MARIANA PANȚIRU

TEHNOLOGIA INFORMAȚIEI şi a COMUNICAȚIILOR

TIC 4 (sisteme de gestiune a bazelor de date)

Manual pentru clasa a 12-a

filiera teoretică / profil real /
specializarea: ştiințele ale naturii

Editura ALL

Cuprins

Lecția 1 Modelarea conceptuală a unei bazei de date

- ✓ *Baze de date şi SGBD-uri. Particularitățile modelului relațional*
- ✓ *Proiectarea structurii conceptuale a unei baze de date*
- ✓ *Studiu de caz*

1.1. Despre informații şi date. Problema organizării datelor

Activitatea umană, în cele mai diverse forme ale sale, a fost întotdeauna caracterizată prin entități faptice exprimate fie sub formă de valori numerice, fie ca percepții sau observații nenumerice făcute de ființele umane sau de maşini. Aceste entități faptice independente şi neevaluate se vor numi DATE. Datele obținute în cadrul activităților productive, de conducere, de cercetare, educaționale, artistice constituie un material informațional brut care poate fi *evaluat, ordonat şi prelucrat,* având în vedere diferite obiective. În urma acestui proces de transformare a datelor se obțin **informații,** care reprezintă o interpretare a datelor în raport cu anumite situații particulare sau cu înțelegerea de către mintea umană în general. Informațiile constituie baza raționamentelor, experimentărilor imaginate de mintea umană în scopul obținerii de noi **cunoştințe.**

Producerea informațiilor susține anumite acțiuni umane cu finalitate practică, creându-se totodată un fond informațional utilizabil pentru generarea de noi informații şi cunoştințe.

Informația este definită ca „o comunicare susceptibilă de a produce cunoştințe“, are un caracter semantic şi de noutate, de aport la cantitatea de cunoştințe a celui care o primeşte. Informația se referă întotdeauna la obiecte, persoane, procese, fenomene, locuri, situații, condiții etc. O informație trebuie să fie utilă, precisă, completă, fără prea multe amănunte care să facă mesajul greu de interpretat, să sosească la timp pentru luarea unor decizii. Eschil zicea că înțelept nu este cel ce ştie lucruri *multe* ci lucruri *utile.* O altă trăsătură fundamentală a informației este subiectivitatea sa. Ceea ce este o informație pentru o persoană poate să nu însemne nimic pentru alta.

Datele constituie materializarea, reprezentarea simbolică a informațiilor (prin semne, litere, cifre) într-o formă convențională (scrisă, vorbită, luminoasă, semne grafice, desene) convenabilă unei comunicări. Datele descriu ce s-a întâmplat prin număr sau cuvânt, ele nu furnizează vreo judecată sau interpretare şi nici bază pentru acțiune. Datele reprezintă „materia primă“ din care, după o serie de operațiuni efectuate de către oameni sau echipamente, se obțin informații.

Informația reprezintă „semnificația ce poate fi desprinsă dintr-un ansamblu de date, pe baza asociațiilor dintre acestea“, este deci partea utilizabilă dintr-un ansamblu de date. Informațiile furnizează răspunsuri la întrebări de genul „cine”, „unde”, „când”. Datele sunt exprimate prin atribut şi valoare. Exemple: produs=„imprimantă”, cantitate=20, prețul=150 lei. Informația se exprimă prin propoziții. Exemplu: „S-au vândut 20 imprimante la prețul de 50 lei”. Informația presupune înțelegerea relațiilor dintre date.

Colecția de date reprezintă un ansamblu de date care se referă la acelaşi fenomen, obiect sau situație.

Între componentele unei colecții de date, ca şi între colecții, pot fi identificate sau, eventual, pot fi introduse **relații** care să determine pe mulțimea respectivă o anumită **structură**, adică un anumit mod de ordonare astfel încât să se faciliteze prelucrarea. Relațiile pot fi unidirecționale sau bidirecționale. O relație unidirecțională poate determina obținerea unei valori sau a mai multora din componenta legată, plecând numai de la o componentă (părinte); o relație bidirecțională determină obținerea aceloraşi date/valori, plecând de la ambele colecții. Componentele structurii pot fi individualizate şi selectate prin **poziția** pe care o ocupă în structură, în raport cu ordinea specificată sau, mai comod, prin **numele componentei.**

O colecție de date pe care s-au definit anumite relații şi căreia îi este specific un anumit mecanism de selecție şi identificare a componentelor constituie o **structură de date**. Mecanismul de selecție al unei structuri de date este implementat de obicei în programele de prelucrare a datelor respective, la nivelul sistemului de operare, al sistemului de gestiune sau al programelor aplicative.

Sunt două mari **tipuri de acces**:

– accesul secvențial presupune parcurgerea tuturor datelor situate înaintea datei care trebuie prelucrată.

– accesul direct presupune un mecanism prin care se poate determina direct poziția datei necesare prelucrării.

Operații generale asupra unei structuri de date:

- creare (memorarea pe suport a datelor);
- consultare (acces la elementele structurii pentru prelucrarea valorilor);
- actualizare (inserare, ştergere sau corecție a valorilor);
- copiere (duplicarea structurii pe un alt suport, în general);
- ventilare (desfacerea structurii în două sau mai multe structuri);
- fuzionare (combinarea a două sau mai multe structuri);
- sortare (aranjarea elementelor structurii după anumite criterii).

Cerințele complexe impuse sistemelor informatice moderne de a avea acces simultan la aceleaşi date ale mai multor utilizatori, local sau la distanță, au condus la perfecționarea metodelor de organizare şi, astfel, au apărut bazele de date. Văzute ca un depozit central de date împreună cu descrierea lor, bazele de date, pot fi accesate de diferiți utilizatori, nu neaparat programatori. Principalul avantaj îl constituie separarea problemelor de organizare a datelor față de cele de proiectare ale programelor utilizatorilor. La nivelul unei baze de date se definesc 3 SCHEME (descrieri pentru structurile conceptuale, logice şi fizice).

O bază de date trebuind să facă față cerințelor uneori contradictorii ale diferiților utilizatori, va conține acele date strict necesare tuturor utilizatorilor, într-un format care este în majoritate acceptat, iar programe speciale de gestiune-grupate sub numele de SGBD-sistem de gestiune a bazei de date – vor rezolva cerințele particulare.

Baza de date este văzută ca o colecție de date legate funcțional între ele. Se vorbeşte deci de o bază pentru biblioteca şcolii, de altă bază de date pentru cantină etc.

Baza de date (BD) poate fi definită ca un ansamblu de date interconectate, împreună cu descrierea lor, care răspunde calităților de centralizare, coordonare, integrare şi difuzie a informațiilor şi care asigură satisfacerea tuturor necesităților de prelucrare ale tuturor utilizatorilor dintr-un sistem.

Sistemul de Gestiune a Bazei de Date (SGBD) este un pachet de programe ce permite utilizatorilor să interacționeze cu o bază de date, asigurând acesteia următoarele caracteristici:

1. Independența datelor față de programul care le gestionează.

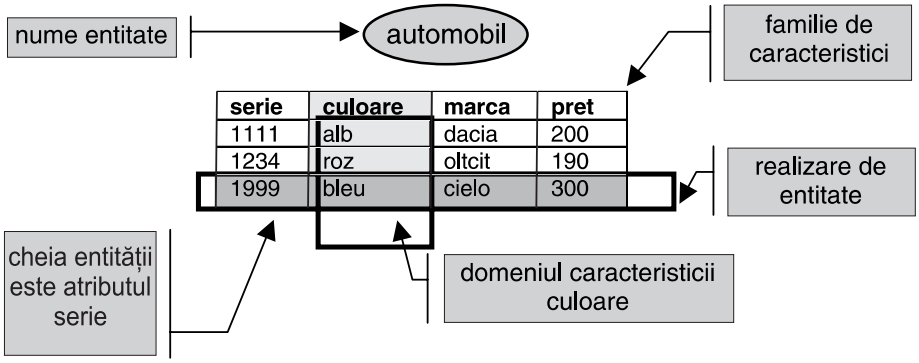
2. Nivel redus de redundanță.
3. Securitatea datelor (protecția la accesul neautorizat în vederea extragerii sau distrugerii unor date cu caracter confidențial).
4. Integritatea datelor (protecția la defecțiuni hard sau soft).
5. Transparență (facilități de utilizare a datelor, fără ca utilizatorii să cunoască baza de date în întreaga ei complexitate).
6. Limbaje de descriere şi manipulare a datelor de nivel foarte înalt**.** Existența unor limbaje performante de regăsire a datelor care permit exprimarea sub forma unor conversații a unor criterii cât mai complexe de selectare a informației şi indicării unor reguli cât mai generale de editare a informațiilor solicitate.
7. Facilități multiutilizator. Datele pot fi accesate şi chiar gestionate din diferite noduri ale rețelei de calculatoare de către diferiți utilizatori.
8. Accesibilitate. Gestiunea datelor organizate în baze de date a fost preocuparea multor specialişti soft, ajungându-se la oferirea unor pachete de gestiune care permit gestionarea datelor foarte complexe, în condiții de eficiență maximă.

Bazele de date pot conține date de diferite tipuri (texte, numere, imagini, documente). Pot cuprinde date operaționale unui domeniu de activitate, ca, de exemplu, baze de date pentru gestiunea resurselor umane, pentru gestiunea producției etc. Unele baze de date conțin date comerciale sau statistice ale unor instituții externe, accesibile on-line şi care servesc procesului decizional.

După modelul bazelor de date SGBD-urile pot fi ierarhice (ex. IMS, GIS), rețea (ex. SOCRATE), relaționale (ex. Dbase, ACCESS, ORACLE, FOXPRO), orientate-obiect (ex. O2, UniSQL, IRIS), obiectual-relațional (ORACLE).

1.2. Terminologie şi concepte specifice bazelor de date

Elementul fundamental al modelului conceptual este *entitatea,* ca termen generic pentru a desemna obiectele similare ca structură, dar care sunt identificabile, deci se pot deosebi între ele prin trăsături specifice. O entitate este formată dintr-o mulțime de *atribute (*sau *caracteristici)* care pot defini complet obiectul. Lista atributelor care pot caracteriza o entitate poartă numele de *familie de caracteristici.* Valorile familiei de caracteristici corespunzătoare unui obiect distinct poartă numele de *realizare de entitate* sau *instanță.* Caracteristica asociază câte o valoare dintr-un domeniu fiecărei realizări a respectivei entități. *Domeniile* pot fi deci numere întregi, şiruri etc. Un atribut sau un set de atribute care identifică în mod unic fiecare instanță a unei entități se numeşte *cheie.*



O entitate poate fi definită prin liste diferite de caracteristici în funcție de sistemul informațional căruia îi aparține.[1]

Relațiile care pot fi stabilite între două entități ale unei baze de date pot fi:

Relația 1-1 (unu-la-unu) – unei instanțe a entității „părinte" îi poate corespunde o singură instanță în entitatea „copil". De exemplu, între entitatea CLASA şi PROFESOR, relația „diriginte" este de tip 1-1 dacă un profesor este diriginte la o singură clasă, iar o clasă are un singur diriginte.

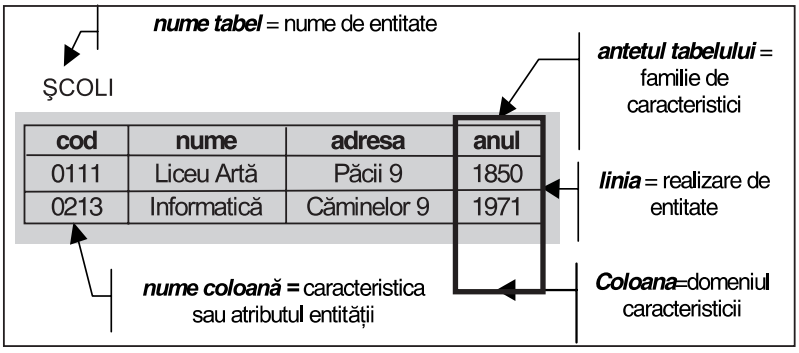
Relația 1-N (unu-la-mulți) – unei realizări a entității „părinte" îi poate corespunde una sau mai multe realizări în entitatea „copil". Unei realizări din entitatea „copil" îi corespunde o singură realizare în entitatea „părinte". De exemplu, între entitatea CLASA şi ELEV relația „învață" este de tip 1-N, pentru că o clasă are mai mulți elevi şi un elev învață într-o singură clasă.

Relația N-N (mulți-la-mulți) – unei realizări a entității „părinte" îi poate corespunde una sau mai multe realizări în entitatea „copil" şi unei realizări din „copil" îi pot corespunde *n* realizări în „părinte". De exemplu, între entitățile PROFESOR şi CLASA, relația „predă" este de tip N-N pentru că la o clasă pot preda mai mulți profesori, iar un profesor poate preda la una sau mai multe clase.

Modelul relațional

În bazele de date relaționale entitățile sunt organizate în tabele simple, bidimensionale, fără legături fixe. Relațiile necesare sunt stabilite prin asocierea între ele a unor câmpuri cheie ale fiecărei tabele. Modelul relațional este caracterizat prin unitatea şi simplitatea reprezentărilor: totul se reduce la tabele. De asemenea, modelul păstrează rigoarea fundamentării sale matematice, fapt care a permis standardizarea unor limbaje de nivel foarte înalt – în special SQL – care utilizează elemente de algebră relațională. Este modelul care a „revoluționat" lumea bazelor de date şi asupra lui vom insista în lecțiile care urmează.

Observați corespondența între termenii generali acceptați de teoria bazelor de date şi termenii specifici modelului relațional, în figura următoare:



O entitate este o relație, adică o tabelă care are linii şi coloane.

[1] entitatea AUTOMOBIL ce desemnează un obiect în cadrul sistemului informațional al unei unități producătoare de maşini poate fi descrisă prin atributele *tip-motor, model, număr-de-portiere, forma-caroseriei* etc. Dacă, însă, entitatea AUTOMOBIL se referă la obiectele supuse vânzării într-o unitate comercială, atunci va trebui să conțină: *seria-motorului, firma producătoare, prețul* etc.

În SGBDR nu pot fi reprezentate relații tip N-N. Ele vor fi „sparte" în procesul normalizării bazei de date în relații 1-1 şi/sau 1-n.

O ***schemă relațională*** poate fi definită ca un ansamblu de relații asociate semantic prin domeniul lor de definiție şi prin anumite *restricții de integritate.* Schema relațională este independentă în timp. Pentru păstrarea integrității unei baze de date se propune un limbaj declarativ care memorează *regulile de integritate;* acestea pot fi:

a. restricțiile cheilor primare – condiția de unicitate şi de valori non-nule;
b. restricții referențiale – apar atunci când o tabelă este în relație cu alta (de exemplu, cheia străină nu trebuie să aibă valori care nu se regăsesc drept valori ale cheii primare în tabela de referință);
c. restricții de comportament – puse pe valorile diferitelor atribute sau pe întreaga înregistrare.

1.3. Proiectarea unei baze de date

Pentru modelarea unei baze de date se parcurg câteva faze care se încadrează în metodologia generală pentru dezvoltarea aplicațiilor informatice. Prezentăm acum numai aspectele legate de proiectarea bazei de date.

Prima fază este analiza sistemului informațional existent, în care se cercetează:

❶ informațiile, sursele şi destinațiile lor;
❷ principalele activități legate de date: prelucrări, reguli de calcul, validări, puncte de control, modalități de arhivare;
❸ sistemul de codificare existent etc.

Analiza sistemului informațional existent se poate realiza prin două metode. Prima, cea tradițională, este *metoda procedurală,* în care se pune accent pe fluxurile de prelucrări, întrebarea fundamentală fiind: „de ce date am nevoie pentru a realiza sarcina X?" Deci, se pleacă de la studiul cererilor de informații (=informații de ieşire) şi, mergând pe fluxul invers al prelucrărilor, se determină informațiile de intrare, cele care trebuie colectate, validate şi organizate într-o bază de date. O altă metodă de analiză este cea orientată obiect, în care întrebările sunt de felul: „Ce clase de obiecte distincte pot fi definite pe domeniul aplicației? Cum se comportă ele? Ce evenimente pot apărea? Ce acțiuni sunt posibile?"

A doua fază constă în proiectarea entităților şi a relațiilor „fireşti" dintre ele; obținem astfel SCHEMA. Pentru fiecare legătură se stabileşte tipul ei (1-1, 1-n, n-n).

Faza a treia ar fi proiectarea structurii unei baze relaționale, transformând entitățile şi relațiile în tabele. Tabelele vor fi definite prin câmpurile lor, caracteristicile acestor câmpuri (tip, lungime), condiții de validare, cheia principală şi, eventual, cheile externe.

Analiza unei probleme şi proiectarea bazei de date pornind de la cererile de informații.

Problema 1.1.

La emisiunea «Fotbal-minut cu minut» se solicită telefonic informații variate. Crainicul ar dori o bază de date care să permită obținerea promptă a răspunsurilor la solicitările publicului. Iată o listă cu câteva întrebări:

a. Care sunt jucătorii de fotbal cu vârstă mai mică de 30 ani?
b. Cine locuieşte în altă țară şi nu este căsătorit? Care este adresa stabilă a acestora?
c. Care sunt echipele de fotbal? Când s-au înființat? Care este sediul? Patronul? Care sunt membrii echipei Dinamo în prezent?
d. Cine a mai jucat la echipa Rapid de-a lungul timpului? Pe ce post?
e. Care sunt meciurile planificate azi, unde, la ce oră şi între ce echipe?

f. Câte meciuri au fost câştigate de echipa Rapid?
g. Care este CV-ul lui Mutu? Este căsătorit? Cu cine? Are şi copii?
h. Afişaţi fotografia lui Mutu!

Rezolvare

Pentru analiza problemei trebuie să răspundem la următoarele întrebări:

1. Care sunt entităţile?
   JUCĂTORI, ECHIPE, MECIURI.
2. Care sunt atributele, caracteristicile pentru fiecare entitate?
   JUCĂTORI (cod, nume, data_nasterii, foto, CV, stare civila?, are_copii?, adresa),
   ECHIPE (cod_echipa, nume, data_înfiintarii, sediu),
   MECIURI (cod_meci, echipa_1, echipa_2, goluri_1, goluri_2, stadion, data, ora).
3. Care sunt relaţiile dintre entităţi?
   Un jucător poate să fi fost înscris la mai multe echipe; o echipă are mai mulţi jucători. Un meci este jucat de două echipe; o echipă participă la mai multe meciuri.
4. Cum arată SCHEMA conceptuală a bazei de date?

5. Cum se modelează SCHEMA pentru o bază de date relaţională?
   Relaţia JUCĂTORI→ECHIPE este spartă în două relaţii prin intermediul entităţii PARTICIPANŢI (codul jucatorului, codul echipei, data_intrarii_in_echipa, data_iesirii_din_echipa, functia).
   Relaţia ECHIPE→MECIURI va fi modelată ca atare prin reţinerea a două câmpuri legate de echipe (echipa 1 este cea care joacă „acasă”, iar echipa 2 este cea din „deplasare”). Fiecare entitate şi relaţie se structurează sub foma unui tabel. Obţinem deci următoarele tabele:

JUCĂTORI

Id	Nume	Data_n	Casat	Adr	Copii	Foto	cv
100	Mutu	01.12.1976	T	Bucureşti	T		
101	Popescu	03.05.1978	F	Cluj	F		

ECHIPE

Id	Nume	Data_insc	Sediu
1	Steaua	01.01.1800	Bucureşti
2	Dinamo	01.01.1980	Bucureşti

PARTICIPANŢI

Id	Cod_juc	Cod_ech	Data-intr	Data-ies	post
1	100	1	01.01.2000	01.12.2000	Portar
2	100	2	01.01.2001		Fundas
3	101	2	15.07.1980		portar

MECIURI

Id	E1	E2	G1	G2	Stadion	Data	Ora
1	1	2	1	3	Lia Manoliu	01.05.2003	15
2	2	1	0	0	Dinamo	07.05.2003	17

6. Care sunt cheile unice ale fiecărei tabele?
   Pentru simplificarea lucrului am numit un câmp special „Id” care să joace rolul de cheie unică a fiecărui tabel.
7. Care sunt cheile străine (cele care asigură legăturile dintre tabele)?
   Participanți.cod_ech, Participanți.cod_juc – pentru legătura cu tabelele ECHIPE şi JUCĂTORI şi Meciuri.e1, Meciuri.e2 pentru legătura cu tabela ECHIPE.
8. Care sunt restricțiile de integritate a bazei de date?
   a. Un articol în PARTICIPANȚI nu poate fi introdus dacă nu se regăseşte valoarea din câmpul Participant. Cod_juc în Jucatori.id şi valoarea din câmpul Participant.cod_ech în Echipe.id.
   b. Un articol în tabela MECIURI nu poate fi introdus când codurile celor două echipe nu figurează deja în tabela ECHIPE.
   c. Modificarea cheii Echipe.id trebuie să determine modificarea cheilor străine Participant.cod_ech şi Meciuri.E1 sau Meciuri.E2.
   d. Modificarea cheii Jucatori.id trebuie să determine modificarea cheii Participant.cod_juc.
   e. Ştergerea unei echipe – adică scoaterea ei din evidență – nu va determina şi anularea activității jucătorilor prin eliminarea liniilor din tabela PARTICIPANȚI, dar va provoca ştergerea meciurilor planificate cu această echipă (meciurile deja jucate pot rămâne!).
9. Care sunt restricțiile de validitate asupra câmpurilor din tabele?
   Data planificării unui meci trebuie să fie mai mare decât data curentă! Ora planificată trebuie să fie între 8–20.
   În tabela PARTICIPANȚI data_intrarii trebuie să fie mai mică decât data_iesirii unui jucător dintr-o echipă.
10. Care sunt evenimentele care determină acțiuni asupra datelor?
    Planificarea unui meci; desfăşurarea unui meci; înscrierea unui jucător la o echipă; plecarea unui jucător de la o echipă; înscrierea unui nou jucător; înființarea unei noi echipe; schimbarea stadionului sau a datei planificate pentru un meci; modificarea unor valori pentru un jucător; modificarea unor valori pentru echipă etc.

Problema 1.2.

Organizați datele pentru următoarele solicitări de informații:
– Lista tuturor persoanelor care locuiesc la case.
– Lista persoanelor fără ocupație care au domiciliul flotant în localitatea X.
– Care sunt proprietățile unei persoane X (x = numărul de buletin)?
– Situația imobilelor de pe strada X sub forma următorului raport:

Nr. imobil	Tip imobil	Suprafața	Număr apartamente	Număr locatari
	(vilă, bloc, casă)			

– Care sunt apartamentele unde locuiesc copii?
– Câte apartamente sunt proprietate personală în imobilul X?
– Sunt locuințele cu încălzire prin sobe cu lemne în localitatea X?
– Care este numărul mediu de persoane pe o cameră în blocurile de pe strada X?
– Când au fost cumpărate apartamentele proprietate personală ale lui X?

Lecția 2 Prezentarea generală a aplicației MSAccess

- ✓ *Caracteristici SGBDR Access*
- ✓ *Obiectele – proprietăți, evenimente, metode*
- ✓ *Tipuri de date şi operații specifice fiecărui tip*

Microsoft Access este un SGBDR ce pune la dispoziția utilizatorilor *aplicații complexe* care să ajute la crearea şi manipularea bazelor de date şi la obținerea rapoartelor pe baza acestor date.

Microsoft Access dispune de un *limbaj de programare* procedural foarte puternic şi flexibil, *limbaj orientat pe obiecte*, prin care programatorii îşi pot descrie datele şi aplicațiile: Visual Basic for Applications. De asemenea, are implementat *limbajul de cereri SQL* (AccessSQL) pentru utilizatorii neinformaticieni. De asemenea, pune la dispoziția programatorilor nu numai un compilator şi un mecanism performant de accesare a datelor, ci şi un set de utilitare puternice pentru proiectare, încorporate într-un *mediu integrat şi omogen*. Acest mediu este foarte confortabil pentru proiectanții de aplicații. Microsoft Access recunoaşte şi se adaptează automat la *mediile multiutilizator,* fără a fi nevoie de o variantă specială pentru rețea. Microsoft Access permite *comunicarea cu alte aplicații* (de exemplu, Excel, Word) prin mecanisme **DDE** (transferul dinamic de date). Microsoft Access *are facilitatea OLE* (legarea şi încorporarea de obiecte cum sunt sunetele, imaginile, obiecte spreadsheet etc.). Microsoft Access *importă şi exportă date în/din alte formate* pe diferite suporturi, local sau la distanță. Microsoft Access permite *programarea vizuală a aplicațiilor* folosind asistenți *wizard*, *builder-e* şi *designer-e* pentru proiectarea interactivă a tuturor obiectelor cu care lucrează o aplicație.

Microsoft Access este o aplicație Windows şi, de aceea, interfața sa verifică toate aspectele unei interfețe clasice Windows: lucrează cu ferestre, în mod grafic, multitasking, într-o manieră condusă de evenimente.

Programul Access se poate găsi într-un folder ce conține toate aplicațiile Microsoft Office. Deschiderea aplicației se poate face prin Programs→Microsoft Office→Access.

Încheierea sesiunii de lucru se face prin butonul Close al ferestrei Access.

Fereastra sistem Access are meniul principal pe prima linie, o bară de butoane, *linia de stare* pe care sunt trecute informații legate de obiectul şi acțiunea executată la un moment dat.

2.1. Obiecte, proprietăți si metode

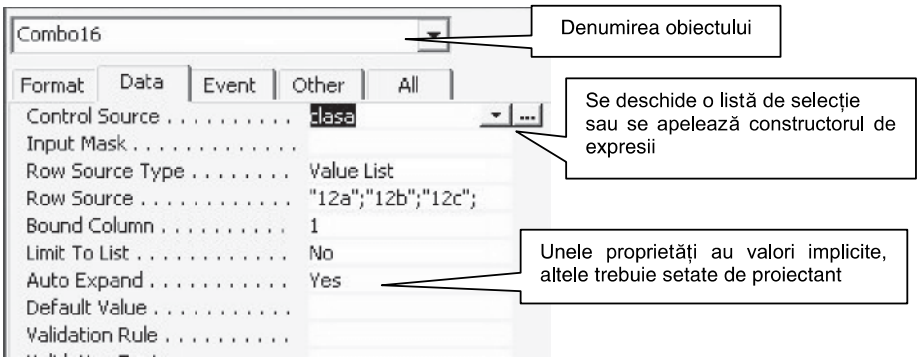
În Microsoft Access bazele de date, tabelele, câmpurile din tabele, interogările, rapoartele şi formularele reprezintă obiecte. Unui obiect îi putem defini anumite *proprietăți,* îi putem inventaria *evenimentele* care pot acționa asupra lui şi *acțiunile* prin care respectivul obiect răspunde la aceste evenimente sau pe care le are incluse în funcționalitatea sa.

O *proprietate* este un atribut al lucrurilor pe care îl stabilim pentru a defini una dintre caracteristicile acestuia, ca aspect sau comportament. De exemplu, o fereastră

poate fi caracterizată prin: textul explicativ ca titlu al ferestrei, distanța în pixeli față de marginea din stânga a ecranului, distanța față de marginea superioară a ecranului, înălțimea, starea vizibilă sau ascunsă etc.

Fiecare obiect Access are definite toate proprietățile într-o fereastră specială **Properties,** care se deschide din meniul **View** sau din meniul contextual ataşat obiectului. Utilizatorul poate seta anumite proprietăți la valorile dorite.

Fereastra de proprietăți



Obiectele recunosc şi răspund la anumite acțiuni numite evenimente.

Un *eveniment* este o activitate specifică şi predeterminată, inițiată de sistemul de operare (eveniment intern) sau de utilizator (eveniment extern) şi la care un obiect ştie să reacționeze (de exemplu: apăsarea pe butonul mouse-ului, mişcarea mouse-ului, apăsarea pe o tastă sau întâlnirea unei erori la introducerea datelor într-un textbox). Unui eveniment i se poate ataşa o anumită procedură (este numită chiar Event procedure) sau o macroinstrucțiune în limbajul Visual Basic pus la dispoziție de Access. *Metodele* sunt deci proceduri asociate unui obiect. De exemplu, o metodă asociată pentru selectarea ferestrei poate să schimbe culoarea de fond, să-i schimbe poziția etc.

Obiectele pot să difere între ele prin proprietăți. Astfel, două declanşatoare pot avea dimensiuni diferite, poziții diferite pe ecran, texte explicative diferite etc., dar se pot comporta asemănător (răspund la aceleaşi evenimente prin aceleaşi acțiuni). Spunem că aparțin aceleiaşi *clase sau colecții de obiecte.*

2.2. Tipuri de date şi funcții uzuale

SGBDR Access permite folosirea datelor numerice, şiruri de caractere, date calendaristice şi valori logice (booleene). De asemenea, pot fi încorporate sau doar legate la câmpurile unei tabele, obiecte ca documente word, imagini sau sunete. O altă facilitate este posibilitatea includerii unei legături (Hyperlink) pentru accesarea directă din tabela de date a site-ului web dorit.

Vom prezenta principalele tipuri de date, operatorii specifici şi câteva funcții standard (interne) uzuale:

• **Date tip şir de caractere**

Sunt acceptate caracterele ASCII delimitate prin ghilimele şi sunt recunoscute ca tip text (la câmpuri) sau string (la variabile). *Exemplu:* “alfa”.

Operatori:

❶ de concatenare: &, +;

❷ de relaționare: <,<=,>,>=,<>,=, Like;

Exemplu: “alfa”&“bet” – expresie text având ca rezultat şirul “alfabet”
“alfa”>“alfabet” – expresie logică având valoarea False/NO

Funcții uzuale pentru şiruri:

1. **LEN**(sir) – lungimea şirului.
   Exemplu: len(“maria” & “ana”) întoarce valoarea 8.
2. **STR**(numar) – face conversia numărului la şir.
   Exemplu: Str(123) întoarce “123”.
3. **MID**(sir,start,lung) – extrage un subşir de lungime lung din şir, începând de la poziția start. *Exemplu:* Mid(“alfabet”,3,4) întoarce “fabe”.
4. **LEFT**(sir,lung) – extrage primele lung caractere din şir. *Exemplu:* Left(“alfa”,2) întoarce “al”.
5. **RIGHT**(sir,lung) – extrage ultimele lung caractere din şir.
   Exemplu: Right(“alfa”,2) întoarce “fa”.
6. **VAL**(sir) – face conversia şirului la număr. *Exemplu:* Val(“123”) întoarce 123.
7. **TRIM**(sir**)** – elimină spațiile situate în exteriorul şirului.
   Exemplu: trim (“ alfa ”) întoarce “alfa”.

• **Date numerice**

Sunt recunoscute tipurile Byte, Integer, Long (pentru valorile întregi), Single, Double (pentru valorile reale) şi Currency (pentru valori monetare).

Operatori:

❶ aritmetici: +,-,*,/ (împărțire reală), \ (împărțire întreagă), mod (restul), ^ (ridicare la putere);

❷ de relații: <,<=,>,>=,<>,=, Between.

Exemplu: x Between 4 and 10: expresie logică cu valoare Yes dacă x este între 4 şi 10
.123 constantă numerică interpretată ca 0.123
1e3 + 123.2 expresie numerică cu valoarea 1123.2
123.008$ valoare monetară

Funcții uzuale pentru date numerice:

1. **INT**(numar) – întoarce partea întreagă. *Exemplu:* int(12.45) întoarce 12;
2. **ABS**(numar) – valoarea absolută. *Exemplu:* Abs(-15) întoarce 15;
3. **RND**() – întoarce un număr aleator subunitar;
4. **ISNUMERIC**(exp) – testează dacă expresia este numerică.

• **Date calendaristice**

Acestea rețin anul, ziua, luna şi eventual ora, minutul şi secunda. Delimitator este caracterul diez. Aparțin tipului DateTime (la câmpuri) sau Date (la variabile).

Operatori: ❶ specifici: +, - (adunarea, scăderea unui număr de zile)
❷ relaționali: <,<=,>,>=,<>,= , Between

Exemplu: # 1/12/2001# dată calendaristică în format ll/zz/aaaa: 12 ianuarie 2002;
5/3/2002 4:30:05# dată calendaristică ce include şi ora;
Date() +3 expresie de tip dată calendaristică: data curentă plus 3 zile;
[Data-nastere] between # 3-ian-1952 # and 3-dec-2000.

Funcții standard uzuale pentru date calendaristice:

1. **DATE**() – întoarce data curentă (sistem);
2. **NOW**() – întoarce data şi ora sistem;
3. **ISDATE**(exp) – testează dacă expresia este o dată calendaristică.

• **Date logice**

Folosite pentru date care pot lua doar valorile adevărat sau fals; aparțin tipului Yes/No (la câmpuri) sau Boolean (la variabile).

Operatorii logici sunt Not, And, Or, Xor, Eqv, Imp (implică).

Yes, true, No, false = constante logice;

Exemplu: [Pret]<5e6 and [cant] >100 **–** expresie logică cu valoarea Yes dacă prețul este mai mic ca 5 milioane şi cantitatea mai mare decât 100.

Funcții standard comune tuturor tipurilor de date:

1. **IIF**(expresie logică, expresie1, expresie2) – întoarce expresie1 dacă primul parametru este adevărat, sau expresie2 dacă acesta este fals.

Exemplu: iif(varsta>18,“major”,“minor”) va întoarce şirul “minor” dacă varsta=6.

2. **FORMAT**(expresie,sablon) – întoarce expresia sub o anumită formă indicată de şablon.

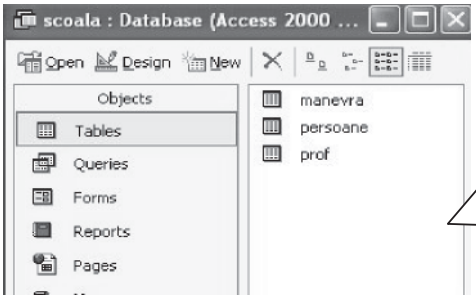
Exemplu: Format(#6/25/98#,“mmm dd”,“yyyy”) va întoarce June 25, 1998. Şablonul este un şir de caractere de cod. Se foloseşte pentru expresii numerice, şiruri, date calendaristice.

Baza de date Access

Orice activitate în Access se desfăşoară asupra unei baze de date, deci prima operație pe care o face un utilizator când intră în mediul Access este să deschidă baza de date – dacă aceasta există – sau să creeze una nouă.

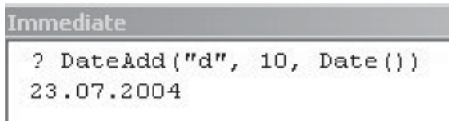
1. **Crearea unei baze de date** se face prin File→New→Blank Database. Se va deschide fereastra Database Design.
2. **Deschiderea unei baze de date** se face prin File→Open**.**
3. **Închiderea bazei de date** se face odată cu închiderea ferestrei asociate sau prin File→Close Database**.**
4. **Ştergerea unei baze de date** se face prin ştergerea fişierului .Mdb care o conține.

Toate activitățile se desfăşoară având deschisă fereastra bazei de date DataBase Window care permite atât crearea, deschiderea sau închiderea tuturor obiectelor ei, cât şi evidența acestor operații.



Observați în imagine fereastra Database Window deschisă pentru baza de date Scoala. În stânga sunt butoanele pentru selectarea clasei de obiecte dorite, iar în dreapta se găsesc instanțe ale obiectelor. Baza de date Access conține mai multe categorii de obiecte: tabele, interogări, rapoarte, formulare, macro-uri şi module în VBA.

Pentru fixarea efectului câtorva funcții va trebui să creați/deschideți o bază de date şi să apelați editorul VBA(Visual Basic) prin obiectul modules→new. Apoi deschideți fereastra **Immediate** prin meniul View→Immediate window. Comanda **?** va permite afişarea efectului funcției. Observați captura ferestrei pentru funcția DateAdd.



Încercați funcțiile şi expresiile din cele două coloane; notați efectul:

```
Dateadd("d",10,date())                  Mid("alba ca zapada", 1,4)
Date() - #24.01.1952#                   Left("alba ca zapada",4)
Dateadd("m",3,date())                   Right("alba ca zapada",4)
Dateadd("yyyy",2,date())                Format(#6/25/97#,"mmm"","" dd")
dateDiff("m", Date, „2/2/2004")         Format(123.45, "(999)-999.999")
month(date())                           Time()+"*"
dateDiff("yyyy",Date, „2/2/2004")       Len("albatros")
year(date()) - 1990                     Format( 1.5, "000.000")
day(date()) +2                          Format( 1234.5, „00")
Iif(month(date()=1,"ianuarie","eroare" Format( #6/25/97#,"yyyy mmm")
Now()                                   Format( 1.5895, "#.00")
#24.01.1952# + date()                   Val("012")
Isdate(#12/12/2000#)                    Str(123.45)
Trim("albatros")                        Format( #6/25/97#,"mmm")
Format( 1.5, "00")                      Format( 1.5, "##")
```

Întrebări recapitulative

1. Cum este corect să spunem despre aplicația Access că este «o bază de date» sau că este un «SGBD»?

2. Care pot fi proprietățile unei ferestre? La ce acțiuni răspunde o fereastră?

3. Ce operații pot fi efectuate cu şiruri, numere, date calendaristice?

4. Care sunt obiectele gestionate de aplicația Access?

Lecția 3 Proiectarea tabelelor Access

- ✓ *Tipuri de date dintr-o tabelă*
- ✓ *Proiectarea structurii unei tabele în diferite moduri de lucru*

O bază de date Access are ca model conceptual modelul relațional. Putem astfel regăsi toate caracteristicile de bază discutate în capitolul precedent: o tabelă este o colecție de date legate între ele, memorate sub formă de înregistrări compuse din unul sau mai multe câmpuri. Ca regulă generală, un utilizator oarecare are acces la tabela de date prin intermediul unor formulare, interogări şi/sau rapoarte. Desigur, pentru proiectantul bazei de date, Access oferă posibilitățile de acces direct la structură şi la conținut.

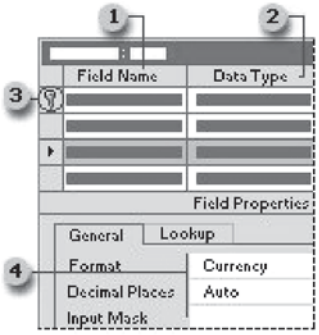
Definirea structurii tabelelor se face din fereastra Database →Table →New, în trei moduri de lucru: Design, Datasheet şi Wizard. După salvarea informațiilor de structură se pot introduce, vizualiza sau edita date prin Database→Table→Open. Reluarea proiectării se permite prin Database→Table→Design.

3.1. Mediul de lucru

Pentru operațiile directe asupra tabelelor beneficiem de un obiect de interfață deosebit de util – bara de butoane **Table Design** care conține, printre alte butoane necunoscute utilizatorilor mediului Windows, şi câteva butoane specifice lucrului cu tabele.

❶ Primul buton permite trecerea de la un mod de lucru la altul: de la introducerea de date (Datasheet) la proiectarea structurii (Design), a unei interogări (PivotTable) sau a unui grafic (PivotChart) etc.
❷ Butonul Indexes permite deschiderea ferestrei pentru editarea indexurilor.
❸ Butonul Primary Key permite specificarea/ştergerea cheilor primare.
❹ Butonul Insert Rows permite inserarea noilor rânduri (câmpuri în structură).
❺ Butonul Delete Rows permite ştergerea câmpurilor selectate.
❻ Butonul Properties deschide fereastra de proprietăți asociată tabelei.
❼ Butonul Relationships deschide fereastra de proiectare/editare a legăturilor dintre tabele.
❽ Butonul Show Table permite adăugarea/ştergerea tabelelor în fereastra Relationships.
❾ Butonul Database Window permite trecerea la fereastra principală a bazei de date.
❿ Butonul Builder deschide utilitarul Table Wizard pentru construirea rapidă a structurii, prin selectarea câmpurilor din tabelele predefinite.

Crearea tabelelor în modul Design View



Modul Design View presupune introducerea informațiilor de structură de către utilizator: numele, tipul şi proprietățile fiecărui câmp, precum şi cheia principală a tabelei.

1. **Numele** câmpului – poate fi orice şir de maxim 64 de caractere. O descriere mai largă a câmpului, care va fi afişată pe linia de stare a ferestrei principale Access atunci când se introduc date în câmpul respectiv, poate fi dată în coloana Description.
2. **Tipul** câmpului se alege din lista deschisă prin butonul DataType.

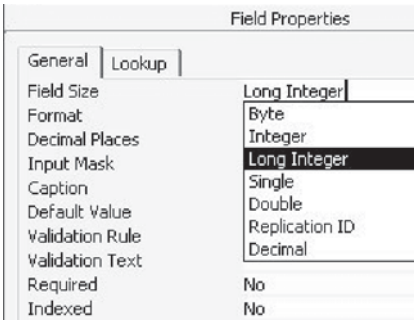
Text	Pentru şiruri de maxim 255 caractere. Delimitatori: ghilimelele. Se poate fixa altă lungime prin proprietatea Size:
Number	Pentru numere întregi sau zecimale. Se poate particulariza. *Exemplu:* tipul Byte permite reținerea valorilor de la 0 până la 255 şi ocupă un byte memorie.
Currency	Pentru valori monetare. Folosiți tipul currency pentru a evita rotunjirea în timpul calculelor. Tipul currency oferă o precizie de 15 cifre întregi şi 4 zecimale.
Date/Time	Pentru date calendaristice. Se poate fixa formatul dorit.
Yes/No	Pentru datele logice. Constantele logice sunt Yes, No.
Memo	Pentru şiruri cu lungime mai mare decât 65536 caractere.
AutoNumber	Pentru serii incrementate automat cu 1 sau valori aleatoare. De obicei este folosit pentru cheia principală a tabelei.
OLE Object	Pentru legarea/încorporarea obiectelor prin mecanismul OLE: documente Word, imagini, fişiere de sunet, tabele Excel etc.
Hyperlink	Pentru reținerea adreselor URL. *Exemplu:* Pagina_scolii # http://www.liis.edu.ro#evenimente
Lookup Wizard	Pentru selectarea valorilor posibile dintr-o listă de constante sau dintr-un câmp al unei tabele.

3. **Cheia principală a tabelei** – se declară astfel:
   - se poziționează cursorul pe câmpul care dorim să devină cheia tabelei.
   - din meniul Edit activăm opțiunea Primary Key sau din bara de instrumente apăsăm pe butonul  ;
   - urmărim apariția aceluiaşi simbol în dreptul câmpului care va avea rolul de cheie unică pentru tabelă – în cazul nostru câmpul Cod.

Dacă se folosesc mai multe câmpuri pentru cheia primară trebuie selectate toate şi apoi apăsat butonul Primary Key. Dacă ați salvat tabela fără să fixați cheia primară, Access vă va avertiza despre acest aspect şi vă va da posibilitatea de a reveni în modul de proiectare Design, sau va ataşa automat un câmp cu rol de cheie, de tip AutoNumber, numindu-l ID.

De fiecare dată când se deschide tabela, articolele sunt ordonate după câmpul cheie primară.

4. Proprietățile câmpului:

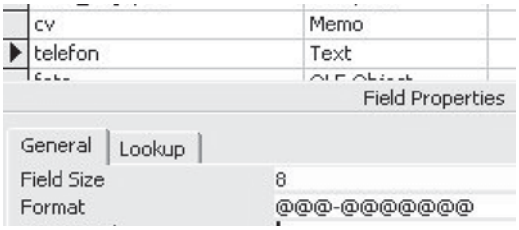


a. Field size – permite fixarea mărimii zonei alocate coloanei. Implicit, pentru texte se rezervă 50 de caractere. La tipul numeric se asociază lungimea în funcție de varianta folosită: Byte, Double etc.

b. Format – **determină modul în care va fi afişat câmpul.** Mai multe tipuri de date dispun de formate predefinite.

Exemplu:

Vrem ca numărul de telefon să fie automat împărțit în 2 zone printr-o linie (de exemplu: 032-22445678). Vom folosi tipul text şi la format vom plasa linia pe poziția dorită. Simbolul @ înlocuieşte orice caracter.



c. Input mask permite introducerea unui şablon care să controleze modul de introducere a datelor. Se folosesc simboluri ca 0 sau 9 care obligă la introducerea unei cifre, L obligă la introducerea unei litere etc.

d. Decimal Places – precizarea numărului de zecimale. Valoarea Auto va indica faptul că numărul de zecimale este dat de proprietatea Format.

e. Default Value – permite precizarea valorii inițiale, în câmpul respectiv, pentru un nou articol.

f. Caption – denumirea sub care va apărea coloana în formularele sau rapoartele utilizator. Implicit este trecut numele câmpului.

g. Validation Rule – permite specificarea unei condiții la introducerea valorii.

Exemple de reguli de validare:

`<> 0`	Intrarea trebuie să fie diferită de zero.
`> 1000 Or Is Null`	Intrarea trebuie să fie mai mare ca 1000 sau vidă.
`Like «A????»`	Intrarea trebuie să fie pe 5 caractere şi să înceapă cu litera “A”.
`>= #1/1/96# And <#1/1/97#`	Intrarea să fie o dată calendaristică a anului 1996.

g. Validation Text – specifică mesajul afişat în cazul introducerii valorilor eronate.

h. Required – specifică acceptarea sau nu a valorilor nule în coloană.

O valoare nulă sau vidă este o valoare zero (pentru coloana de tip numeric sau currency), şirul vid “ ” (tip text, Memo), data vidă #//# (tip date), No (tip boolean Yes/No). Se asociază implicit dacă nu se precizează altă valoare ca proprietate Default Value.

Dacă într-un câmp se doreşte memorarea valorilor nenule, trebuie setată proprietatea **Required** pe valoarea **Yes.** Dacă se permit şi valori nule, atunci se setează proprietatea **Required** pe valoarea **No**.

Implicit, pentru câmpurile definite drept cheie primară se interzice existența valorilor nule.

i. Allow Zero Length – permite tratarea unui şir de lungime zero drept valoare validă sau nu. Se foloseşte pentru tipul Text sau Memo. Această setare este independentă de valoarea proprietății Required.

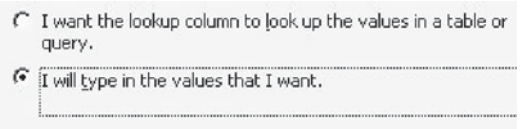
j. **Indexed** – indică prezența sau absența unui index pentru coloană. Pe valoarea NO (No Duplicates) valorile permise sunt unice, iar pe YES (Duplicates) valorile coloanei sunt oarecare.

k. **New Value** – este o proprietate a tipului autonumber şi arată modul cum va fi generată noua valoare: prin incrementare (Increment) sau aleator (Random).

✓ **Definirea câmpurilor de căutare – Lookup Wizard**

Uneori este necesar ca la introducerea datelor într-un câmp să avem o listă de valori din care să putem alege valoarea dorită. Elementele listei pot fi introduse manual de către proiectant odată cu proiectarea structurii tabelei sau pot fi preluate automat dintr-un câmp al unei alte tabele părinte sau dintr-o interogare. De exemplu, dorim ca în tabela ELEVI să putem alege clasa fiecărui elev dintr-o listă. Lista este închisă şi se deschide la plasarea cursorului şi execuția unui click-mouse pe câmp. După selectarea unei valori şi trecerea cursorului pe altă coloană se închide lista.

✓ **Definirea tipului Lookup –** se realizează în fereastra de proiectare a tabelei Design. Pentru câmpul dorit cu acest tip se selectează din listă Lookup Wizard – asistentul care va construi lista de căutare în mai mulți paşi:



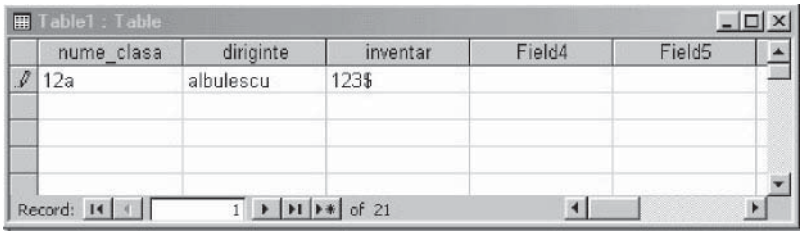
Pasul 1. Se cere proiectantului indicarea modului de completare cu valori a listei. Vom adăuga varianta de introducere manuală.

Pasul 2. Se completează lista cu valori. Este de preferat să introducem o listă ordonată – mai ales pentru listele lungi – astfel încât să poată fi rapid localizată valoarea dorită.

Pasul 3. Se confirmă coloana fixată ca fiind de tip căutare şi se salvează definiția coloanei atribuindu-i un nume. Se ataşează câmpului un obiect tip listă închisă – Combo Box.

3.3. Crearea unei tabele în modul Datasheet View

Modul datasheet permite realizarea simultană a două obiective: crearea structurii tabelei şi popularea ei cu date. Se aseamănă cu foaia de calcul Excel: fiecare linie reprezintă un articol, iar fiecare coloană reprezintă un câmp în structura tabelei.



Se afişează o tabelă vidă, coloanele fiind numite inițial field1, field2, … Se introduc date în aceste câmpuri şi, în funcție de valorile introduse, Access va asocia coloanei tipul corespunzător. Coloanele lăsate libere – fără valori – vor fi şterse la