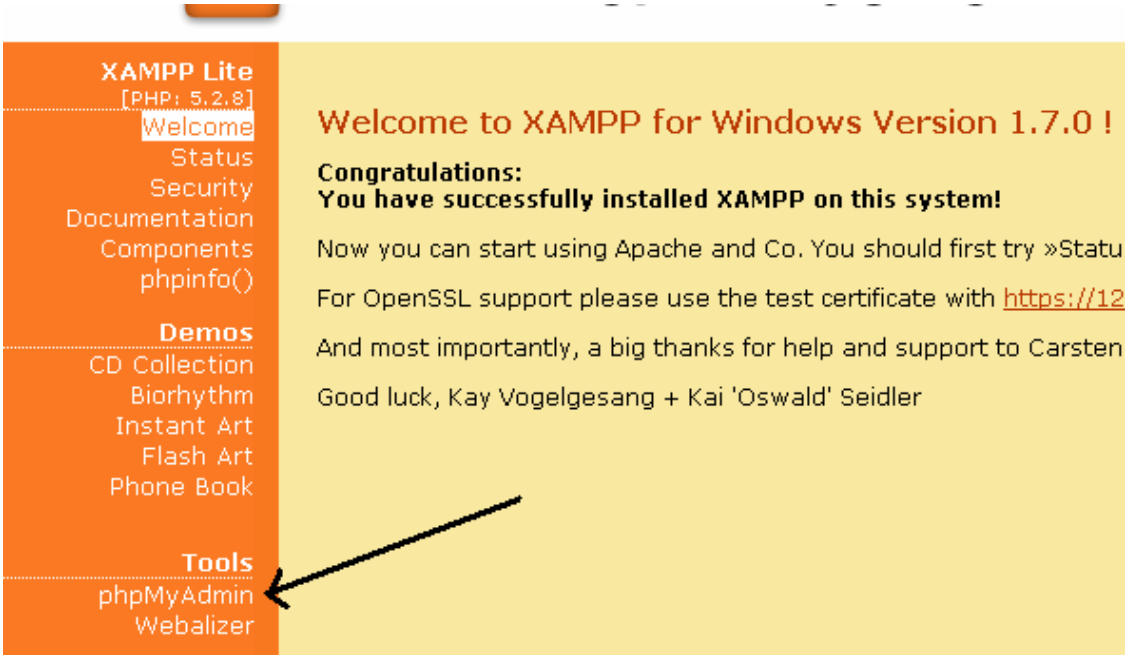


Figura 7. Accesul la SGBD MySql

Prin selectarea butonului phpMyAdmin, se va deschide fereastra ca în figura 8.

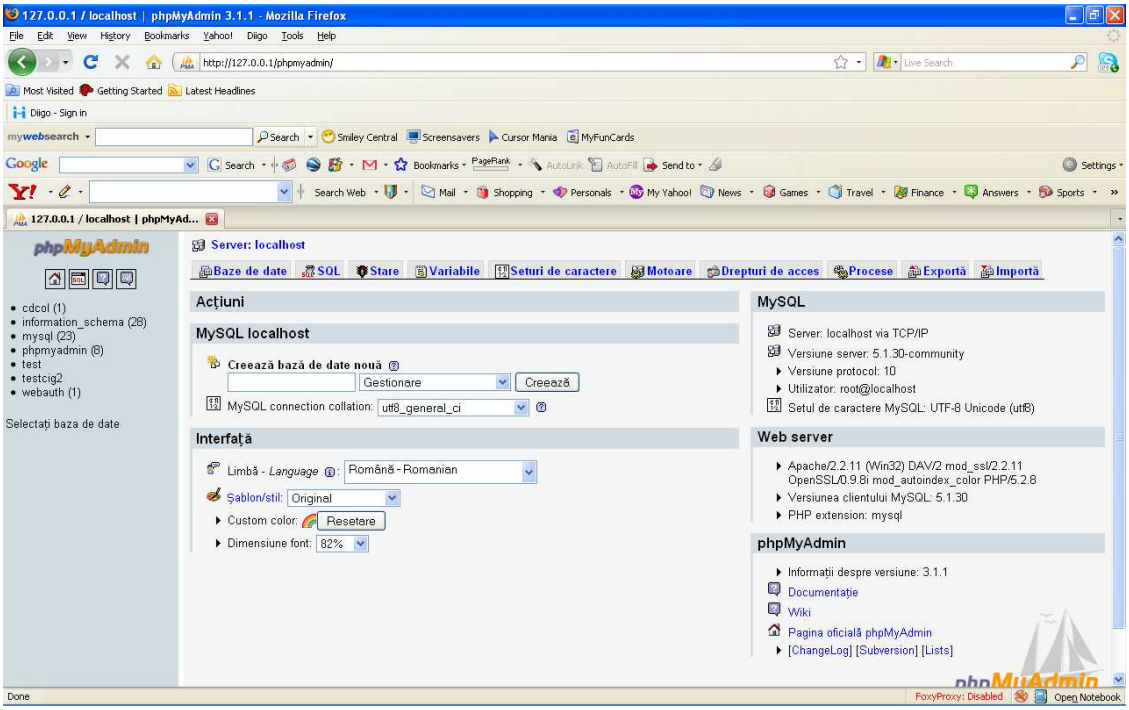


Figura 8. Pagina de start a sistemului MySql

Pentru toate instrucțiunile care vor fi executate, după completarea unor rubrici corespunzătoare, sistemul creează instrucțiuni SQL echivalente, care vor fi afișate într-o fereastră, ca în figura 9.

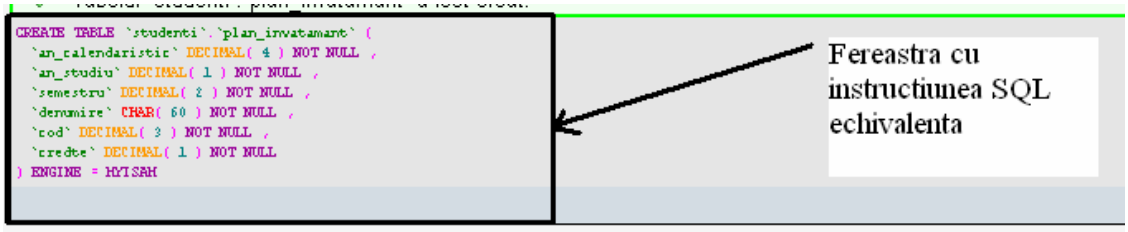


Figura 9. Fereastra cu instrucțiunea SQL echivalentă

Se poate lucra şi în “modul comandă”, prin acționarea butonului SQL, ca în figura 10 , care deschide o fereastră nouă, ca în figura, şi în care se pot introduce, pe rând, prin dactilografiere, toate instrucțiunile(comenzile) SQL, necesare prelucrărilor dorite.

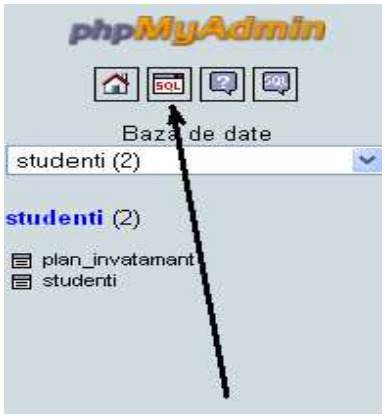


Figura 10. Solicitarea afișării ferestrei comandă

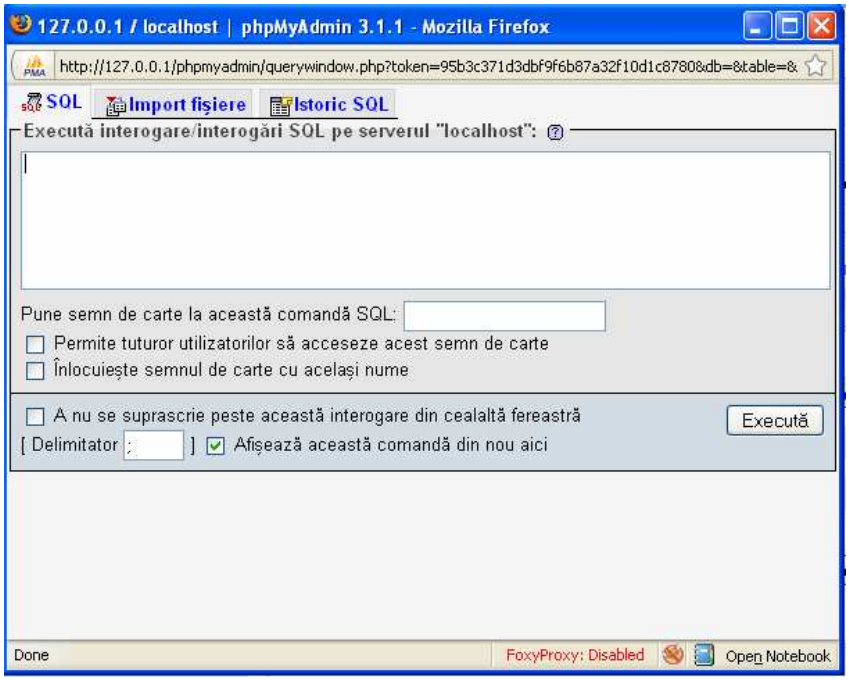
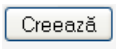


Figura 11. Conținutul ferestrei comandă

Fiecare comandă ce se va introduce, va fi o comandă validă şi corectă sintactic, iar după dactilografierea ei se va “acționa” butonul Execută.

Crearea bazei de date

Se va tasta denumirea bazei de date: “studenti”, apoi click pe butonul 

, ca în figura 12.

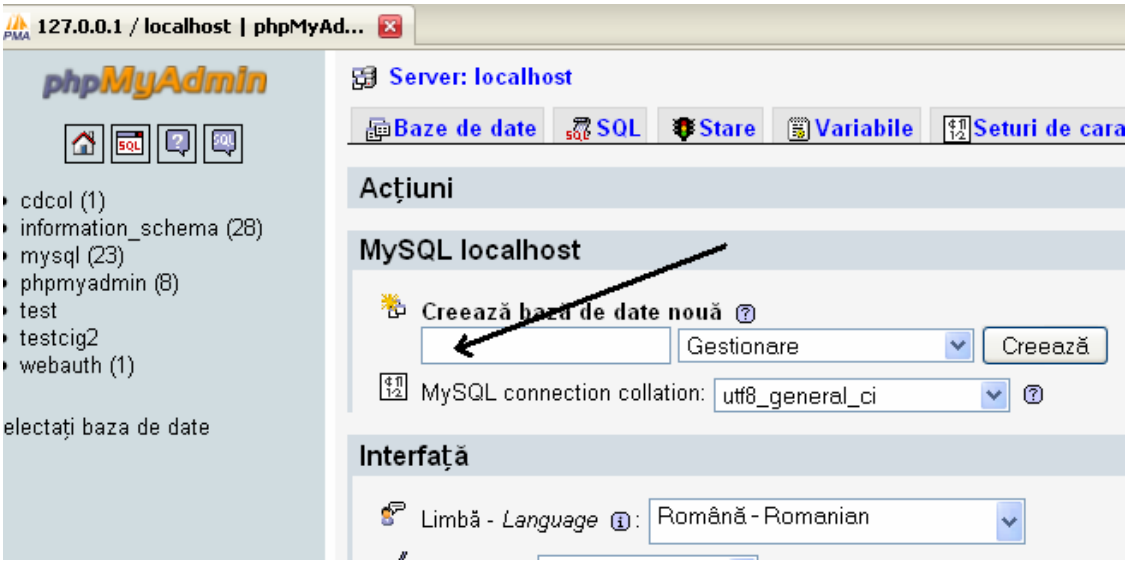


Figura 12. Introducerea denumirii bazei date

Baza de date va fi creată în **C:\xampplite\mysql\data*studenti***, adică, în directorul **C:\xampplite\mysql\data**, director numit "catalog de date", în care se creează directorul(catalogul) cu numele bazei de date pe care o creăm, ***studenti***, denumit catalogul bazei de date.

Pentru fiecare tabelă a unei baze de date se crează trei fişiere situate în catalogul bazei de date.

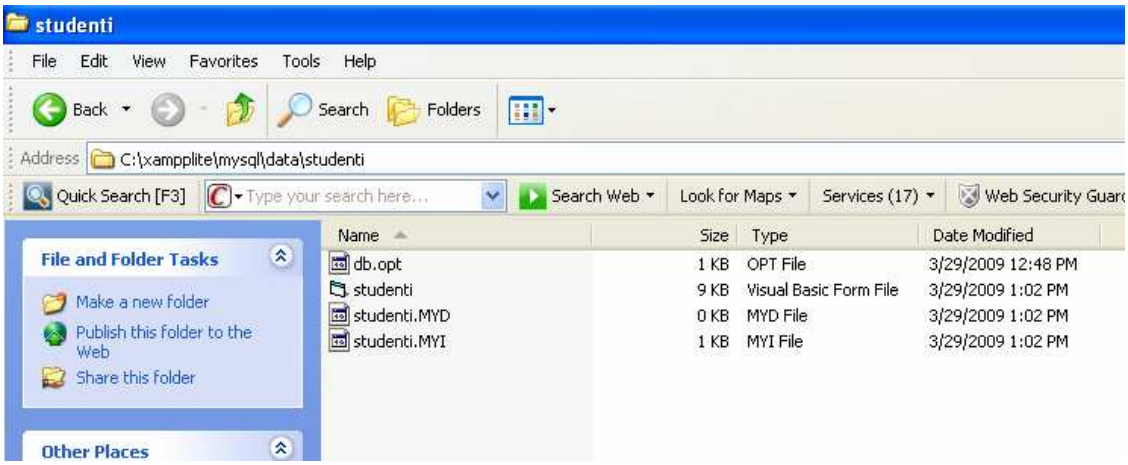


Figura 13. Conținutul directorului de date mysql şi a directorului bazei de date studenti

Fişierele au numele bazei de date (studenti) şi extensiile:

- .frm, pentru fişierul formular. Acest fişier conține structura tabelei, adică numele câmpurilor , tipurile acestora, etc.;
- .myd, pentru fişierul de date. Acest fişier conține datele înscrise în tabelă;
- .myi, pentru fişierul index. Acest fişier conține arborii de index pentru toate indexurile din fişierele de date.

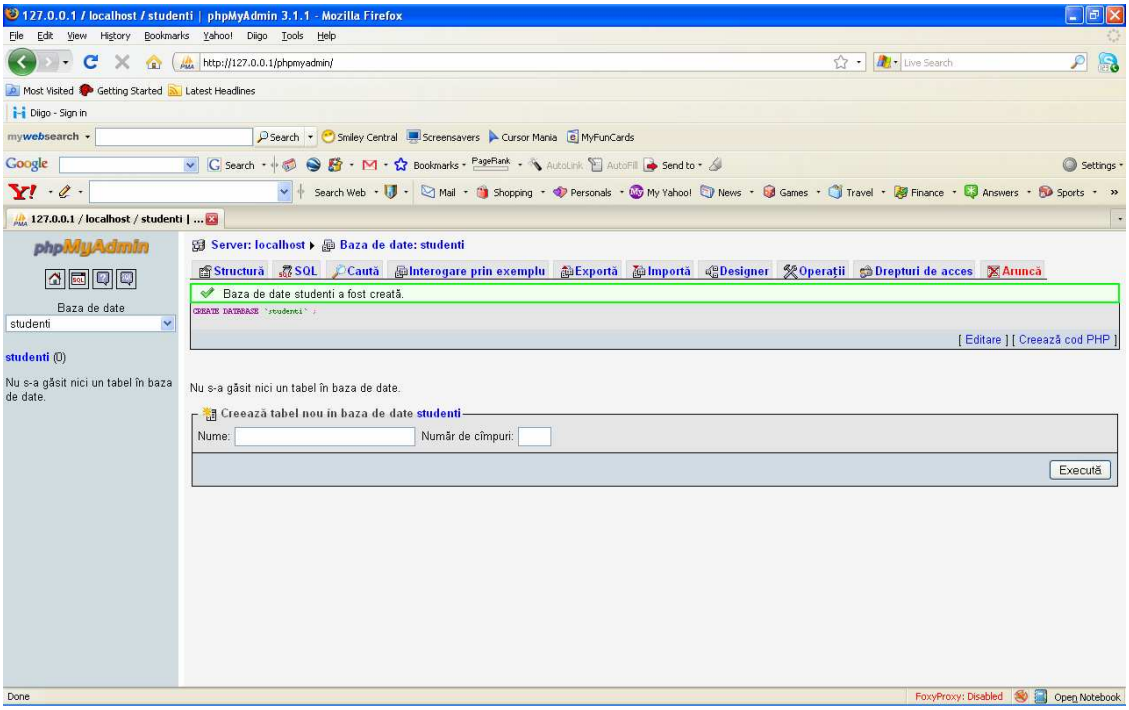
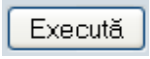


Figura 14. Fereastra afişată după crearea bazei de date

Se poate observa fereastra cu titlul **Baza de date studenti a fost creată**. În interiorul acesteia se află comanda (instrucţiunea) SQL :

CREATE DATABASE ‘studenti’ ;

În cotinuare se vor crea tabele a căror structură a fost definită. Se poate observa că trebuie să furnizăm numele tabelului şi numărul de câmpuri.

Astfel pentru tabela studenţi vom tasta „studenti” pentru numele tabelului şi respectiv “7” pentru numărul de câmpuri, în locaţiile corespunzătoare. Apoi click pe butonul 

.

Se deschide o nouă fereastră, vizualizată în figurile 15, 16, 17 următoare. Au fost necesare trei figuri, datorită faptului că imaginea ferestrei este mai mare decât mărimea ecranului, şi a fost necesară defilarea spre dreapta şi în jos.

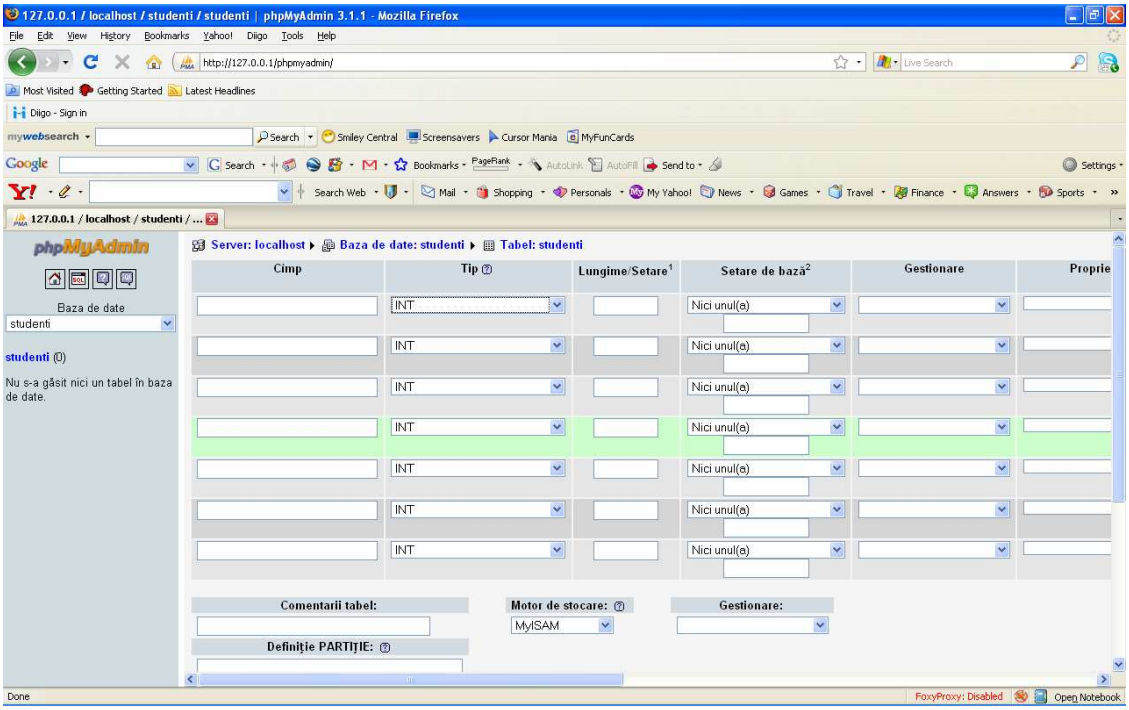


Figura 15. Fereastra de creare a structurii unei tabele (1)

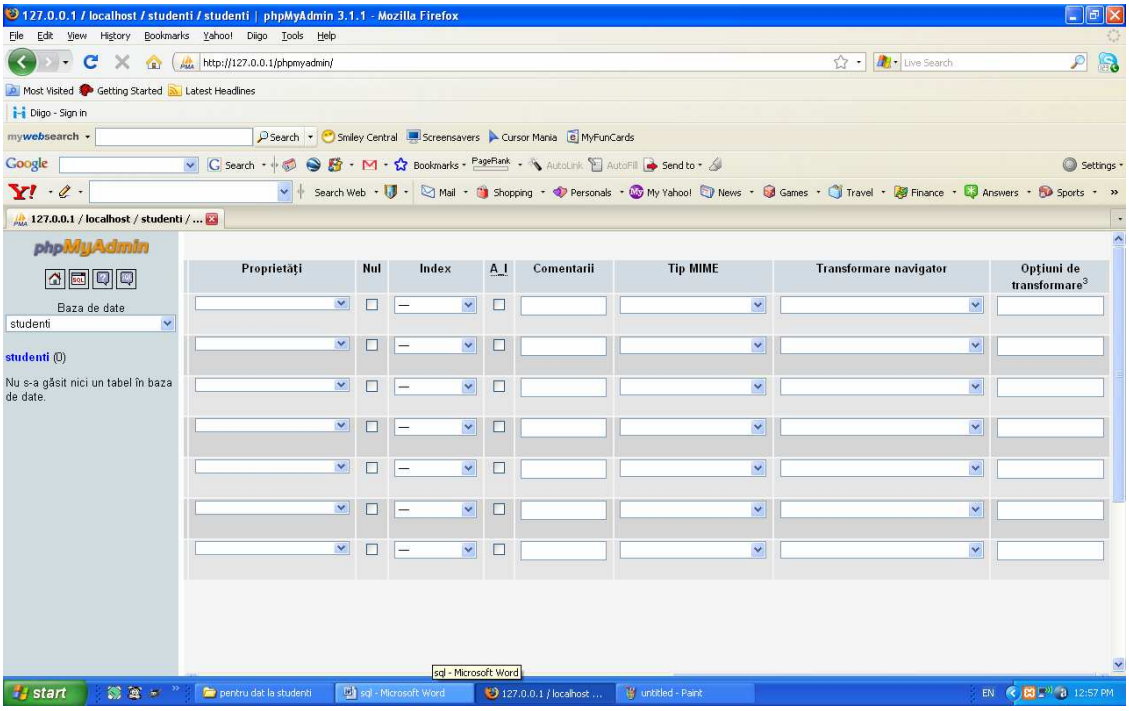


Figura 16. Fereastra de creare a structurii unei tabele (2)

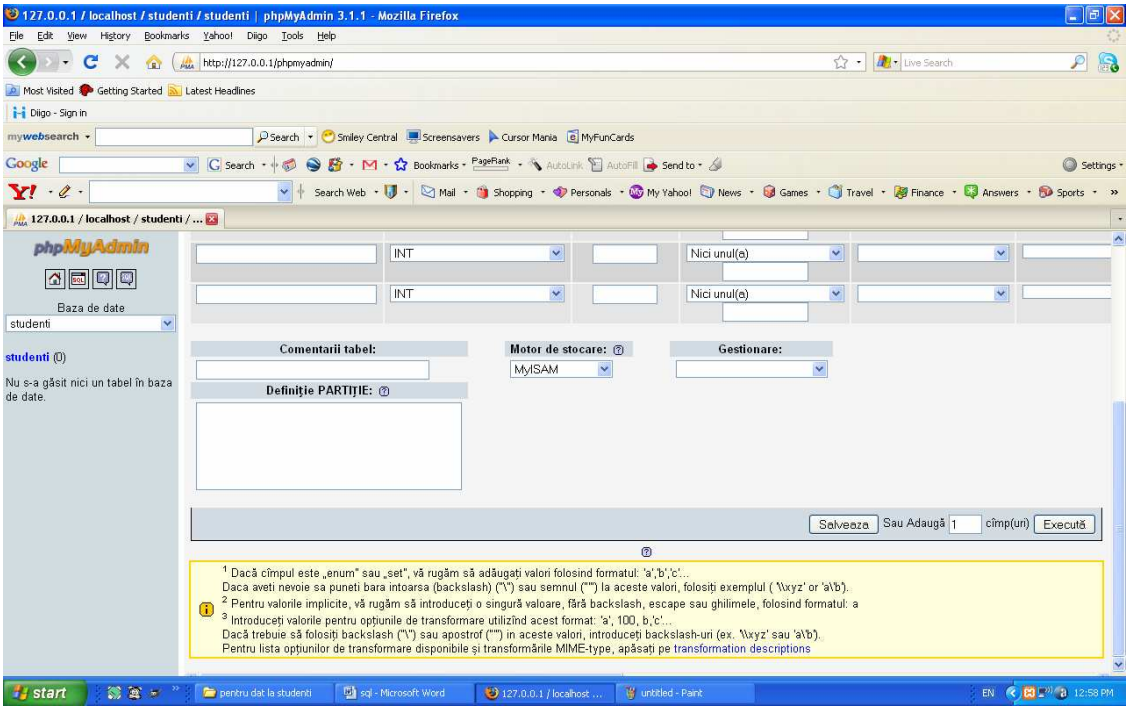


Figura 17. Fereastra de creare a structurii unei tabele (3)

Se vor completa rubricile (cele mai uzuale), conform imaginii din figura 18:

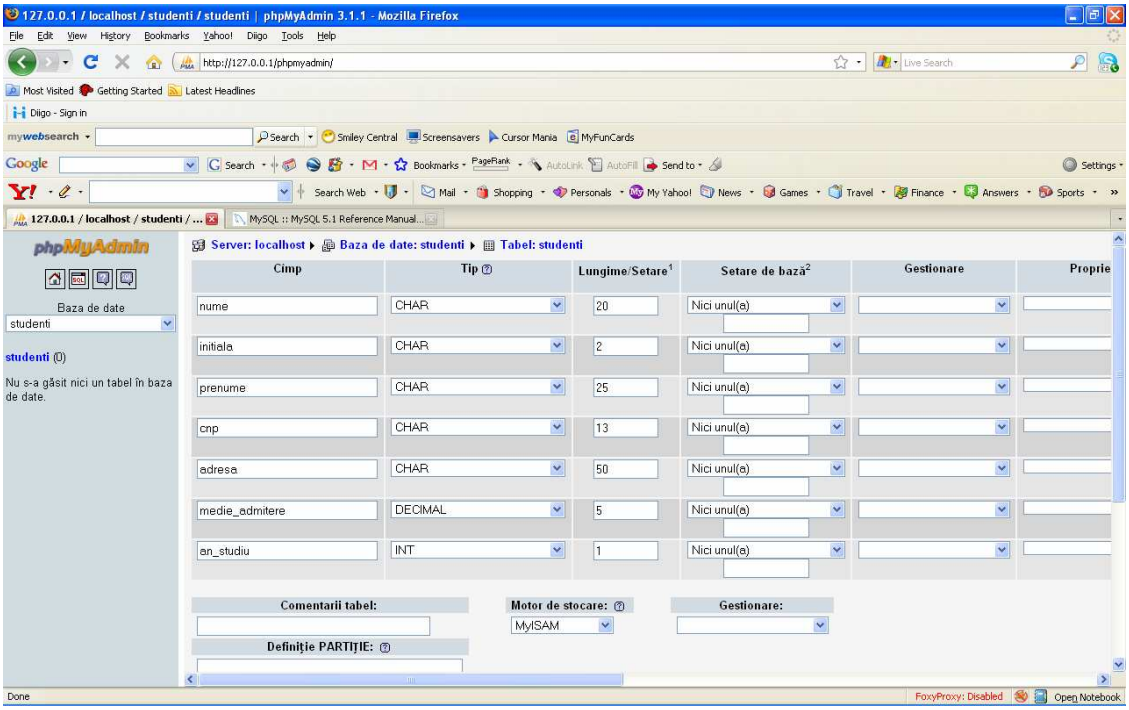


Figura 18. Fereastra de introducere a caracteristicilor câmpurilor tabelei Studenti

După completare, se face click pe butonul Salveaza vizibil în figura 17.

Se deschide fereastra ca în figura 19.

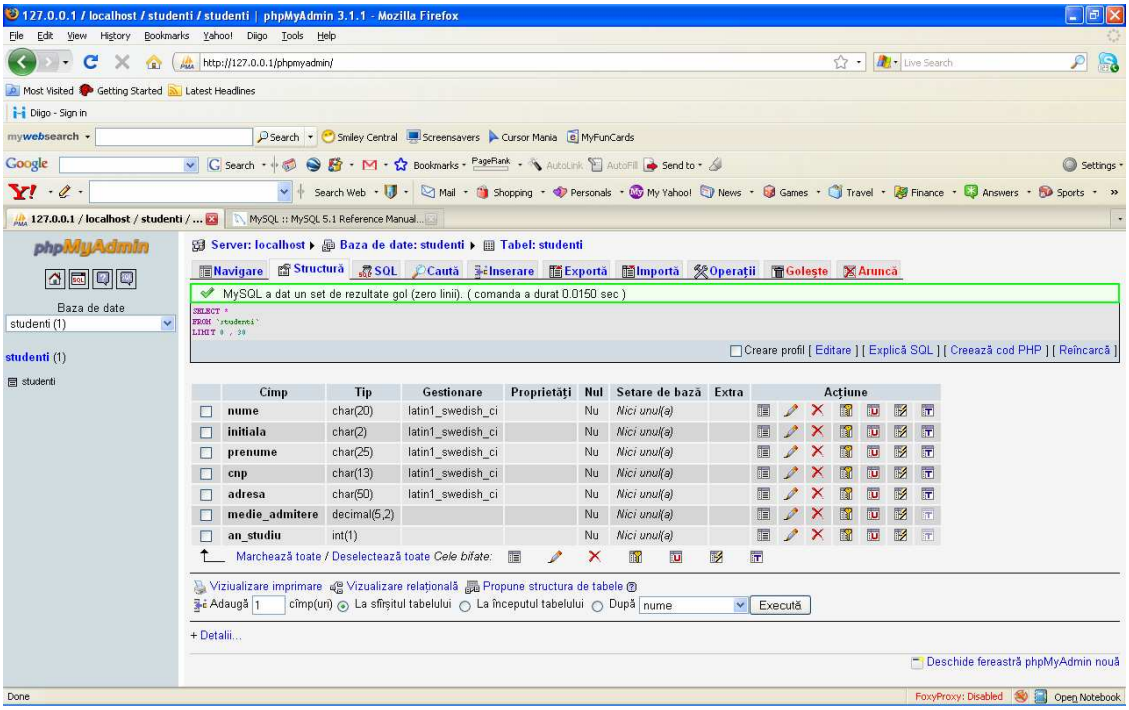


Figura 19. Structura tabelei studenti

Prin selectarea butonului de pe rândul numelui unui câmp, în dreptul coloanelor **Acțiune**, se pot efectua modificări ale câmpurilor respective.

Pentru a introduce şi structurile celorlalte tabele se va selecta opțiunea ca în figura 20.

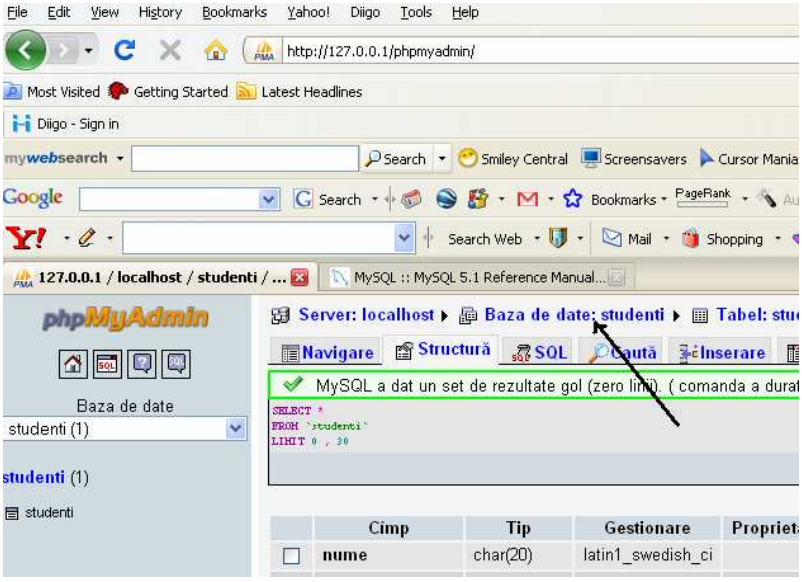


Figura 20. Revenire în fereastra bazei de date studenti

Acțiunea precedentă va deschide fereastra ca în figura 21.

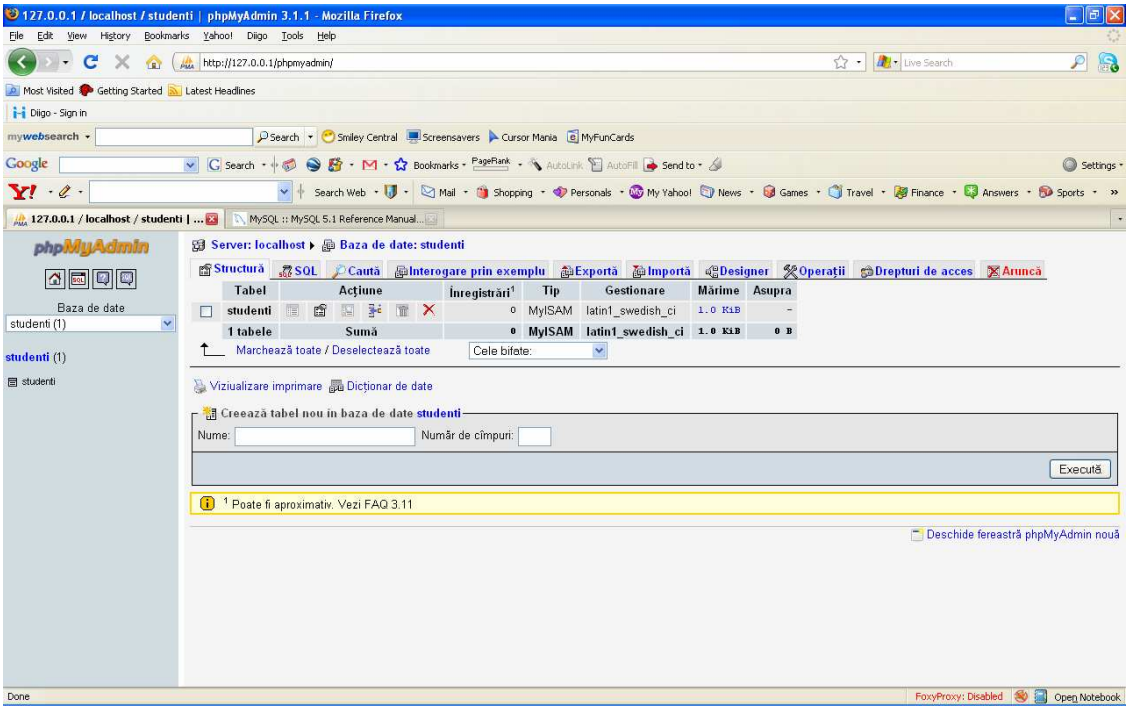


Figura 21. Fereastra Baza de date: studenti

Se vede că avem acces din nou la dialogul care permite crearea structurii unei noi tabele, adică rubricile:

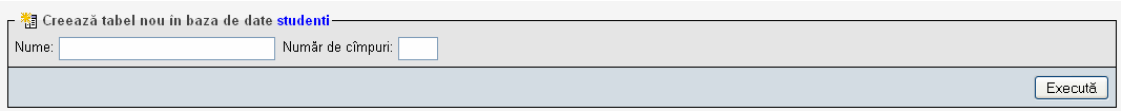


Figura 22. Rubricile ce permit introducerea structurii unei tabele

În mod similar creerii tabelei Studenti, se vor crea şi tabelele Plan_invatamant şi Note.

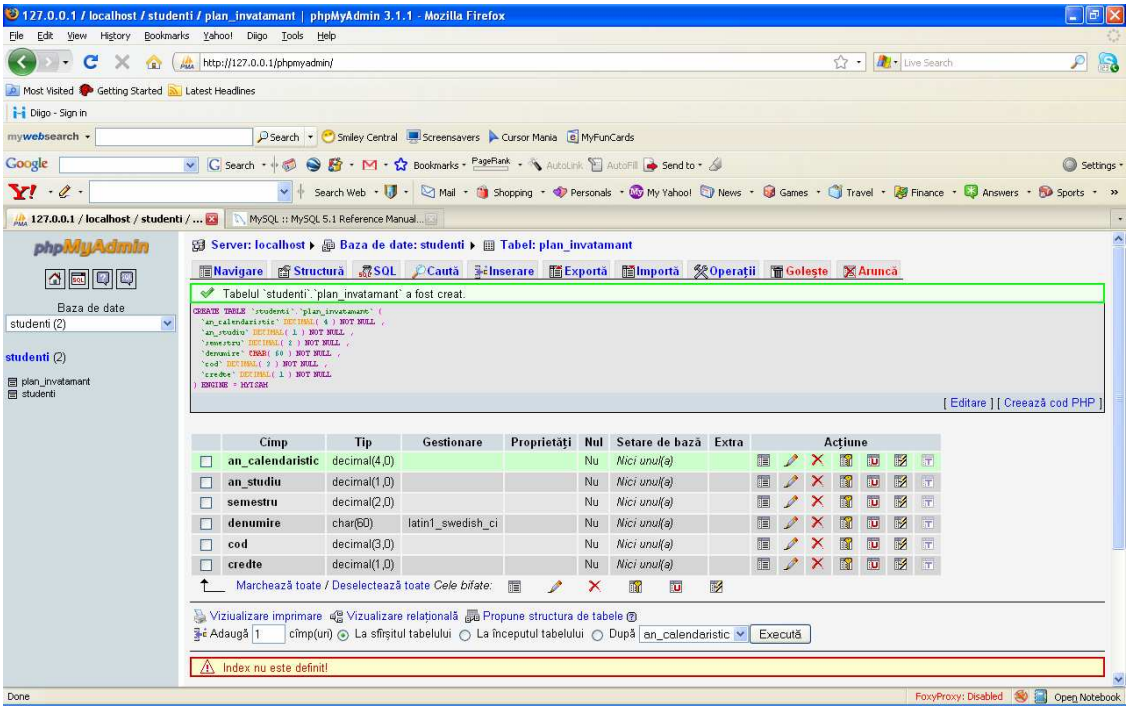


Figura 23. Tabelul plan_invatamant

În figura 23 se poate observa rezultatul creerii structurii tabelei Plan_invatamant ca instrucțiune SQL. Concluzia este următoarea: se poate crea structura unui tabel, prin completarea rubricilor din figurile 15, 16, 17, sau se poate introduce instrucțiunea SQL echivalentă.

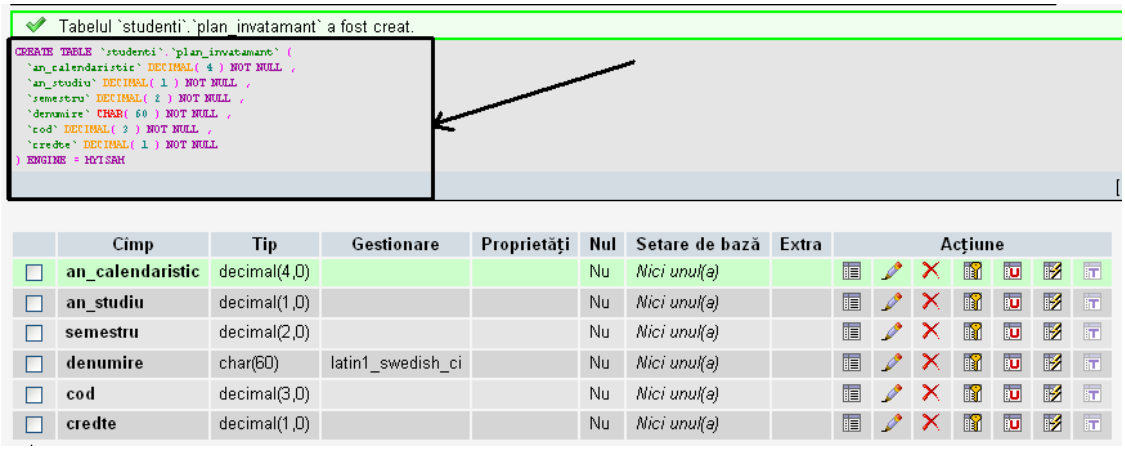


Figura 24. Comanda SQL echivalentă creerii tabelei plan_invatamant

În figura 25 se poate vedea structura tabelei Note

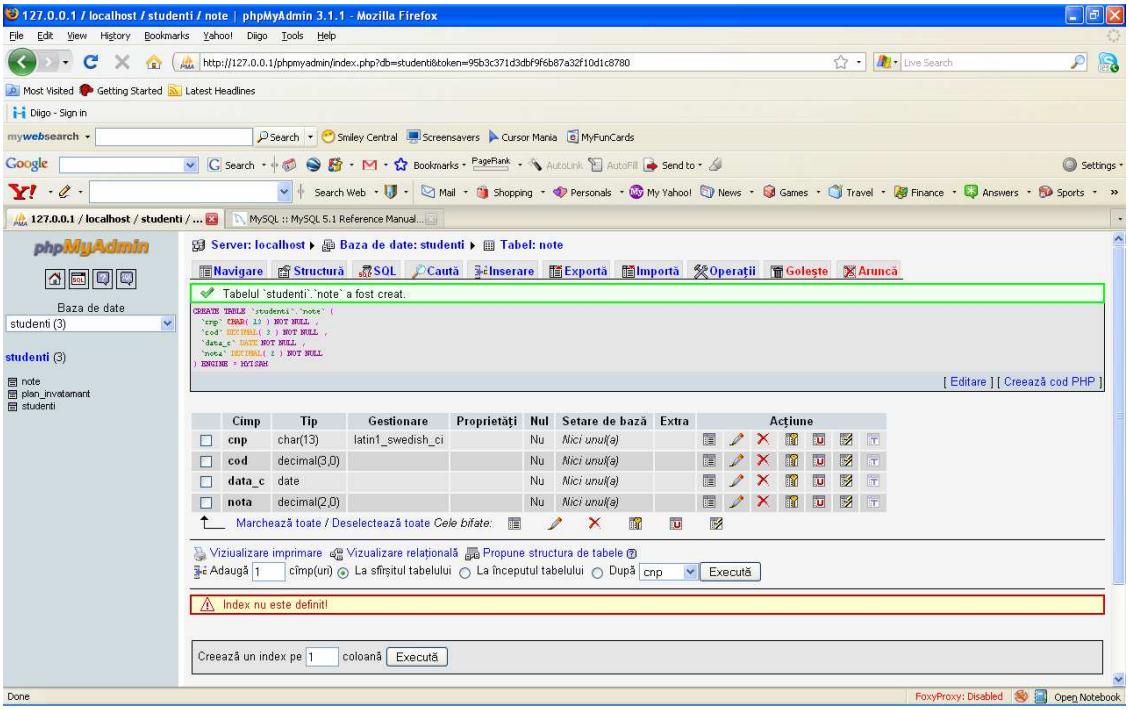


Figura 25. Tabela Note

Pentru a putea introduce date într-o tabelă, de exemplu, în tabela Studenti, se selectează tabela şi apoi se execută click pe tab-ul Inserare, ca în figura 26. În dreptul coloanei Valoare se vor introduce valorile corespunzătoare câmpurilor. După introducere se acționează pe butonul Execută.

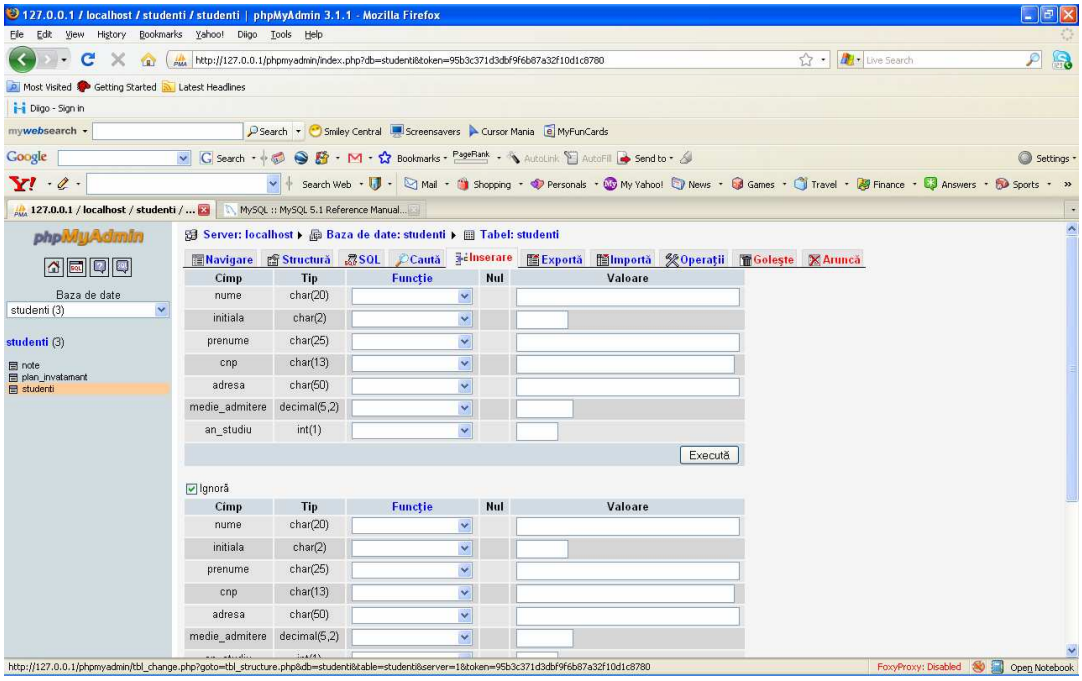


Figura 26. Introducerea de date în tabela Studenti

Pentru a modifica conținutul unei înregistrări se va proceda conform figurii 27.

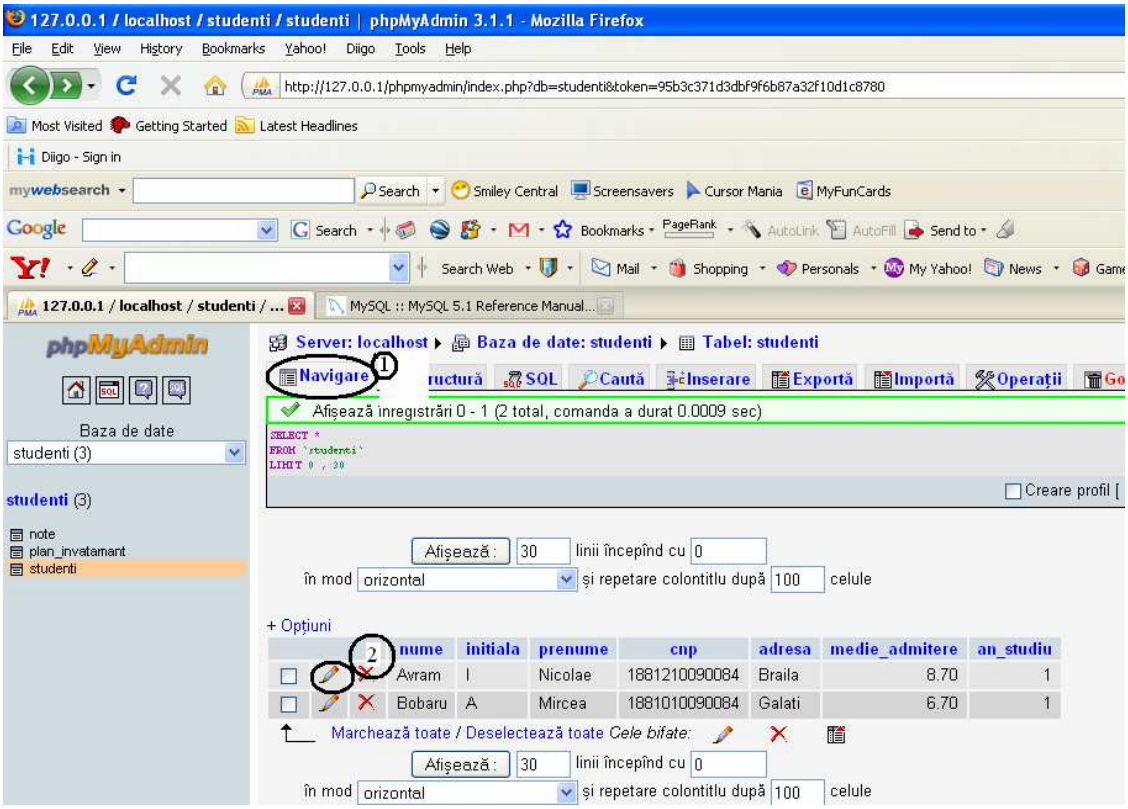


Figura 27. Secvența de acțiuni pentru a intra în modul editare a unei înregistrări

În urma acționării pe imaginile 1 și 2 se va deschide o fereastră ca în figura 28, unde se vor putea modifica orice valoare a câmpurilor afișate.

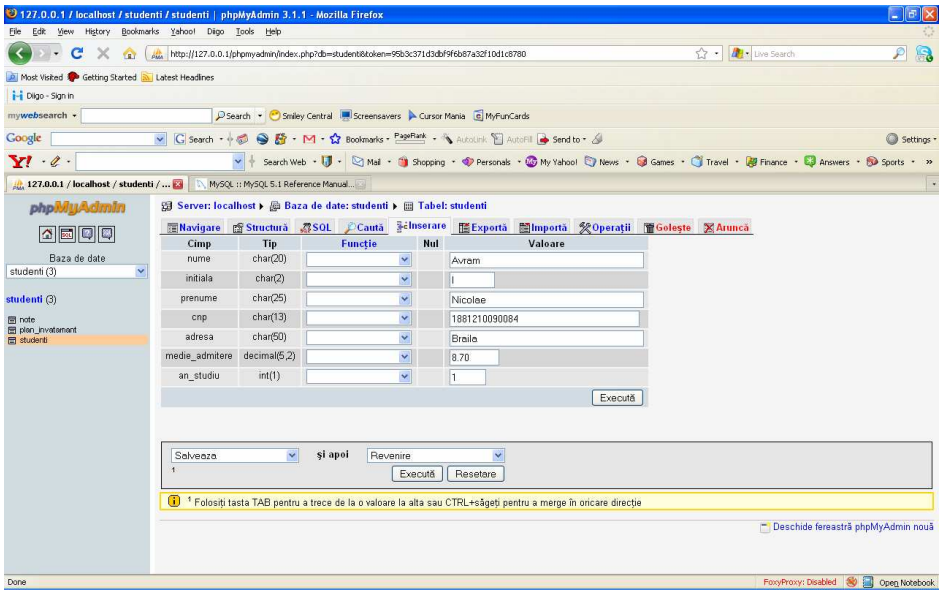


Figura 28. Modificarea coținutului unei înregistrări

Pentru ştergerea unei înregistrări, se va selecta tabela, apoi tab-ul Navigare şi apoi se va executa click pe simbolul 1, de pe rândul corespunzător înregistrării.

Figura 29. Ştergerea unei înregistrări

Pentru siguranţa efectuării operaţiei de ştergere, apare fereastra ca în figura 30, care cere confirmarea intenţiei de ştergere a înregistrării. Se poate observa că sunt afişate în fereastră toate detaliile.

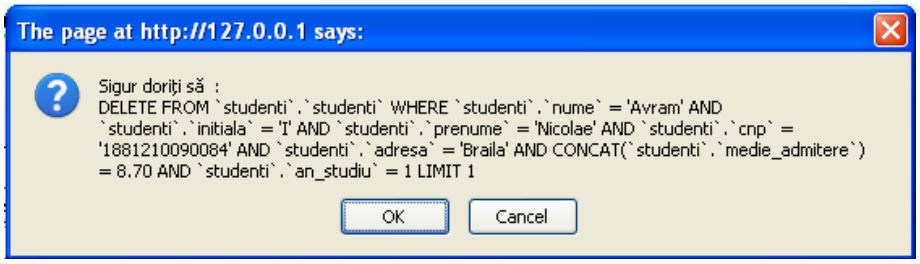


Figura 30. Confirmarea intenţiei de ştergere a unei înregistrări

Consultarea unei baze de date

Consultarea unei baze de date este operaţia care este cea mai importantă. Consultarea unei singure tabele se poate face simplu, acţionând cu mouse-ul, în locaţia 1 sau 2, conform figurii 31.

Figura 31. Consultarea unei tabele

Rezultatul acțiunii solicitate este vizibil în figura 32.

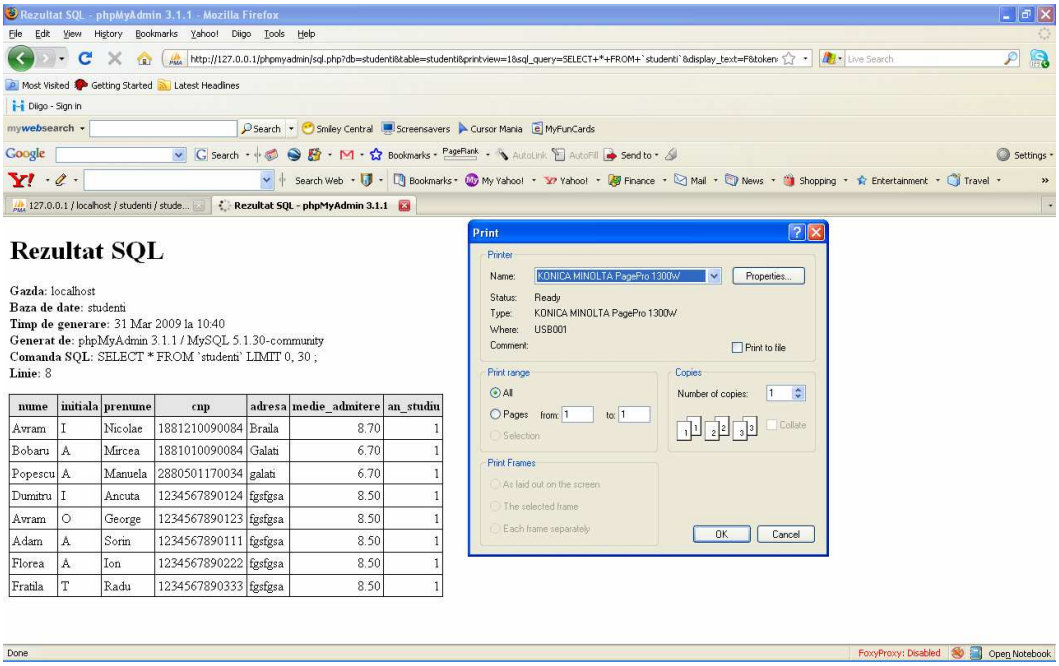


Figura 32. Rezultatul consultării unei tabele, care poate fi listat la imprimantă

De cele mai multe ori este necesar a se consulta date care se află în mai multe tabele relaționate prin intermediul unor coduri. O astfel de situație poate fi, în cazul de față, afișarea mediilor studenților.

Pentru aceasta se va construi o comandă **Select**, ca în secvența următoare:

```
select   studenti.nume, studenti.initiala, studenti.prenume, avg(note.nota) from note left
join studenti
        on note.cnp = studenti.cnp
        group by note.cnp
        order by studenti.nume, studenti.initiala, studenti.prenume
```

Pentru a putea fi executată se va acționa ca în figura 33.

Figura 33. Acțiunile pentru obținerea ferestrei SQL,
care permite introducerea unei comenzi SELECT

Se obține o stare, ca în figura 34.

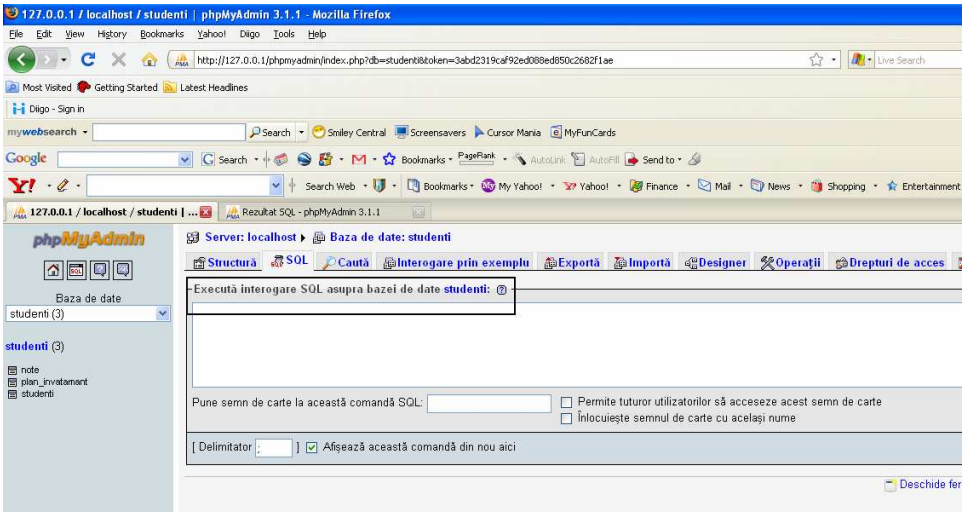


Figura 34. Fereastra SQL, care permite introducerea unei comenzi SQL

Se poate observa cu ușurință, fiind marcată, fereastra “Execută interogare SQL asupra bazei de date studenti:”. Se plasează cursorul în această fereastră și se introduce secvența de mai sus, ca în figura 35. După dactilografierea comenzii, sau frazei, se va acționa pe butonul Execută, cum de altfel, cred, că a devenit familiar.

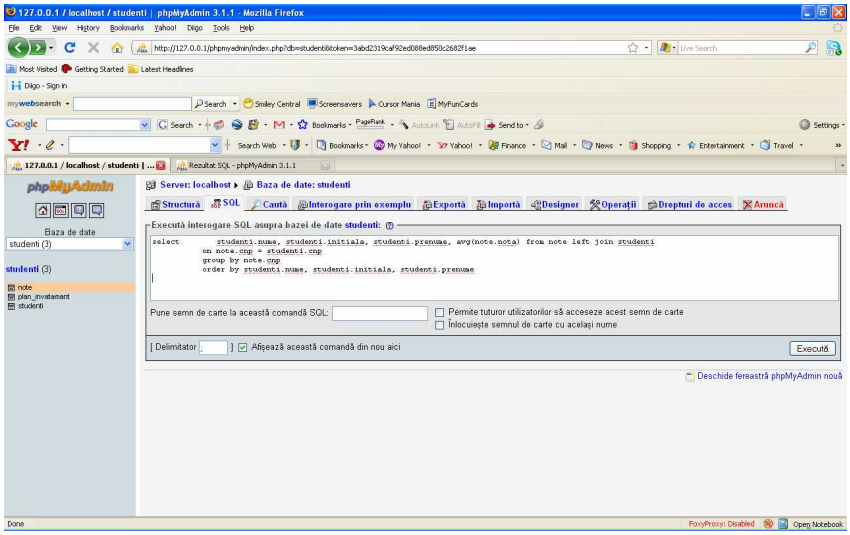


Figura 35. Fraza (comanda) Select pentru interogarea bazei, obținerea mediilor studenților

Rezultatul este prezentat în figura 36, observându-se faptul că rezultatul poate fi şi listat la imprimantă, acționând pe butoanele prezentate în figura 31.

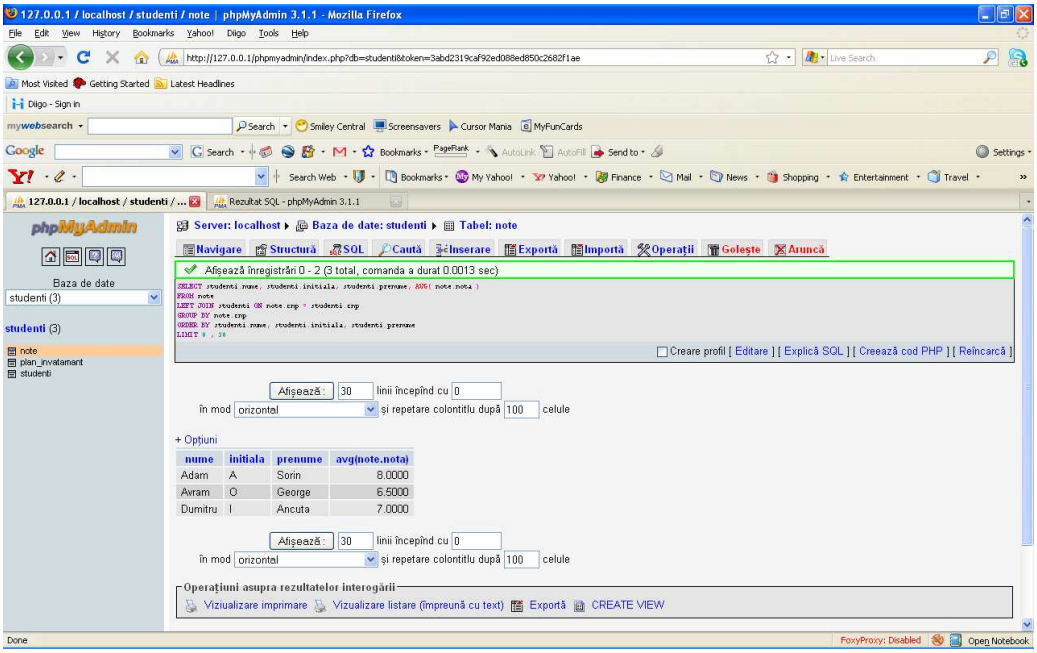


Figura 36. Lista în ordine alfabetică a mediilor studenților

Documentația privind sintaxa comenzii select, pentru “culegerea datelor” din mai multe tabele poate fi găsită la adresa:

http://dev.mysql.com/doc/refman/5.1/en/join.html

Anexă

1. La terminarea creerii tuturor tabelelor bazei de date, conform exemplului, în catalogul bazei de date Studenti se află fişierele conform figurii 37.

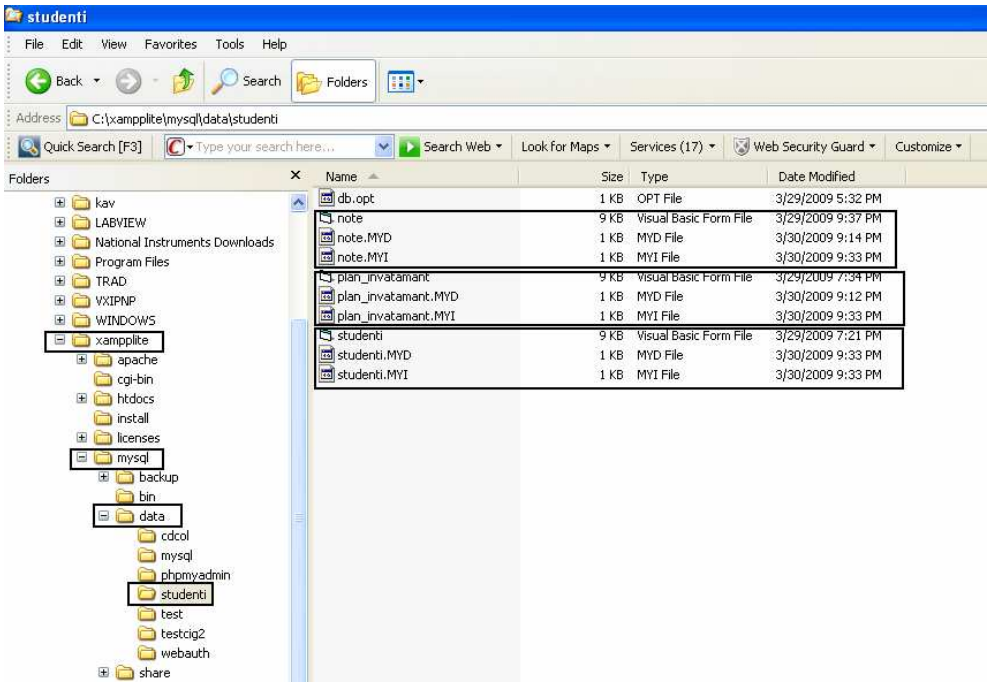


Figura 37. Conținutul catalogului bazei de date

Am marcat locația fişierelor. Am marcat şi cele 3 fişiere corespunzătoare celor trei tabele.

BIBLIOGRAFIE

1. Andone, Ioan - **Baze de date inteligente în managementul firmei** – Iaşi, Editura Dosoftei, 1997;
2. Bâscă, Octavian - **Baze de date** - Bucureşti, Editura All, 1997;
3. Bernhard, E.; Odym, W.; Lechner, F. – **Access : Baze de date** - Bucureşti, Editura All Educational, 2002;
4. Burdescu, Dumitru Dan ş.a. – **Baze de date** – Craiova, Editura Universitaria, 2004;
5. Coord. Florescu, Vasile, ş.a. – **Baze de date : Fundamente teoretice şi practice** – Bucureşti, Editura Infomega, 2002;
6. Florescu, Vasile, ş.a. – **Baze de date** – Bucureşti, Editura Economică, 1999;
7. Fotache, Marin - **Baze de date relaţionale : Organizare, interogare şi normalizare** - Iaşi, Editura Junimea, 1997;
8. Fotache, Marin – **Proiectarea bazelor de date : Normalizare şi postnormalizare : Implementări SQL şi ORACLE**- Iaşi, Editura Polirom, 2005;
9. Gavotă, Mihai – **Baze de date** - Bucureşti, Editura SNSPA – Facultatea de Comunicare şi Relaţii Publice, 2001;
10. Gheorghiu, Ana; Bichiş, Corina Maria – **Baze de date** – Bucureşti, Editura Victor, 2004;
11. Iorga, Marin ş.a. – **Baze de date : Să învăţăm sistemul ORACLE în 28 de lecţii** – Bucureşti, Editura Economică, 2002;
12. Oppel, Andy – **SQL fără mistere** – Bucureşti, Editura Rosetti Educational, 2006;
13. Petersen, John V. - **Baze de date pentru începători** – Bucureşti, Editura Bic All, 2003;
14. Tamaş, Ilie ş.a. – **Limbaje de programare şi baze de date** – Bucureşti, Editura Infomega, 2003
15. Anghel, Traian – **Pagini Web dinamice**- Galaţi, Editura Scorpion, 2002

Cerințele pentru examenul la disciplina Baze de date

Se va alege o temă, la dorința studentului, stabilindu-se un nume aplicației respective.

Se va crea o documentație în care se vor prezenta concret, nu teoretic, principalele etape din proiectarea unei baze de date cu 3-4 tabele. Se va preciza schema bazei de date, structura tabelelor.

La data stabilită pentru examinare se vor prezenta personal: documentația scrisă şi partea practică pe suport magnetic(CD). Examenul va consta în susținerea proiectului: ce şi-a propus să facă, cum s-a rezolvat o anumită problemă, ce se obține, etc. şi întrebări puse de profesorul examinator despre anumite elemente din proiect: modificat obiecte ale proiectului, modificat structuri, interogări ad-hoc la tema aleasă, etc.

OBSERVAȚII

1. Luați exemplul din suportul de curs ca exemplu. Nu faceți un proiect care să fie similar celui prezentat de autor.

2. Nu se acceptă proiecte cu un grad de asemănare între ele mai mare de 50%.

Pentru lămuriri suplimentare, vă invit să corespondăm prin e-mail, adresa mea fiind:

panfiloiu.gheorghe@univ-danubius.ro