Programa Curs Baze de Date

Lecţia 1 – Noţiuni teoretice

Dată, Informaţie, Bază de date, Sistem de gestiune a bazelor de date, Exemple de SGBD-uri, Tabelă, Entitate, Atribut, Cheie, Constrângeri de integritate

Elemente generale de algebră relaţională (noţiuni de bază de teoria mulţimilor: reunine, intersecţie, diferenţă, produs cartezian, etc.)

Normalizarea bazelor de date, Definiţie, Prezentarea formelor normale

Lecţia 2 – Design-ul bazei de date

Proiectarea unei baze de date; Relaţii între tabele; Tipuri de relaţii între tabele; Exemple; Prezentarea mediului de lucru pentru utilizarea MySQL (HeidiSQL); Limbajul SQL - Introducere

Lecţia 3 – Limbajul de descriere a datelor (LDD)

Crearea şi ştergerea unei baze de date

Crearea şi ştergerea unei tabele; Modificarea structurii tabelelor

Tipuri de date MySQL; Modificarea câmpurilor; Chei primare, chei externe

Lecţia 4 – Limbajul de manipulare a datelor (LMD)

Introducerea înregistrărilor în tabele (instrucţiunea INSERT)

Modificarea înregistrărilor din tabele (instrucţiunea UPDATE)

Ştergerea înregistrărilor din tabele (instrucţiunea DELETE)

Extragerea informaţiilor din tabele (instrucţiunea SELECT)

Clauzele instrucţiunii SELECT (WHERE, GROUP BY, HAVING, ORDER BY, LIMIT)

Lecţia 5 – Operatori şi funcţii MySQL

Operatori: aritmetici, logici, de comparare;

Funcţii predefinite MySQL: matematice, de compararare, pentru date calendaristice, pentru şiruri de caractere, etc.

Lecţia 6 – Join-uri şi Reuniuni

Noţiuni teoretice; Tipuri de join-uri; Reuniuni

Lecţia 7 – Subinterogări, View-uri

Tipuri de subinterogări; Definirea şi utilizarea view-urilor

Lecţia 8 – Proceduri şi funcţii, Triggere, Tranzacţii

Definire, apelare, parametrii unei proceduri

Conceptul de trigger; Definire și referirea la valorile stocate de un trigger

Tranzacţii

Introducere

Aplicaţiile informatice utilizate în prezent lucrează cu un număr foarte mare de date care trebuie stocate în aşa fel încât să le putem accesa rapid şi uşor. Astfel, majoritatea aplicaţiilor, de la site-uri şi alte aplicaţii web până la aplicaţii bancare sau de gestiune a clienţilor, folosesc baze *de date relaţionale.*

În acest curs de **Fundamente Baze de date MySQL** ne propunem să parcurgem câteva noţiuni teoretice fundamentale pentru înţelegerea conceptelor folosite în lucrul cu baze de date relaţionale, concepte tot mai des întâlnite în limbajul informatic curent (*informaţie*, *date*, *baze de date*, etc.) dar şi să trecem în revistă etapele care sunt parcurse în realizarea aplicaţiilor informatice care folosesc baze de date (de la proiectarea unei baze de date până la interogări avansate asupra bazei de date).

Cursul se adresează persoanelor fără experienţă şi cunoştinţe în domeniul bazelor de date, dar şi persoanelor care au cunoştinţe şi o minimă experienţă în lucrul cu baze de date.

Noţiunile teoretice fundamentale despre bazele de date relaţionale, precum şi elementele limbajului de interogare **SQL**, sunt introduse pas cu pas şi sunt însoţite de exemplificări şi utilizări practice.

Am ales ca mediu de dezvoltare a aplicaţiilor ce vor fi realizate în acest curs sistemul de gestiune a bazelor de date **MySQL**.

Acest sistem de gestiune a bazelor de date (**SGBD**), **MySQL**, este foarte cunoscut datorită utilizării sale în aplicaţiile şi site-urile web împreună cu limbajul de programare PHP.

Aplicaţii ale bazelor de date pe Web

Orice site care are un modul de creare cont şi login, orice magazin online, site de ştiri, blog, etc. are o bază de date în care este ţinută informaţia în mod structurat. Cea mai mare parte a site-urilor de pe Internet care trec de nivelul de site de prezentare folosesc baze de date.

Astfel o persoană care doreşte să lucreze în domeniul programării web, pe lângă cunoştinţele de HTML şi CSS folosite în partea de dezvoltare a aplicaţiei ce interacţionează cu utilizatorul, şi cele de programare PHP, are nevoie şi de cunoştinţe de baze de date relaţionale, întrucât aproape toate site-urile şi toate aplicaţiile web conţin informaţia în baze de date relaţionale.

Jobul de programator web este unul foarte interesant şi există o cerere mare pe piaţă pentru persoane care au cunoştinţe de programare web. Este un job provocator, pentru că fiecare proiect aduce ceva nou, la fiecare proiect se pot învăţa lucruri suplimentare şi aduce şi satisfacţia că produsul realizat este vizualizat de foarte mulţi oameni (ne referim aici la site-urile web).

Tipologia site-urilor web porneşte de la site-uri de prezentare şi continuă cu magazine online, bloguri, ajungând până la site-uri complexe (portaluri).

În cadrul acestor tipologii avem site-uri web statice, realizate doar în HTML sau site-uri web dinamice cu informaţia introdusă dintr-o secţiune de administrare, în acest sens folosindu-se un limbaj de programare şi baze de date relaţionale.

Iată modul de funcţionare şi de interacţiune cu serverul web şi cu serverul de baze de date MySQL la realizarea unui site web:

Browser-ul web interpretează doar cod HTML. Astfel că, dacă avem un site cu pagini statice, unde nu folosim nici un limbaj de programare, ci doar limbajul de marcare HTML, iar fişierele ce conţin aceste pagini au extensia .html sau .htm nu avem nevoie să instalăm altceva pe calculatorul nostru pentru a putea deschide acele pagini.

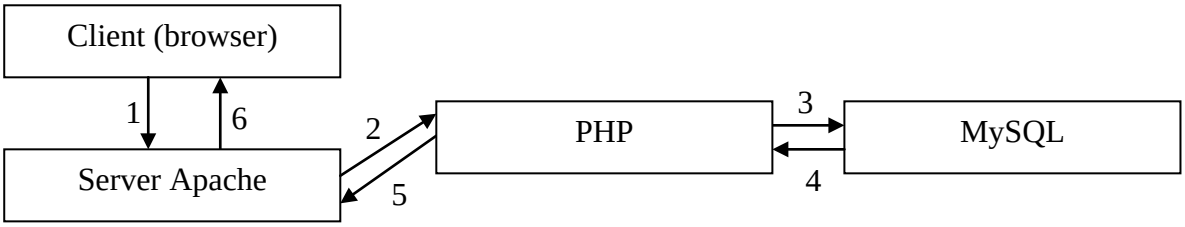
În schimb, dacă vom realiza un site web dinamic folosind limabjul PHP şi baze de date **MySQL**, avem nevoie de instalarea pe calculator a unui server.

PHP este un limbaj care funcţionează pe partea de server, în timp ce HTML este un limbaj pe partea de client.

Codul PHP este transmis către serverul de Apache, acesta interpretează codul primit şi generează cod HTML pe care îl transmite către browser astfel că browserul web primeşte tot cod HTML, singurul pe care ştie să îl interpreteze şi să îl afişeze.

De aceea, dacă vizualizăm codul sursă al unui site realizat în PHP vom vedea că el este transformat în cod HTML şi astfel este afişat în sursa paginii. Deci, acces la codul PHP nu putem avea, pentru a vedea cum a fost readactat codul, decât dacă accesăm fişierele .php de pe server.

În schema de mai jos vom reprezenta modul în care interacţionează PHP-ul cu serverul Apache şi cu browserul web:



Primul pas (1) este reprezentat de cererea pe care clientul (browser-ul web) o adresează serverului în momentul în care se accesează o pagină PHP printr-un URL.

Severul trimite pagina spre procesare interpretorului PHP (2).

Dacă avem şi instrucţiuni MySQL, din PHP se face conectarea la baza de date MySQL şi se trimite cererea către serverul MySQL (3).

Serverul MySQL execută instrucţiunile specifice şi returnează rezultatele către PHP (4).

Interpretorul PHP returnează aceste rezultate către serverul Apache (5).

Serverul Apache returnează clientului cod HTML pe care acesta ştie să îl interpreteze şi să îl afişeze (6).

Cursul nu se ocupă de realizarea site-urilor web dar am prezentat acest mod de interacţiune dintre client – server – PHP – server MySQL pentru a înţelege modul în care sunt folosite bazele de date şi în realizarea site-urilor web.

Aplicaţii ale bazelor de date în alte domenii

O marte parte din aplicaţiile software existente folosesc baze de date pentru stocarea şi extragerea informaţiilor. Aceste baze de date se interoghează ulterior pentru obţinerea de diverse statistici. De exemplu, informaţiile despre clienţii şi facturile unei companii se ţin într-o bază de date.

Obiective

Pe parcursul acestui curs de iniţiere în baze de date vom parcurge noţiunile fundamentale teoretice şi vom învăţa cum se creează/şterg tabele în **MySQL**, cum se inserează/modifică/şterg înregistrări într-o tabelă din baza de date, precum şi diverse interogări de regăsire a datelor din tabele pornind de la cele simple şi ajungând la cele mai complexe. De asemenea, vom învăţa să lucrăm cu **vederi** (**view**-uri), **join**-uri, **reuniuni**, **proceduri** şi **funcţii** (**rutine**), **triggere** şi **tranzacţii**.

Obiectivele cursului **Fundamente Baze de Date MySQL** sunt următoarele:

- să înţelegeţi ce noţiunile teoretice folosite în domeniul bazelor de date (dată, informaţie, bază de date, sistem de gestiune a bazelor de date, tabelă, atribut, înregistrare, etc.);
- să înţelegeţi principiile generale ale algebrei relaţionale, să cunoaşteţi operaţiile de reuniune, intersecţie, diferenţă, produs cartezian folosite în teoria mulţimilor;
- să înţelegeţi principiul normalizării bazelor de date;
- să cunoaşteţi şi să puteţi instala mediile de lucru folosite pentru lucrul cu baze de date **MySQL** (serverul **WAMP** şi **HeidiSQ/MySQL Workbench**);
- să înţelegeţi noţiunea de relaţie între tabele şi să cunoaşteţi tipurile de relaţii între tabele;
- să puteţi proiecta o bază de date;
- să cunoaşteţi sintaxa **SQL** şi instrucţiunile de bază (**INSERT**, **UPDATE**, **DELETE**, **SELECT**);
- să cunoaşteţi limbajul de descriere a datelor (**LDD**), limbajul de manipulare a datelor (**LMD**), precum şi **tipuri de date** şi **operatori MySQL**;
- să cunoaşteţi câteva **funcţii predefinite MySQL** (pentru lucrul cu şiruri de caractere, cu date calendaristice, funcţii matematice, etc.);
- să înţelegeţi noţiunile de **join**, **union** şi **subinterogări** şi să le folosiţi în interogări complexe;
- să înţelegeţi utilizarea **vederilor** (**tabele virtuale**) şi modul de folosire precum şi noţiunile de **trigger** şi **tranzacţie**;
- să cunoaşteţi cum se creează **procedurile** şi **funcţiile** în **MySQL**;

- să puteţi realiza aplicaţii de complexitate medie cu baze de date **MySQL** (de la proiectarea bazei de date până la proceduri şi funcţii aplicate pe baza de date).

Pe parcursul acestui curs, pe măsură ce învăţăm noţiunile **MySQL**, vom lucra şi la realizarea unui proiect mai complex, este vorba despre o bază de date care asigură gestiunea unui anumit domeniu (de exemplu o bază de date pentru gestiunea cărţilor dintr-o bibliotecă sau o bază de date folosită la gestiunea angajaţilor unei companii, o bază de date folsoită la gestiunea pacienţilor şi medicilor dintr-un spital, etc.).

La finalul cursului, pentru a obţine diploma finală, înainte de a susţine testul final ce va conţine întrebări de tip grilă, trebuie să prezentaţi o aplicaţie complexă în care să fie folosite noţiunile învăţate.

Prezentare lecţii

Iată în continuare o descriere succintă a lecţiilor ce vor fi parcurse în cadrul cursului de **Fundamente Baze de Date MySQL**.

În prima parte a cursului sunt prezentate pe larg noţiuni teoretice precum şi câteva noţiuni fundamentale de algebră relaţională. Înţelegerea conceptului de normalizare precum şi prezentarea formelor normale reprezintă următorul subiect abordat în partea de început.

Cea de-a doua lecţie se axează pe prezentarea concretă a mediului de lucru (**HeidiSQL/MySQL Workbench**) şi pe prezentarea modului în care este proiectată o bază de date. De asemenea, vom aborda subiectul relaţionării – concept fundamental în bazele de date relaţionale – şi vom explica şi exemplifica tipurile de relaţii ce pot exista între tabelele unei baze de date. Continuăm această lecţie cu o introducere în limbajul **SQL** (**S**tructured **Q**uery **L**anguage).

În lecţia numărul trei, **L**imbajul de **D**escriere a **D**atelor (**LDD**) sau, în limba engleză **D**ata **D**escription **L**anguage (**DDL**), vom prezenta şi vom exemplifica instrucţiunile de creare şi ştergere a unei baze de date, creare şi ştergere a unei tabele dintr-o bază de date precum şi instrucţiunile de modificare a structurii tabelelor.

În cea de-a patra lecţie prezentăm **L**imbajul de **M**anipulare a **D**atelor (**LMD**), în limba engleză **D**ata **M**anipulation **L**anguage (**DML**). Limbajul de Manipulare a Datelor se referă la cele 4 instrucţiuni fundamentale ale limbajului **SQL**:

- **INSERT** – pentru introducerea înregistrărilor într-o tabelă;
- **UPDATE** – pentru modificarea înregistrărilor din tabele;
- **DELETE** – pentru ştergerea înregistrărilor din tabele;
- **SELECT** – instrucţiunea de regăsire a informaţiilor din baza de date.

Este prezentată sintaxa fiecărei instrucţiuni în parte. De asemenea, în cazul instrucţiunii **SELECT** sunt prezentate şi explicate clauzele ce pot să apară în cadrul instrucţiunii.

Lecţia următoare prezintă noţiuni despre **operatorii** şi **funcţiile predefinite** ale **MySQL**. Este vorba de operatorii aritmetici, logici şi de comparare şi de funcţii matematice, funcţii pentru lucrul cu şiruri de caractere, cu date calendaristice, etc.

Cursul continuă cu noţiuni teoretice despre **JOIN**-uri, tipuri de **JOIN**-uri şi **reuniuni**.

Un alt capitol important al acestui curs este cel în care sunt prezentate **subinterogările** şi **vederile** (**view**-uri sau **tabele virtuale**).

În final avem capitolul dedicat **Procedurilor şi Funcţiilor** în **MySQL**. Procedurile şi funcţiile mai poartă şi numele de **rutine MySQL** sau de **proceduri stocate**. Această ultimă denumire provine de la faptul că aceste **rutine** sunt stocate pe server după ce au fost create. De asemenea, introducem conceptul de **trigger** şi prezentăm noţiuni generale despre rolul **tranzacţiilor** şi modul de folosire al acestora.

Lecţia 1 – Noţiuni teoretice

1.1 Date, informaţii, cunoştinţe

Auzim adesea folosindu-se termenii „societate informaţională", „tehnologia informaţiei" însă de multe ori cuvântul „informaţie" este folosit fără a înţelege clar sensul acestui cuvânt şi faptul că este o diferenţă între **date**, **informaţii**, **cunoştinţe** şi **obiecte**.

În general, conţinutul gândirii umane operează cu următoarele concepte:

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

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

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

Obiectele reprezintă cunoştinţe pentru care se ştie comportamentul şi proprietăţile, referitoare la o entitate din lumea reală.

1.2 Bază de date

O *bază de date* (**BD**) conţine toate informaţiile necesare despre obiectele ce intervin într-o mulţime de aplicaţii, relaţiile logice între aceste informaţii şi tehnicile de prelucrare corespunzătoare.

O *bază de date* reprezintă o colecţie de date organizate ce pot fi accesate simultan de mai mulţi utilizatori. Prelucrarea datelor se referă la operaţiile de *introducere*, *ştergere*, *actualizare* şi *interogare* a datelor. O *bază de date* este o colecţie de înregistrări sau de informaţii introduse şi stocate într-un calculator într-un mod sistematic (structurat).

O *bază de date* poate fi interogată (întrebată) de către noi sau de către un program prin intermediul unui limbaj relativ simplu (în general **SQL**) şi răspunde cu informaţie, în funcţie de care se iau decizii. Pentru valorificarea informaţiei ce poate fi extrasă dintr-o bază de date, este esenţial modul în care organizăm şi stocăm datele într-o bază de date.

1.3 Sistem de gestiune a bazelor de date (Database Management System)

Un *sistem de gestiune a bazelor de date* (**SGBD**) reprezintă un sistem de programe care permite construirea unor baze de date, introducerea informaţiilor în bazele de date şi dezvoltarea de aplicaţii privind bazele de date.

Cu alte cuvinte, aplicaţia care este folosită pentru a realiza, a administra şi a interoga o bază de date este numită sistemul de gestiune sau de management al bazei de date (**SGBD**). În limba engleză denumirea pentru sistemul de management al bazei de date este **Database Management System** (**DBMS**).

Printr-un **SGBD** se realizează interacţiunea utilizatorului cu baza de date.

Orice **SGBD** conţine, printre alte componente un limbaj de descriere a datelor (**LDD**) care permite descrierea structurii bazei de date, a fiecărei componente a ei, a relaţiilor dintre componente, a drepturilor de acces ale utilizatorilor la baza de date, a restricţiilor în reprezentarea informaţiilor.

O altă componentă a unui **SGBD** este limbajul de manipulare a datelor (**LMD**) ce permite operaţii asupra datelor aflate în baza de date, cum ar fi: inserarea unui element, ştergerea unui element, modificarea unui element, căutarea (regăsirea) unor elemente.

Teoria relaţională, foarte bine pusă la punct într-un domeniu de cercetare distinct, a dat o fundamentare solidă realizării de **SGBD**-uri performante.

La sfârşitul anilor ‘80 şi apoi în anii ‘90 au apărut, în special o dată cu pătrunderea în masă a microcalculatoarelor, numeroase sisteme de gestiune bazelor de date relaţionale (**SGBDR**-uri).

Aceasta a însemnat o evoluţie de la **SGBD**-urile de generaţia întâi (arborescente şi reţea) spre cele de generaţia a doua (relaţionale). Această evoluţie s-a materializat, în principal în: oferirea de limbaje de interogare neprocedurale, îmbunătăţirea integrităţii şi securităţii datelor, optimizarea şi simplificarea acceselor.

Teoria relaţională este un ansamblu de concepte, metode şi instrumente care a dat o fundamentare riguroasă realizării de **SGBDR** performante.

SGBD relaţional este un ansamblu de produse software complex şi complet, care implementează modelul logic de date relaţional, precum şi cel puţin un limbaj de programare relaţional.

Elementele necesare evaluării unui **SGBDR** sunt prezentate prin regulile lui Codd.

1.4 Regulile lui Codd

Edgar F. Codd (cercetător la IBM) a formulat 13 reguli care exprimă cerinţele maximale pentru ca un **SGBD** să fie relaţional.

Regulile sunt utile pentru evoluarea performanţelor unui **SGBDR**. Acestea sunt:

R_0. *Gestionarea datelor se face la nivel de relaţie*: limbajele utilizate trebuie să lucreze cu relaţii (tabele). Relaţia trebuie să fie unitatea de informaţie pentru operaţii.

R_1. *Reprezentarea logică a datelor*: toate informaţiile din **BDR** trebuie stocate şi prelucrate ca relaţii (tabele).

R_2. *Garantarea accesului la date*: **LMD** trebuie să permită accesul la fiecare valoare atomică din **BDR** (tabelă, coloană, cheile de diferite tipuri).

R_3. *Valoarea NULL*: trebuie să se permită mai întâi declararea şi apoi prelucrarea valorii de tip NULL ca date lipsă sau inaplicabile. Deoarece valoarea NULL înseamnă date lipsă ea nu poate fi prelucrată în expresii aritmetice, dar se pot face alte prelucrări, se pot aplica alţi operatori (de exemplu, operatorul existenţial IS şi operatorul logic NOT).

R_4. *Metadatele*: informaţiile despre descrierea **BDR** se stochează în dicţionar ca tabele, la fel ca datele propiu-zise.

R_5. *Limbajele utilizate*: **SGBDR** trebuie să permită utilizarea mai multor limbaje, dintre care cel puţin unul să permită definirea tabelelor (de bază şi virtuale), definirea restricţiilor de integritate (constrângeri), manipularea datelor, autorizarea accesului, tratarea tranzacţiilor.

R_6. *Actualizarea tabelelor virtuale*: trebuie să se permită ca tabelele virtuale (view-uri) să fie şi efectiv actualizabile, nu numai teoretic actualizabile.

R_7. *Actualizările în baza de date*: manipularea unei tabele trebuie să se facă prin operaţii de regăsire (interogare) dar şi de actualizare.

R_8. *Independenţa fizică a datelor*: schimbarea stucturii fizice a datelor (modul de reprezentare (organizare) şi modul de acces) nu afectează programele.

R_9. *Independenţa logică a datelor*: schimbarea structurii de date (logice) a tabelelor nu afectează programele.

R_{10}. *Restricţiile de integritate*: acestea trebuie să fie definite prin **LDD** şi stocate în dicţionarul (catalogul) **BDR**.

R_{11}. *Distribuirea geografică a datelor*: **LMD** trebuie să permită ca programele de aplicaţie să fie aceleaşi atât pentru date distribuite cât şi petru date centralizate (alocarea şi localizarea datelor vor fi în sarcina **SGBDR**-ului).

R_{12}. *Prelucrarea datelor la nivel de bază (scăzut)*: dacă **SGBDR** posedă un limbaj de nivel scăzut (prelucrarea datelor se face la nivel de înregistrare), acesta nu trebuie utilizat pentru a evita restricţiile de integritate.

Regulile lui Codd sunt greu de indeplinit în totalitate de către **SGBDR**. Pornind de la cele 13 reguli de mai sus, au fost formulate o serie de criterii (cerinţe) pe care trebuie să le îndeplinească un **SGBD** pentru a putea fi considerat relaţional într-un anumit grad. S-a ajuns astfel, la mai multe grade de relaţional pentru **SGBDR**: cu interfaţă relaţională (toate datele se reprezintă în tabele, există operatorii de selecţie, proiecţie şi joncţiune doar pentru interogare), pseudorelaţional (toate datele se reprezintă în tabele, există operatorii de selecţie, proiecţie şi joncţiune fără limitări), minimal relaţional (este pseudorelaţional şi în plus, operaţiile cu tabele nu fac apel la pointeri observabili de utilizatori), complet relaţional (este minimal relaţional şi în plus, există operatorii de

reuniune, intersecţie şi diferenţă, precum şi restricţiile de integritate privind unicitatea cheii şi restricţia referenţială).

Cele 13 reguli ale lui Codd pot fi grupate în cinci categorii:

- reguli de bază (fundamentale): R_0 şi R_{12};
- reguli structurale: R_1 şi R_6;
- reguli pentru integritatea datelor: R_3 şi R_{10};
- reguli pentru manipularea datelor: R_2, R_4, R_5 şi R_7;
- reguli pentru independenţa datelor: R_8, R_9 şi R_{11}.

În concluzie, **SGBDR** este un sistem software complet care implementeză modelul de date relaţional şi respectă cerinţele impuse de acest model. El este o interfaţă între utilizatori şi baza de date.

1.5 Exemple de SGBD-uri

Oracle. Este realizat de firma Oracle Corporation USA. Sistemul este complet relaţional, robust, se bazează pe **SQL** standard extins. Arhitectura sistemului este client/server, permţând lucrul, cu obiecte şi distribuit. Sistemul respectă teoria relaţională, este robust şi se bazează pe **SQL** standard. Permite lucrul distribuit şi are modul de optimizare a regăsirii.

SQL Server. Este realizat de firma Microsoft. Se bazează pe **SQL** şi rulează în arhitectura client/server.

MySQL. Este un **SGBD** produs de compania suedeză MySQL AB şi distribuit sub Licenţa Publică Generală **GNU** (în engleză GNU General Public License, prescurtat **GNU GPL** – este o licenţă software care are scopul de a da dreptul oricărui utilizator de a copia, modifica şi redistribui programe şi coduri sursă ae programatorilor care îşi licenţiază operele sub tutela **GPL**). Este cel mai popular **SGBD** open-source la ora actuală. Se bazează pe **SQL**.

Visual FoxPro. Este realizat de firma Microsoft. Are un limbaj procedural propiu foarte puternic, o extensie orientată obiect, programare vizuală şi nucleu extins de **SQL**.

Access. Este realizat de firma Microsoft. Se bazează pe **SQL**, are limbajul procedural gazdă (Basic Access) şi instrumente de dezvoltare.

Modelul unei baze de date este o specificaţie tehnică acceptată de mai mulţi furnizori de programe de baze de date (**DBMS**) ce se referă la modul în care sunt stocate informaţiile în baza de date şi modul în care sunt folosite.

Exemple de modele sunt: modelul relaţional, modelul orientat-obiect, modelul ierarhic, etc.

Cel mai răspândit în prezent este modelul relaţional. Bazele de date relaţionale au informaţiile organizate în tabele, iar între informaţiile din aceste tabele pot fi stabilite legături.

1.6 Colectarea şi analizarea datelor. Modelul conceptual

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

Modelul conceptual al datelor este o reprezentare vizuală clară şi corectă a activităţii unei organizaţii.

Modelul conceptual al datelor include *entităţile* (informaţiile) majore şi *relaţiile* dintre acestea şi nu conţine informaţii detaliate privind atributele (caracteristicile) entităţilor. Este generat prin identificarea cerinţelor afacerii modelate, aşa cum rezultă din documente, discuţii cu personalul implicat, cu analişti şi experţi ai domeniului studiat, cu beneficiarii finali. *Modelul conceptual* realizat va fi prezentat echipelor funcţionale în vederea revizuirii.

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

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

Informaţiile obţinute în etapa documentării vor fi reprezentate într-o formă convenţională care să poată fi uşor înţeleasă de toată lumea. O astfel de reprezentare este diagrama entităţi-relaţii numită şi harta relaţiilor **ERD** (**E**ntity **R**elationship **D**iagram).

Aceste scheme sunt un instrument util care uşurează comunicarea dintre specialiştii care proiectează bazele de date şi programatori, pe de o parte, şi beneficiari, pe de altă parte. Aceştia din urmă pot înţelege cu uşurinţă o astfel de schemă, chiar dacă nu sunt cunoscători în domeniul IT.

Diagramele **ERD** sunt uşor de creat şi de actualizat, însă, avantajul major al lor este dat de simplitatea reprezentărilor ce facilitează înţelegerea şi de către nespecialişti.

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

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

Iată în continuare principalele elemente care intră în componenţa unui **ERD** precum şi convenţiile de reprezentare a acestora.

1.7 Entităţi. Instanţe. Atribute. Identificator unic.

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

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

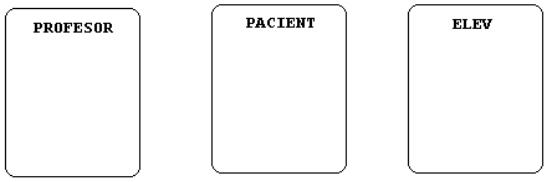


Fig.1.1

Entităţile sunt clase de obiecte de acelaşi tip, un obiect al clasei reprezentând o *instanţă* a entităţii.

O *instanţă* a unei entităţi este un obiect, persoană, eveniment, particular din clasa de obiecte care formează entitatea. De exemplu, elevul X din clasa a IX-a A de la Liceul de Informatică din localitatea Y este o instanţă a entităţii ELEV.

După cum se vede pentru a preciza o instanţă a unei entităţi, trebuie să specificăm unele caracteristici ale acestui obiect, să-l descriem (în cazul entităţii ELEV precizăm de exemplu numele, clasa, şcoala etc).

Aşadar, după ce am identificat entităţile trebuie să descriem aceste entităţi în termeni reali, adică să le stabilim atributele.

Entităţile sunt descrise folosind **atribute**, fiecare atribut având fie o singură valoare, fie niciuna. Valorile atributelor pot fi de tip numeric (ex. 100; -5; 14.3), şir de caractere (ex. Popescu, XII A), dată calendaristică (ex. 12/02/2010) etc.

Un *atribut* este orice detaliu care serveşte la identificarea, clasificarea, cuantificarea, sau exprimarea stării unei instanţe a unei entităţi. Atributele sunt informaţii specifice ce trebuie cunoscute şi memorate.

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

Atributele cărora le poate lipsi valoarea se numesc *atribute opţionale*, iar cele cărora trebuie să le atribuim o valoare se numesc *atribute obligatorii*. Dacă un atribut este obligatoriu, pentru fiecare instanţă a entităţii respective trebuie să avem o valoare pentru acel atribut, de exemplu este obligatoriu să cunoaştem numele elevilor.

Pentru un atribut opţional putem avea instanţe pentru care nu cunoaştem valoarea atributului respectiv. De exemplu atributul email al entităţii ELEV este opţional, un elev putând să nu aibă adresă de email. Un atribut obligatoriu este precedat în **ERD** de un asterisc „*”, iar un atribut opţional va fi precedat de un cerculeţ „o”.

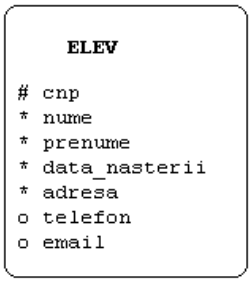


Fig. 1.2

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

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

Fig. 1.3

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

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

Atributul sau *ansamblul de atribute* ce identifică în mod unic o instanţă a entităţii (valoarea sau combinaţia de valori ale atributelor este unică pentru fiecare instanţă) se numeşte **identificator unic** (**Unique IDentifier** - **UID**).

Identificatorii unici sunt obligatorii şi sunt precedaţi de caracterul # (diez). Se pot folosi identificatori unici *naturali* (atribute ce au o semnificaţie concretă pentru entitatea respectivă) sau *artificiali* (atribute create şi menţinute în mod artificial, arbitrar de noi).

Entitatea CANDIDAT este descrisă de unsprezece atribute: id_candidat, nume, initiala, prenume, cnp, mediul, serii_anterioare, taxa, adresa, telefon, data_inscriere. Identificatorul unic al entităţii este *id_candidat,* un **UID** de tip artificial. Un **UID** natural al entităţii poate fi stabilit atributul *cnp*, întrucât nu există două persoane cu acelaşi cod numeric personal, sau o pereche de atribute ce ar asigura unicitatea : perechea (nume, iniţiala, prenume, adresa).

În continuare aveţi reprezentarea grafică a entităţii CANDIDAT

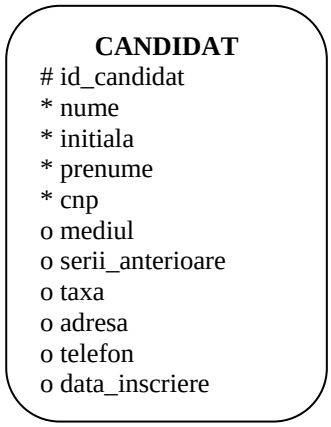


Fig 1.4 Entitatea CANDIDAT

Atributele obligatorii sunt nume, initiala, prenume, cnp, mediul, iar cele opţionale serii_anterioare, taxa, adresa, telefon, data_inscriere.

Se recomandă evitarea folosirii atributelor ale căror valori se modifică frecvent (de exemplu vârsta unei persoane), aşa-numitele atribute volatile. Acestea pot fi înlocuite cu atribute non-volatile (de ex. data naşterii, ce este o constantă pentru fiecare persoană).

De asemenea sunt de evitat atributele ale căror valori pot fi deduse prin diferite prelucrări din atribute definite anterior. În entitatea CANDIDAT nu are sens să adăugăm atributele data naşterii, sex, vârstă, întrucât ele pot fi obţinute prin prelucrări elementare din atributul cnp.

1.8 Etapele realizării unei aplicaţii informatice

Realizarea unei aplicaţii informatice ce utilizează baze de date necesită parcurgerea unei succesiuni de etape: analiza sistemului, proiectarea bazei de date, realizarea componentelor logice, punerea în funcţiune, dezvoltarea şi întreţinerea (mentenanţa).

1.9 Analiza sistemului informatizat

Scopul analizei sistemului este de a evidenţia cerinţele aplicaţiei şi resursele utilizate, precum şi de a evalua aceste cerinţe prin modelare (analiza). Studiul situaţiei existente se realizează prin: identificarea caracteristicilor generale ale unităţii, identificarea activităţilor desfăşurate,

identificarea resurselor existente (informaţionale, umane, echipamente), identificarea necesităţilor de prelucrare.

Analiza este o activitate de modelare (conceptuală) şi se realizează sub trei aspecte: structural, dinamic şi funcţional.

a) *Analiza structurală* evidenţiază, la nivel conceptual, modul de structurare a datelor şi a legăturilor dintre ele. Cea mai utilizată tehnică este entitate-asociere. Aceasta conţine:

- Identificarea entităţilor: fenomene, procese, obiecte concrete sau abstracte;
- Identificarea asocierilor dintre entităţi ca fiind legăturile semnificative de un anumit tip;
- Identificarea atributelor ce caracterizează fiecare entitate în parte;
- Stabilirea atributelor de identificare unică a instanţelor entităţilor;
- Identificarea şi eliminarea anomaliilor de date şi minimizarea redundanţei datelor prin procesul de normalizare a entităţilor.

Rezultatul analizei structurale este modelul *conceptual static* al datelor, numit şi diagrama **ERD** – **E**ntity **R**elationship **D**iagram. Pornind de la o astfel de diagramă, se pot construi, în activitatea de proiectare, schemele relaţiilor (tabelelor).

b) *Analiza dinamică* evidenţiază comportamentul elementelor sistemului la anumite evenimente.

Una din tehnicile utilizate este diagrama stare-tranziţie. Aceasta presupune:

- Identificarea stărilor în care se pot afla componentele sistemului;
- Identificarea evenimentelor care determină trecerea unei componente dintr-o stare în alta;
- Stabilirea tranziţiilor admise între stări;
- Construirea diagramei stare-tranziţie.

Rezultatul analizei dinamice este *modelul dinamic*.

c) *Analiza funcţională* evidenţiază modul de asigurare a cerinţelor informaţionale (fluxul prelucrărilor) din cadrul sistemului, prin care intrările sunt transformate în ieşiri. Prin analiza funcţională se delimitează:

- Aria de cuprindere a aplicaţiei informatice;
- Se identifică sursele de date;
- Se identifică modul de circulaţie şi prelucrare a datelor;
- Se identifică apoi rezultatele obţinute.

Rezultatul analizei funcţionale este *modelul funcţional*.

1.10 Modelul relaţional

Modelul relaţional a fost propus de către IBM şi a revoluţionat reprezentarea datelor făcând trecerea la generaţia a doua de baze de date. Modelul este simplu, are o solidă fundamentare

teoretică fiind bazat pe teoria seturilor (ansamblurilor) şi pe logica matematică. Pot fi reprezentate toate tipurile de structuri de date de mare complexitate, din diferite domenii de activitate.

Modelul relaţional este definit prin: structura de date, operatorii care acţionează asupra structurii şi restricţiile de integritate.

Conceptele utilizate pentru definirea structurii de date sunt: *domeniul*, *tabela* (*relaţia*), *atributul*, *tuplul*, *cheia* şi *schema tabelei*.

Domeniul este un ansamblu de valori caracterizat printr-un nume. El poate fi explicit sau implicit.

Tabela/relaţia este un subansamblu al produsului cartezian al mai multor domenii, caracterizat printr-un nume, prin care se definesc atributele ce aparţin aceleaşi clase de entităţi.

Atributul este coloana unei tabele, caracterizată printr-un nume.

Cheia este un atribut sau un ansamblu de atribute care au rolul de a identifica un tuplu dintr-o tabelă. Tipuri de chei: *primare*, *externe*.

Tuplul este linia dintr-o *tabelă* şi nu are nume. Ordinea liniilor (tupluri) şi coloanelor (atribute) dintr-o tabelă nu trebuie să prezinte nici o importanţă.

Schema tabelei este formată din numele tabelei, urmat între paranteze rotunde de lista atributelor, iar pentru fiecare atribut se precizează domeniul asociat.

Schema bazei de date poate fi reprezentată printr-o diagramă de structură în care sunt puse în evidenţă şi legăturile dintre tabele. Definirea legăturilor dintre tabele sau a relaţiilor dintre tabele se face logic construind asocieri între tabele cu ajutorul unor atribute de legătură.

Cardinalitatea relaţiei este egală cu numărul de linii sau tupluri conţinute de un tabel.

Cheia primară a unei tabele reprezintă un *atribut* sau un *ansamblu de atribute* care *identifică în mod unic* o înregistrare dintr-o tabelă a unei baze de date.

Cheia primară formată dintr-un singur atribut este o cheie simplă, iar cea formată dintr-un ansamblu de atribute se numeşte cheie compusă.

Orice tabelă are o cheie primară care trebuie să aibă valori unice şi nenule.

Deci, două înregistrări (linii) nu pot avea aceaşi valoare a cheii primare, fiecare înregistrare trebuie să aibă o valoare în câmpul (coloana) care este cheie primară. Coloana care conţine valorile cheilor primare nu poate di modificată sau actualizată. Valorile cheilor primare nu pot fi refolosite, dacă o înregistrare este ştearsă din tabelă cheia ei nu va fi atribuită altor înregistrări noi.

Dacă avem o cheie primară compusă atunci regulile enunţate mai sus se aplică asupra ansamblului de atribute ce alcătuiesc cheia primară luate laolaltă.

Un *atribut* ce îndeplineşte condiţiile necesare cheii primare se numeşte *cheie candidat*. Într-o entitate pot exista mai multe atribute ce pot fi cheie primară. Dintre aceste chei candidat se va alege, de fapt, *cheia primară*.

Atributele implicate în realizarea legăturilor se găsesc fie în tabelele asociate, fie în tabele distincte construite special pentru legături.

Atributul din tabela iniţială se numeşte *cheie externă* iar cel din tabela finală este *cheie primară*.

Cheia externă (foreign key) este atributul sau ansamblul de atribute ce serveşte la realizarea legăturii cu o altă tabelă în care acest atribut sau ansamblu de atribute este cheie primară. Valorile asociate atributului cu rol de cheie externă pot fi duplicate sau nule.

Condiţiile pe care trebuie să le îndeplinească cheile externe sunt următoarele:

- fiecare valoare a unei *chei externe* trebuie să se regăsească printre mulţimea valorilor *cheii primare corespondente*, reciproca nu este valabilă;

- o *cheie externă* este simplă dacă şi numai dacă *cheia primară* corespondentă este simplă şi este compusă dacă şi numai dacă *cheia primară* corespondentă este compusă;

- fiecare atribut component al unei *chei externe* trebuie să fie definit pe acelaşi domeniu al componentei corespondente din *cheia primară*;

- o valoare a unei *chei externe* reprezintă o referinţă către o înregistrare care conţine aceeaşi valoare pentru *cheia primară corespondentă*, deci această valoare trebuie să existe.

Această ultimă condiţie este cea a *integrităţii referinţei*, deci, dacă avem o cheie externă A care face referire la o cheie primară B, atunci B trebuie să existe.

Constrângerile de integritate reprezintă reguli pe care valorile conţinute într-o tabelă trebuie să le respecte. Aceste constrângeri previn introducerea de date eronate în tabele.

Deci *cheia externă* este o *constrângere*, prin aplicarea acestei constrângeri valorile unei coloane sunt forţate să fie doar dintre cele ale cheii primare corespondente. Aceasta poartă numele de *constrângere de integritate referenţială*.

Constrângerile de integritate sunt verificate automat de **SGBD** atunci când au loc operaţii de modificare a conţinutului unei tabele (introducere, modificare sau ştergere de înregistrări). În cazul în care valorile nu sunt valide operaţia nu se efectuează şi se generează o eroare.

Fiecare constrângere de integritate poate avea asociat un nume, în cazul în care nu se asociază un nume, sistemul generează automat unul.

O constrângere poate fi definită în descrierea unei coloane dacă se referă doar la acea coloană sau la finalul listei de descrieri a coloanelor.

Tipuri de constrângeri:

- NOT NULL – valorile nu pot fi nule;
- PRIMARY KEY – defineşte cheia primară;
- UNIQUE – defineşte uncitatea;
- FOREIGN KEY – defineşte o cheie externă;

- CHECK – introduce o condiţie (expresie logică).

Avem următoarele legături posibile între tabele:

- *unu-la-unu* (*one-to-one*, *1:1*);
- *unu-la-mai mulţi* (*one-to-many*, *1:m*);
- *mai-mulţi-la-mai-mulţi* (*many-to-many*, *m:n*).

Potenţial, orice tabelă se poate lega cu orice tabelă, după orice atribute. Legăturile se stabilesc la momentul descrierii datelor prin limbaje de descriere a datelor (**LDD**), cu ajutorul restricţiilor de integritate. Practic, se stabilesc şi legături dinamice la momentul execuţiei.

1.11 Operatorii modelului relaţional

Operatorii modelului relaţional sunt operatorii din *algebra relaţională* şi operatorii din *calculul relaţional*.

Algebra relaţională este o colecţie de operaţii formale aplicate asupra tabelelor (relaţiilor), şi a fost concepută de E.F.Codd.

Operaţiile sunt aplicate în expresiile algebrice relaţionale care sunt cereri de regăsire. Acestea sunt compuse din operatorii relaţionali şi operanzi.

Operanzii sunt întotdeauna tabele (una sau mai multe). Rezultatul evaluării unei expresii relaţionale este format dintr-o singură tabelă.

Algebra relaţională are cel puţin puterea de regăsire a calculului relaţional. O expresie din calculul relaţional se poate transforma într-una echivalentă din algebra relaţională şi invers. Codd a introdus şase operatori de bază (reuniunea, diferenţa, produsul cartezian, selecţia, proiecţia, joncţiunea) şi doi operatori derivaţi (intersecţia şi diviziunea). Ulterior au fost introduşi şi alţi operatori derivaţi (speciali).

În acest context, operatorii din algebra relaţională pot fi grupaţi în două categorii: pe mulţimi şi speciali.

Operatori pe mulţimi (R1, R2, R3 sunt relaţii (tabele)) sunt:

• **Reuniunea.** $R3 = R1 \cup R2$, unde R3 va conţine tupluri din R1 sau R2 luate o singură dată;

• **Diferenţa.** $R3 = R1 \setminus R2$, unde R3 va conţine tupluri din R1 care nu se regăsesc în R2;

• **Produsul cartezian.** $R3 = R1 \times R2$, unde R3 va conţine tupluri construite din perechi (x1,x2), cu x1 aparţine R1 şi x2 aparţine R2;

• **Intersecţia.** $R3 = R1 \cap R2$, unde R3 va conţine tupluri care se găsesc în R1 şi R2 în acelaşi timp;

Operatori relaţionali speciali sunt:

• **Selecţia**. Din R1 se obţine o subtabelă R2, care va conţine o submulţime din tuplurile iniţiale din R1 ce satisfac o condiţie. Numărul de atribute din R2 este egal cu numărul de atribute din R1. Numărul de tupluri din R2 este mai mic decât numărul de tupluri din R1.

• **Proiecţia**. Din R1 se obţine o subtabelă R2, care va conţine o submulţime din atributele iniţiale din R1 şi fără tupluri duplicate. Numărul de atribute din R2 este mai mic decât numărul de atribute din R1.

• **Joncţiunea** este o derivaţie a produsului cartezian, ce presupune utilizarea unui calificator care să permită compararea valorilor unor atribute din R1 şi R2, iar rezultatul în R3. R1 şi R2 trebuie să aibă unul sau mai multe atribute comune care au valori comune.

Proiectarea schemei conceptuale porneşte de la identificarea setului de date necesar sistemului. Aceste date sunt apoi integrate şi structurate într-o schemă (exemplu: pentru **BDR** relaţionale cea mai utilizată tehnică este *normalizarea*).

1.12 Normalizarea

Tehnica de *normalizare* este utilizată în activitatea de proiectare a structurii **BDR** şi constă în eliminarea unor anomalii (neajunsuri) de actualizare din structură.

Anomaliile de actualizare sunt situaţii nedorite care pot fi generate de anumite tabele în procesul proiectării lor:

• Anomalia de ştergere semnifică faptul că ştergând un tuplu dintr-o tabelă, pe lângă informaţiile care trebuie şterse, se pierd şi informaţiile utile existente în tuplul respectiv;

• Anomaliile de adăugare semnifică faptul că nu pot fi incluse noi informaţii necesare într-o tabelă, deoarece nu se cunosc şi alte informaţii utile (de exemplu valorile pentru cheie);

• Anomalia de modificare semnifică faptul că este dificil de modificat o valoare a unui atribut atunci când ea apare în mai multe tupluri.

Normalizarea este o teorie construită în jurul conceptului de forme normale (**FN**), care ameliorează structura **BDR** prin înlăturarea treptată a unor neajunsuri şi prin imprimarea unor facilităţi sporite privind manipularea datelor.

Normalizarea utilizează ca metodă descompunerea (top-down) unei tabele în două sau mai multe tabele, păstrând informaţii (atribute) de legătură.

FN1. O tabelă este în **FN1** dacă toate atributele ei conţin valori elementare (nedecompozabile), adică fiecare tuplu nu trebuie să aibă date la nivel de grup sau repetitiv. Structurile de tip arborescent şi reţea se transformă în tabele cu atribute elemntare. O tabelă în **FN1** prezintă încă o serie de anomalii de actualizare datorită eventualelor dependenţe funcţionale incomplete. Fiecare structură repetitivă generează (prin descompunere) o nouă tabelă, iar atributele la nivel de grup se înlătură, rămânând doar cele elemntare.

FN2. O tabelă este în **FN2** dacă şi numai dacă este în **FN1** şi fiecare atribut noncheie al tabelei este dependent funcţional complet de cheie. Un atribut B al unei tabele depinde funcţional de atributul A al aceleiaşi tabele, dacă fiecărei valori a lui A îi corespunde o singură valoare a lui B, care îi este asociată în tabelă. Un atribut B este dependent funcţional complet de un ansamblu de

atribute A în cadrul aceleiaşi tabele, dacă B este dependent funcţional de întreg ansamblul A (nu numai de un atribut din ansamblu). O tabelă în **FN2** prezintă încă o serie de anomalii de actualizare, datorită eventualelor dependenţe tranzitive. Eliminarea dependenţelor incomplete se face prin descompunerea tabelei iniţiale în două tabele, ambele conţinând atributul intermediar (B).

FN3. O tabelă este în **FN3** dacă şi numai dacă este în **FN2** şi fiecare atribut noncheie depinde în mod netranzitiv de cheia tabelei. Într-o tabelă T, fie A,B,C trei atribute cu A cheie. Dacă B depinde de A (A → B) şi C depinde de B (B → C) atunci C depinde de A în mod tranzitiv. Eliminarea dependenţelor tranzitive se face prin descompunerea tabelei iniţiale în două tabele, ambele conţinând atributul intermediar (B). O tabelă în **FN3** prezintă încă o serie de anomalii de actualizare, datorate eventualelor dependenţe multivaloare.

O definiţie mai riguroasă pentru **FN3** a fost dată prin forma intermediară **BCNF** (Boyce Codd Normal Form): o tabelă este în **BCNF** dacă fiecare determinant este un candidat cheie. Determinantul este un atribut elementar sau compus faţă de care alte atribute sunt complet dependente funcţional.

FN4. O tabelă este în **FN4** dacă şi numai dacă este în **FN3** şi nu conţine două sau mai multe dependenţe multivaloare. Într-o tabelă T, fie A,B,C trei atribute. În tabela T se menţine dependenţa multivaloare A dacă şi numai dacă mulţimea valorilor lui B ce corespunde unei perechi de date (A,C), depinde numai de o valoare a lui A şi este independentă de valorile lui C.

FN5. O tabelă este în **FN5** dacă şi numai dacă este în **FN4** şi fiecare dependenţă joncţiune este generată printr-un candidat cheie al tabelei. În tabela T (A,B,C) se menţine dependenţa joncţiune (AB, AC) dacă şi numai dacă T menţine dependenţa multivaloare A -->> B sau C.

Dependenţa multivaloare este caz particular al dependenţei joncţiune. Dependenţa funcţională este caz particular al dependenţei multivaloare.

În concluzie, în această primă lecţie au fost prezentate şi explicate mai multe noţiuni teoretice fundamentale pentru a înţelege de ce folosim baze de date relaţionale şi ce noţiuni sunt întâlnite frecvent în lucrul cu baze de date relaţionale. Pe lângă definirea unor noţiuni de bază, a fost abordat şi subiectul normalizării unei baze de date, precum şi prezentarea modelului relaţional.

Lecţia următoare se va axa în continuare pe prezentarea unor noţiuni teoretice legate de tipurile de relaţii întâlnite între tabele, dar va conţine şi exemple concrete de proiectare a bazelor de date, precum şi o prezentare a mediilor de lucru folosite pentru a avea instalat un server de baze de date **MySQL** şi pentru conectarea la baza de date, respectiv efectuarea de comenzi **SQL**.