



**Program postuniversitar de conversie profesională
pentru cadrele didactice din mediul rural**

Specializarea TEHNOLOGIA INFORMAȚIEI

Forma de învățământ ID - semestrul I

**BAZE DE DATE
ȘI
UTILIZAREA ACESTORA**

Adriana OLTEANU

Magdalena ANGHEL

Radu Nicolae PIETRARU

Ministerul Educației și Cercetării
Proiectul pentru Învățământul Rural

TEHNOLOGIA INFORMAȚIEI

Baze de date și utilizarea acestora

Adriana OLTEANU

Radu Nicolae PIETRARU

Magdalena ANGHEL

2005

© 2005

**Ministerul Educației și Cercetării
Proiectul pentru Învățământul Rural**

**Nici o parte a acestei lucrări
nu poate fi reprodusă fără
acordul scris al Ministerului Educației și Cercetării**

BAZE DE DATE ȘI UTILIZAREA ACESTORA

Cuprins	Pagina
INTRODUCERE	vi
UNITATEA DE ÎNVĂȚARE NR. 1 - Problematica organizării informațiilor în mediul electronic	
Obiectivele unității de învățare nr. 1	2
1.1. Introducere	2
1.2. Ce este o bază de date?	3
1.3. Clasificarea sistemelor de baze de date	5
1.3.1. Clasificare după modelul de date	5
1.3.2. Clasificare după numărul de utilizatori	7
1.3.3. Clasificare este cea după numărul de stații pe care este stocată baza de date	8
1.4. Securitatea și protecția datelor în bazele de date	8
Lucrare de verificare a cunoștințelor	10
Răspunsuri și comentarii la întrebările din testele de autoevaluare	11
Bibliografie	12
UNITATEA DE ÎNVĂȚARE NR. 2 – Sistemul de baze de date. Concepte și arhitectură	
Obiectivele unității de învățare nr. 2	14
2.1. Componentele unui sistem de baze de date	14
2.1.1. Hardware	14
2.1.2. Software	15
2.1.3. Utilizatorii	16
2.1.4. Date persistente	16
2.2. Arhitectura internă a sistemelor de baze de date. Modele de date, scheme și instanțe	17
2.3. Independența datelor	19
2.4. Limbaje SGBD	20
2.5. Interfețe SGBD	21
Interfețe bazate pe meniuri	
Interfețe grafice	

Interfețe bazate pe forme	
Interfețe în limbaj natural	
Interfețe specializate aferente cererilor repetate	
Interfețe pentru administratorii bazelor de date	
2.6. Exemple de SGBD	22
Lucrare de verificare a cunoștințelor	24
Răspunsuri și comentarii la întrebările din testele de autoevaluare	25
Bibliografie	27
UNITATEA DE ÎNVĂȚARE NR. 3 – Proiectarea bazelor de date	
Obiectivele unității de învățare nr. 3	28
3.1. Ce este proiectarea?	29
3.2. Modelul entitate-relație. Obiectele bazelor de date relaționale	32
Tabelă (relație)	
Câmp (atribut)	
Înregistrare (nuplu)	
3.3. Construcția schemelor relație	34
3.3.1. Relația unul-la-unul (1-1 sau one to one)	34
3.3.2. Relația unul-la-multe (1-N sau one to many)	35
3.3.3. Relația multe-la-multe (M-N sau many to many)	36
3.3.4. Relația unară	37
3.4. Diagrama entitate-relație	38
3.5. Constrângeri de integritate	39
3.5.1. Constrângerile de domeniu	40
3.5.2. Constrângerile referitoare la n-upluri (înregistrările din tabelă)-	41
Cheia primară	
3.5.3. Constrângeri între relații	41
3.6. Dependente funcționale	44
3.7. Normalizare. Forme normale.	44
3.7.1. Forma normală de ordin 1 (FN1)	45
3.7.2. Forma normală de ordin 2 (FN2)	46
3.7.3. Forma normală de ordin 3 (FN3)	46
3.7.4. Forma normală Boyce-Codd (FNBC)	47
3.8. Structuri de indecși în tabelele de date	48
3.8.1. Indexul primar	50
3.8.2. Indexul secundar	51
3.8.3. Indexul de grup	52
3.8.4. Indexul multinivel	53

Lucrări de verificare a cunoștințelor	55
Răspunsuri și comentarii la întrebările din testele de autoevaluare	56
Bibliografie	57
UNITATEA DE ÎNVĂȚARE NR. 4 – Un limbaj pentru bazele de date relaționale (SQL)	
Obiectivele unității de învățare nr. 4	61
4.1. Introducere	62
4.1.1. Deschiderea și închiderea aplicației Microsoft Access	62
4.1.2. Crearea unei baze de date noi	63
4.1.3. Închiderea unei baze de date	63
4.2. Tipuri de date MICROSOFT Access	64
4.3. Operatorii logici	66
4.4. Limbajul standard SQL	68
4.4.1. Scurt istoric al limbajului SQL	68
4.4.2. Crearea unei tabele	69
4.4.3. Salvarea unei tabele	72
4.4.4. Ștergerea unei tabele	73
4.4.5. Modificarea structurii unei tabele	74
4.5. Modificarea datelor în SQL	74
4.5.1. Inserarea de noi linii într-o tabelă	74
4.5.2. Ștergerea unor linii dintr-o tabelă	75
4.5.3. Modificarea unor linii dintr-o tabelă	77
4.6. Limbajul de cereri în SQL	79
4.6.1. Cereri simple	79
Expresii aritmetice	
Alias de coloană	
Constante (literali)	
4.6.2. Clauza DISTINCT	84
4.6.3. Clauza ORDER BY	85
4.6.4. Clauza WHERE	86
Operatorul BETWEEN	
Operatorul IN	
Operatorul IS NULL	
Operatorul LIKE	
4.6.5. Funcții de grup	90
4.6.6. Clauza GROUP BY	92
4.6.7. Clauza HAVING	93
4.6.8. Cereri conținând mai multe tabele	94
Lucrări de verificare a cunoștințelor	99

Răspunsuri și comentarii la întrebările din testele de autoevaluare	101
---	-----

Bibliografie	102
--------------	-----

UNITATEA DE ÎNVĂȚARE NR. 5 – Construirea interfețelor cu ajutorul formularelor în Microsoft Access

Obiectivele unității de învățare nr. 5	104
--	-----

5.1. Ce este un formular?	105
---------------------------	-----

5.2. Lucrul cu formularele	105
----------------------------	-----

5.2.1. Deschiderea unui formular	105
----------------------------------	-----

5.2.2. Crearea unui formular	105
------------------------------	-----

5.2.3. Utilizarea unui formular	109
---------------------------------	-----

pentru a introduce și a modifica date în tabelă

5.2.4. Parcurgerea înregistrărilor utilizând formularele	110
--	-----

5.2.5. Adăugarea și modificarea textului în antet și subsol	110
---	-----

5.2.6. Ștergerea unui formular	112
--------------------------------	-----

5.3. Salvarea și închiderea unui formular	112
---	-----

Lucrare de verificare a cunoștințelor	113
---------------------------------------	-----

Răspunsuri și comentarii la întrebările din testele de autoevaluare	114
---	-----

Bibliografie	114
--------------	-----

UNITATEA DE ÎNVĂȚARE NR. 6 – Rapoarte în Microsoft Access

Obiectivele unității de învățare nr. 6	116
--	-----

6.1. Ce este un raport?	117
-------------------------	-----

6.2. Lucrul cu rapoarte	117
-------------------------	-----

6.2.1. Deschiderea unui raport	117
--------------------------------	-----

6.2.2. Crearea unui raport	117
----------------------------	-----

6.2.3. Adăugarea și modificarea textului în antet și subsol	123
---	-----

6.2.4. Ștergerea unui raport	125
------------------------------	-----

6.3. Salvarea și închiderea unui raport	125
---	-----

Lucrare de verificare a cunoștințelor	125
---------------------------------------	-----

Răspunsuri și comentarii la întrebările din testele de autoevaluare	127
---	-----

Bibliografie	127
--------------	-----

BAZE DE DATE ȘI UTILIZAREA ACESTORA

INTRODUCERE

Stimate cursant,

Încă de la început doresc să îți urez bun venit la studiul cursului destinat studierii bazelor de date și utilizării acestora. Acest curs se adresează atât cursanților cu un grad mai mare de familiarizare cu universul gestionării datelor, cât și cursanților începători. Acest curs este un curs introductiv în proiectarea și utilizarea bazelor de date.

Există totuși anumite cunoștințe legate de utilizarea calculatorului necesare parcurgerii acestui curs, cum ar fi: cunoașterea modalității de organizare și manipulare a informației în format electronic, operații de bază privind sistemul de operare Microsoft Windows și utilizarea tastaturii PC standard și a mouse-ului.

Manualul de față este organizat în 6 unități de învățare, fiecare dintre aceste unități conținând o parte de prezentare teoretică a subiectului tratat, o parte de exemple, teste de autoevaluare și rezolvările acestora, precum și lucrări de verificare a cunoștințelor.

Cele 6 unități de învățare sunt prezentate gradat, începând cu noțiuni introductive despre baze de date, clasificarea lor, până la concepte ale bazelor de date, proiectarea bazelor de date, folosirea limbajului de interogare a bazelor de date. La finalul manualului vom studia împreună construirea interfețelor și realizarea rapoartelor în mediul Microsoft Access.

La începutul fiecărei unități de învățare vor fi detaliate obiectivele propuse în respectiva unitate.

În cadrul fiecărei unități de învățare există câte o lucrare de verificare, care cuprinde mai multe întrebări. Lucrările de verificare sunt poziționate la sfârșitul fiecărei unități de învățare, rezolvarea problemelor propuse din lucrările de verificare fiind asemănătoare cu cea din exemple din cadrul unității de învățare respective.

În cadrul fiecărei unități de învățare există teste de autoevaluare, acestea fiind necesare pentru a fixa cunoștințele dobândite în fiecare capitol și pentru a permite evaluarea continuă a cursantului. Răspunsurile la testele de evaluare se vor completa în spațiile speciale din cadrul manualului.

De-a lungul modului „Baze de date și utilizarea acestora” există 22 de teste de autoevaluare, care cuprind 28 de întrebări și 6 lucrări de verificare a cunoștințelor, care cuprind 43 de probleme. Problemele din testele de autoevaluare sunt punctate cu 0,5 puncte, fiind considerate evaluare pe parcurs, iar cele din lucrările de verificare

sunt punctate cu 2 puncte, aceste puncte fiind considerate evaluare finală, astfel:

-în unitatea 1 sunt 4 întrebări în testele de autoevaluare, deci se obțin 2 puncte și 2 întrebări la lucrarea de verificare unde se obțin 4 puncte.

-în unitatea 2 sunt 7 întrebări în testele de autoevaluare, deci se obțin 3,5 puncte și 1 întrebare la lucrarea de verificare unde se obțin 2 puncte.

-în unitatea 3 sunt 7 întrebări în testele de autoevaluare, deci se obțin 3,5 puncte și 6 întrebări la lucrarea de verificare unde se obțin 12 puncte.

-în unitatea 4 sunt 4 întrebări în testele de autoevaluare, deci se obțin 2 puncte și 27 întrebări la lucrarea de verificare unde se obțin 54 puncte.

-în unitatea 5 sunt 3 întrebări în testele de autoevaluare, deci se obțin 1,5 puncte și 3 întrebări la lucrarea de verificare unde se obțin 6 puncte.

-în unitatea 6 sunt 3 întrebări în testele de autoevaluare, deci se obțin 1,5 puncte și 4 întrebări la lucrarea de verificare unde se obțin 8 puncte.

Însumate punctele obținute de cursanți de-a lungul semestrului sunt 100.

Lucrările de verificare vor fi transmise tutorelui modului într-un fișier separat.

Bibliografia minimală a acestui modul este:

- Cârstoiu, Dorin, *Baze de date relaționale*, Editura Printech, 1999
- Rădulescu, Florin, *Baze de date în Internet*, Editura Printech, 2000
- Ionescu, Felicia, *Baze de date relaționale și aplicații*, Editura Tehnică, 2004
- Baltac, Vasile, *ECDL-Excel, Access, PowerPoint în 20 lecții și 75 de simulări*, Editura Andreco, 2003
- Browne, Allen, Balter Alison, *Bazele Access 95*, Editura Teora, 1999
- Pribeanu, Costin, *Baze de date și aplicații*, Editura MatrixRom, 2000
- Pascu, C., Pascu A., *Totul despre SQL*, Editura Tehnică, 1994

Bibliografia este prezentată și la sfârșitul fiecărei unități de învățare

În cazul în care nu veți reuși să rezolvați problemele propuse trebuie recitite zonele de text care apar înainte de lucrarea de verificare. În speranța că nu vor exista probleme vă urăm:

Spor la treabă!

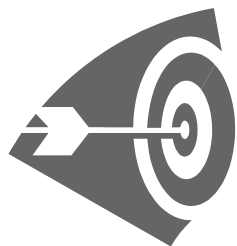
Autorii

Unitatea de învățare Nr. 1

PROBLEMATICA ORGANIZĂRII INFORMAȚIILOR ÎN MEDIUL ELECTRONIC

Cuprins	Pagina
Obiectivele unității de învățare nr. 1	2
1.1. Introducere	2
1.2. Ce este o bază de date?	3
1.3. Clasificarea sistemelor de baze de date	5
1.3.1. Clasificare după modelul de date	5
1.3.2. Clasificare după numărul de utilizatori	7
1.3.3. Clasificare este cea după numărul de stații pe care este stocată baza de date	8
1.4. Securitatea și protecția datelor in bazele de date	8
Lucrare de verificare a cunoștințelor	10
Răspunsuri și comentarii la întrebările din testele de autoevaluare	11
Bibliografie	12

OBIECTIVELE unității de învățare nr. 1



Principalele obiective ale unității de învățare nr. 1 sunt:

După studiul unității de învățare nr. 1 vei fi capabil să demonstrezi că ai dobândit cunoștințe suficiente pentru a înțelege:

- de ce sunt importante în viața de zi cu zi bazele de date
- în ce companii se folosesc bazele de date
- ce categorii de operații se pot realiza asupra datelor din baza de date
- ce reprezintă efectiv o bază de date
- ce avantaje îți oferă utilizarea bazelor de date
- clasificarea bazelor de date
- care sunt caracteristicile fiecărui model de baze de date
- de ce este importantă protecția și securizarea datelor din baza de date
- diferite moduri de protejare a datelor

1.1. Introducere

În ultimii ani, dezvoltarea sistemelor de baze de date reprezintă unul dintre cele mai importante aspecte în domeniul tehnologiei informației, având un impact decisiv asupra modului de organizare și funcționare a numeroaselor instituții și servicii. Acestea sunt companiile de comunicație, întreprinderile de comerț, serviciile bancare, serviciile de transport, asigurările, universitățile etc. Acestea sunt dependente de funcționarea corectă și neîntreruptă a sistemelor de baze de date.

Sistemele de baze de date sunt o componentă importantă a vieții de zi cu zi în societatea modernă. Zilnic, majoritatea persoanelor desfășoară activități care implică interacțiunea cu o bază de date: depunerea sau extragerea unei sume de bani din bancă, rezervarea biletelor de tren sau de avion, căutarea unei cărți într-o bibliotecă computerizată, gestiunea angajaților dintr-o firmă, cumpărarea unor produse etc.

Bazele de date pot avea mărimi (număr de înregistrări) și complexități extrem de variate, de la câteva zeci de înregistrări (de exemplu, baza de date pentru o agendă de telefon a unei persoane)

sau pot ajunge la milioane de înregistrări (de exemplu, baza de date pentru cărțile dintr-o bibliotecă, baza de date cu stocarea angajaților unei firme sau baza de date unde se păstrează informații despre situația studenților etc).

Marea majoritate a sistemelor de baze de date existente în momentul de față sunt relaționale și există un număr mare de astfel de sisteme comerciale care pot fi achiziționate și folosite pentru propriile dezvoltări. Modelul relațional de baze de date a fost introdus în anul 1970 de către E.F.Codd.

- o Utilizatorii unei baze de date au posibilitatea să efectueze mai multe categorii de operații asupra datelor stocate aici:

- Introducerea de noi date (*insert*)
- Ștergerea unor date existente în baza de date(*delete*)
- Actualizarea datelor stocate(*update*)
- Interogarea bazei de date (*query*) pentru regăsirea anumitor informații, selectate după un criteriu ales.



Test de autoevaluare

1. De ce sunt importante bazele de date?

1. 2. Ce este o Bază de Date?

În sensul larg, o bază de date (*database*) este o colecție de date corelate din punct de vedere logic, care reflectă un anumit aspect al lumii reale și este destinat unui anumit grup de utilizatori. În acest sens, bazele de date pot fi create și menținute manual (un exemplu ar fi fișele de evidență a cărților dintr-o bibliotecă, așa cum erau folosite cu ani în urmă) sau computerizat așa cum sunt majoritatea bazelor de date în momentul de față. O definiție într-un sens mai restrâns a unei baze de date este următoarea:



O bază de date este o colecție de date centralizate, creată și menținută computerizat, în scopul prelucrării datelor în contextul unui set de aplicații. Prelucrarea datelor se referă la operațiile de introducere, ștergere, actualizare și interogare a datelor.

Simple colecții de fișe (documente pe hârtie) sau fișiere de date care conțin date, dar nu permit operații de interogare nu sunt considerate baze de date. De exemplu, datele memorate în fișiere pe disc într-o aplicație de calcul tabelar (Microsoft Excel) sau documentele memorate de un editor de texte (ca Microsoft Word) nu sunt considerate baze de date.

Orice bază de date are următoarele proprietăți implicite:

- Baza de date este o colecție logică coerentă de date ce are cel puțin un înțeles
- Baza de date este destinată, construită și populată de date despre un domeniu bine precizat. Ea are un grup de utilizatori și se adresează unui anumit grup de aplicații
- O bază de date reprezintă câteva aspecte ale lumii reale creând orizontul propriu. Schimbările orizontului sunt reflectate în baza de date.

Față de vechile metode de înregistrare a datelor privind diferite activități pe fișe (documente scrise) sau chiar în fișiere pe disc, sistemele de baze de date oferă avantaje considerabile, ceea ce explică extinsa utilizare a acestora. Câteva dintre avantajele oferite sunt:

- *Controlul centralizat al datelor*, putând fi desemnată o persoană ca responsabil cu administrarea bazei de date
- *Viteză mare* de regăsire și actualizare a informațiilor
- *Sunt compacte*: volumul ocupat de sistemele de baze de date este mult mai redus decât documentele scrise
- *Flexibilitatea* ce constă în posibilitatea modificării structurii bazei de date fără a fi necesară modificarea programelor de aplicație
- *Redundanță scăzută* a datelor memorate, care se obține prin partajarea datelor între mai mulți utilizatori și aplicații. În sistemele de baze de date, mai multe aplicații pot folosi date comune, memorate o singură dată. De exemplu, o aplicație pentru gestionarea personalului dintr-o universitate și o aplicație pentru gestionarea rezultatelor la examene din aceeași universitate care folosește o singură bază de date, pot folosi aceleași informații referitoare la structurarea facultăților.
- *Posibilitatea introducerii standardelor* privind modul de stocare a datelor, ceea ce permite interschimbarea datelor între organizații
- *Mentținerea integrității datelor* prin politica de securitate (drepturi de acces diferențiate în funcție de rolul utilizatorilor), prin gestionarea tranzacțiilor și prin refacerea datelor în caz de funcționare defectuoasă a diferitelor componente hardware sau software.

- *Independența datelor* față de suportul hardware utilizat. Sistemul de gestiune a bazelor de date oferă o vizualizare a datelor, care nu se modifică atunci când se schimbă suportul de memorare fizic, ceea ce asigură imunitatea structurii bazei de date și a aplicațiilor la modificări ale sistemului hardware utilizat.



Test de autoevaluare

2. Dați o definiție a bazelor de date?

1. 3. Clasificarea sistemelor de baze de date

Se pot lua în considerare mai multe criterii de clasificare ale sistemelor de baze de date.

1.3.1. Clasificare după modelul de date.

Majoritatea sistemelor de baze de date actuale sunt realizate în modelul de date relațional sau în modelul de date orientat obiect. Dezvoltarea continuă a acestor modele a condus către o nouă categorie de baze de date numite obiect-relaționale, care combină caracteristicile modelului relațional cu caracteristicile modelului orientat obiect.

Modelul de date relațional (Relational Model) se bazează pe noțiunea de relație din matematică, care corespunde unei entități de același tip și are o reprezentare ușor de înțeles și de manipulat, ce constă dintr-un tabel bidimensional, compus din linii și coloane. Fiecare linie din tabel reprezintă o entitate și este compusă din mulțimea valorilor atributelor entității respective, fiecare atribut corespunzând unei coloane a tabelului.

Modelul de date relațional a fost propus de cercetătorul E.F.Codd de la compania IBM, care a publicat în 1970 lucrarea "Un model relațional de date pentru bănci mari de date partajate". Alte lucrări ale lui Codd, ca și ale altor cercetători ca R. Boyce, J.D. Ullman etc au perfecționat modelul de date relațional și au permis dezvoltarea sistemelor de baze de date.

Chiar dacă noțiunile de relație și tabel diferă în esența lor, relația reprezentând o mulțime de entități și tabelul o reprezentare vizuală a acesteia, cele două denumiri se pot folosi, în general pentru același scop.

Pe baza acestor noțiuni, se poate sintetiza esența modelului relațional prin următoarele caracteristici:

- Datele sunt percepute de utilizatori ca tabele

Operatorii relaționali care pot fi folosiți pentru prelucrarea datelor generează un tabel rezultat din tabelele operanți

- Asocierea dintre tabele se realizează prin intermediul egalității valorilor unor atribute comune, ceea ce permite rezolvarea oricărei interogări.

Pe lângă avantajul unui model de date precis și simplu, sistemele de baze de date relaționale mai beneficiază și de un limbaj de programare recunoscut și acceptat, limbajul SQL (*Structured Query Language*), pentru care au fost emise mai multe standarde de către Organizația Internațională de Standardizare (*International Standardization Office-ISO*). Majoritatea sistemelor de gestiune a bazelor de date relaționale actuale implementează versiunea din anul 1992 a standardului pentru limbajul SQL, denumită SQL 92 sau SQL2.

Modelul de date orientat obiect (Object Model) este un concept unificator în știința calculatoarelor, fiind aplicabil în programare, în proiectarea hardware, a interfețelor, a bazelor de date etc. Sistemele de baze de date orientate obiect se bazează pe limbaje de programare orientate obiect cu capacități de persistență, în care datele sunt independente de timpul de viață al programelor care le creează sau accesează, prin memorare pe suport magnetic (disc).

Există și unele domenii, în special cele care manipulează tipuri de date complexe, cum ar fi proiectarea asistată de calculator, sisteme de informații geografice, medicină etc, în care modelul relațional s-a dovedit a fi insuficient de expresiv și cu performanțe de execuție reduse.

Caracteristicile importante ale modelului orientat obiect sunt: abstractizarea, moștenirea, încapsularea, modularizarea.

În programarea orientată obiect, programele sunt organizate ca și colecții de obiecte cooperante, fiecare obiect fiind o instanță a unei clase. Fiecare clasă reprezintă abstractizarea unui tip de entitate din realitatea modelată, iar clasele sunt membre ale unei ierarhii de clase, corelate între ele prin relații de moștenire. Orice obiect este încapsulat, ceea ce înseamnă că reprezentarea lui (adică structura internă a acelui obiect) nu este vizibilă utilizatorilor, care au acces doar la funcțiile (metodele) pe care acel obiect este capabil să le execute. Clasele și obiectele unui program orientat obiect sunt grupate în module, care pot fi compilate separat și între care există granițe bine definite și documentate, ceea ce reduce complexitatea de manevrare a datelor.

Din perspectiva realizării bazelor de date, o altă proprietate a modelului obiect, persistența, este aceea care asigură

memorarea transparentă pe suport magnetic a obiectelor care alcătuiesc o bază de date orientată obiect.

Modelul de date obiect-relațional (Object-Relational Model) reprezintă extinderea modelului relațional cu caracteristici ale modelului obiect, extindere necesară pentru realizarea bazelor de date care definesc și prelucrează tipuri de date complexe.

În esență, modelul obiect-relațional păstrează structurarea datelor în relații (reprezentate ca tabele), dar adaugă posibilitatea definirii unor noi tipuri de date, pentru domeniile de valori ale atributelor. Tipurile de date definite de utilizator pot fi extinse prin mecanismul de moștenire și pentru fiecare tip sau subtip se pot defini metode pe care le pot executa obiectele de acel tip.

De asemenea mai sunt încă în funcțiune baze de date modele mai vechi: modelul ierarhic și modelul rețea.

În modelul de date ierarhic (Hierarchical Model) o bază de date se reprezintă printr-o structură ierarhică de înregistrări de date (records) conectate prin legături (links). Modelul ierarhic a fost primul model folosit pentru dezvoltarea bazelor de date.

Schema conceptuală a unei baze de date în modelul ierarhic se reprezintă printr-un număr oarecare de scheme ierarhice. O schemă ierarhică este un arbore direcționat, reprezentat pe mai multe niveluri, în care nodurile sunt tipurile de înregistrări, iar arcele sunt tipurile de legături. Fiecare nod (cu excepția nodului rădăcină) are o singură legătură către un nod de pe un nivel superior (nodul părinte) și fiecare nod (cu excepția nodurilor frunză) are una sau mai multe legături către noduri de pe nivelul imediat inferior (noduri fii).

Modelul de date rețea (Network Model) folosește o structură de graf pentru definirea schemei conceptuale a bazei de date; nodurile grafului sunt tipuri de entități (înregistrări, records), iar muchiile grafului reprezintă în mod explicit asocierile (legăturile, links) dintre tipurile de entități.

La fel ca și modelul ierarhic, dezavantajul principal al modelului rețea este acela că fiecare interogare trebuie să fie prevăzută încă din faza de proiectare, prin memorarea explicită a legăturilor între tipurile de entități. În plus, complexitatea reprezentării datelor în modelul rețea este deosebit de ridicată, iar programatorii trebuie să o cunoscă pentru a putea realiza aplicațiile necesare.

1.3.2. Clasificare după numărul de utilizatori.

Majoritatea sistemelor de baze de date sunt sisteme *multiutilizator*, adică permit accesul concurent (în același timp) a mai multor utilizatori la aceeași bază de date. Există și un număr

reduc de sisteme *monoutilizator*, adică suportă accesul doar al unui utilizator (la un moment dat).

1.3.3. Clasificare după numărul de stații pe care este stocată baza de date

O altă clasificare este cea după numărul de stații pe care este stocată baza de date

Există două categorii de sisteme de baze de date: centralizate și distribuite.

Un sistem de baze de date *centralizat* (*Centralized Database System*) este un sistem de baze de date în care datele și sistemul de gestiune sunt stocate pe un singur calculator.

Un sistem de baze de date *distribuit* (*Distributed Database System*) poate avea atât datele, cât și sistemul de gestiune, distribuite pe mai multe calculatoare interconectate printr-o rețea de comunicație.



Test de autoevaluare

3. Faceți o clasificare a bazelor de date?

1.4. Securitatea și protecția datelor în bazele de date

Prin protecția și securitatea datelor se înțelege totalitatea mijloacelor, metodelor și a mecanismelor destinate prevenirii distrugerii, modificării sau folosirii neautorizate a informației protejate.

Referitor la protecția și securitatea datelor, în literatura de specialitate se definesc următoarele concepte de bază:

- **Securitatea datelor** – totalitatea măsurilor de protecție împotriva distrugerii accidentale sau intenționate, a modificării neautorizate sau a divulgării acestora
- **Caracterul secret** – este un concept ce se aplică la un individ sau organizație și constă în dreptul acestora de a decide ce informații se pot folosi în comun și în ce condiții
- **Confidențialitatea** – se aplică la date și se referă la statutul acordat, acesta reprezentând nivelul sau gradul de protecție ce trebuie acordat informației respective

- **Integritatea** – se referă la restricția ca sensul datelor să nu difere față de cel înscris pe documentul sursă, impunând totodată ca datele să nu fie alterate accidental sau voit.

Noțiunile de mai sus sunt strâns legate între ele, măsurile parțiale se suprapun și se acoperă reciproc.

Securitatea și protecția datelor din baza de date constituie un domeniu foarte vast, care prezintă două aspecte principale: pe de o parte, elementele legale și etice privind drepturile de acces la anumite informații, iar pe de altă parte, elementele legate de organizarea sistemelor informatice din punct de vedere al posibilităților de acces la datele stocate.

Unele informații care există în baza de date sunt strict private și nu pot fi accesate legal de către persoane neautorizate. Diferite reglementări guvernamentale sau legi existente în majoritatea țărilor stabilesc ce informații privind activitatea instituțiilor sau a persoanelor pot fi făcute publice și în ce condiții.

La nivelul sistemelor informatice se pot diferenția aspecte de securitate la nivel fizic (hardware), la nivelul sistemului de operare și la nivelul sistemului de gestiune al bazei de date.

În principal, de problemele de protecție și securitate este responsabil administratorul bazei de date, care are un cont privilegiat în sistemul de gestiune (numit în general cont de sistem - *system account*) care prevede capabilități foarte puternice, pe care alte conturi sau utilizatori nu le au. Prin intermediul contului de sistem administratorul bazei de date poate efectua mai multe operații: crearea conturilor, acordarea sau retragerea privilegiilor, etc.

Orice persoană care dorește să se conecteze (*log in*) la o bază de date trebuie să dețină un cont (*account, user*) și o parolă (*password*). Sistemul de gestiune verifică contul și parola și autentifică acel utilizator, dacă acestea sunt corecte. Programele de aplicații sunt considerate de asemenea utilizatori și se conectează pe un anumit cont și trebuie să furnizeze parola acestuia.

O altă tehnică de protecție și securitate a datelor este criptarea datelor (*Data Encryption*), prin care datele importante sunt codate folosind diferiți algoritmi de codare, mai ales atunci când sunt transmise prin intermediul rețelelor de comunicație. Interpretarea datelor criptate este dificilă dacă nu este cunoscută cheia (cifrul) de codare. În felul acesta numai utilizatorii autorizați care dețin cheile de decriptare pot interpreta cu ușurință aceste date.

O altă tehnică de securizare a bazei de date pentru aplicațiile web ar fi instalarea unui firewall, acesta fiind un calculator pe care este instalat un software special care permite accesarea calculatorului pe care este stocată baza de date numai de către anumite calculatoare.

Deci, prin securitatea bazei de date se înțelege o multitudine de măsuri destinate protecției informațiilor conținute în baza de date împotriva unor alterări, distrugeri sau divulgări neautorizate.



Test de autoevaluare

4. Specificați anumite tehnici de protecție a bazelor de date.



Lucrare de verificare a cunoștințelor

1. Spuneți câteva avantaje ale faptului că datele se păstrează și se manipulează mai bine stocate în format electronic.
2. Descrieți în câteva cuvinte de ce este important să protejăm datele.



Răspunsuri și comentarii la întrebările din testele de autoevaluare

Întrebarea 1.

Bazele de date sunt importante deoarece în cursul unei zile majoritatea persoanelor desfășoară activități care implică interacțiunea cu o bază de date: depunerea sau extragerea unei sume de bani din bancă, rezervarea biletelor de tren sau de avion, căutarea unei cărți într-o bibliotecă computerizată, gestiunea angajaților dintr-o firmă, cumpărarea unor produse etc.

Întrebarea 2.

O bază de date este o colecție centralizată de date în scopul optimizării prelucrării acestora în contextul unui set dat de aplicații. Operațiile care se pot realiza asupra datelor dintr-o bază de date sunt de inserare a unor date noi, de modificare a datelor existente, de ștergere a datelor sau de crearea a unor interogări pentru regăsirea unor informații după un anumit criteriu.

Întrebarea 3.

O scurtă clasificare a bazelor de date este următoarea:

- Clasificare după modelul de date
 - Model de date relațional
 - Model de date orientat obiect
 - Model de date obiect-relațional
 - Model de date ierarhic
 - Model de date rețea
- Clasificare după numărul de utilizatori
 - Baze de date multiutilizator
 - Baze de date monoutilizator
- Clasificare după numărul de stații pe care este stocată baza de date
 - Sisteme de baze de date centralizate
 - Sisteme de baze de date distribuite

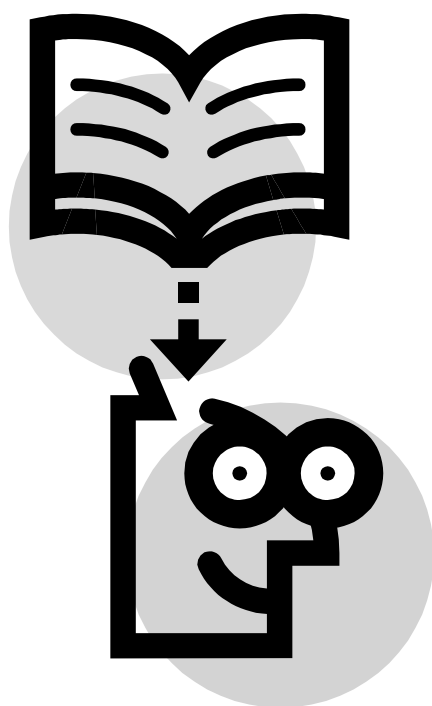
Întrebarea 4.

Securitatea și protecția datelor din baza de date prezintă două aspecte principale: pe de o parte, elementele legale și etice privind drepturile de acces la anumite informații, iar pe de altă parte, elementele legate de organizarea sistemelor informatice din punct de vedere al posibilităților de acces la datele stocate.

Două din tehnicile de securizare și protecție a datelor din baza de date sunt: crearea de conturi de utilizatori cu anumite drepturi pentru fiecare utilizator în parte de către administratorul bazei de date și criptarea datelor. Mai există și alte tehnici în funcție de tipul aplicației.

Bibliografie:

- Cârstoiu, Dorin, *Baze de date relaționale*, Editura Printech, 1999
- Rădulescu, Florin, *Baze de date în Internet*, Editura Printech, 2000
- Ionescu, Felicia, *Baze de date relaționale și aplicații*, Editura Tehnică, 2004
- Baltac, Vasile, *ECDL-Excel, Access, PowerPoint în 20 lecții și 75 de simulări*, Editura Andreco, 2003
- Browne, Allen, Balter Alison, *Bazele Access 95*, Editura Teora, 1999
- Pribeanu, Costin, *Baze de date și aplicații*, Editura MatrixRom, 2000
- Pascu, C., Pascu A., *Totul despre SQL*, Editura Tehnică, 1994



Unitatea de învățare Nr. 2

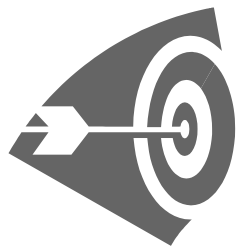
SISTEMUL DE BAZE DE DATE – CONCEPTE ȘI ARHITECTURĂ

Cuprins	Pagina
Obiectivele unității de învățare nr. 2	14
2.1. Componentele unui sistem de baze de date	14
2.1.1. Hardware	14
2.1.2. Software	15
2.1.3. Utilizatorii	16
2.1.4. Date persistente	16
2.2. Arhitectura internă a sistemelor de baze de date. Modele de date, scheme și instanțe	17
2.3. Independența datelor	19
2.4. Limbaje SGBD	20
2.5. Interfețe SGBD	21
Interfețe bazate pe meniuri	
Interfețe grafice	
Interfețe bazate pe forme	
Interfețe în limbaj natural	
Interfețe specializate aferente cererilor repetate	
Interfețe pentru administratorii bazelor de date	
2.6. Exemple de SGBD	22
Lucrare de verificare a cunoștințelor	24
Răspunsuri și comentarii la întrebările din testele de autoevaluare	25
Bibliografie	27

În unitatea de învățare nr. 1 am realizat o scurtă introducere în acest amplu domeniu al bazelor de date. Am adus în discuție unde se folosesc bazele de date, am prezentat o clasificare a sistemelor de baze de date și am dorit să subliniez importanța securizării bazelor de date și a protecției datelor stocate în bazele de date

În această unitate de învățare vom descrie mai în detaliu ce este un sistem de baze de date și în ce constă el.

OBIECTIVELE unității de învățare nr. 2



Principalele obiective ale unității de învățare nr. 2 sunt:

După studiul unității de învățare nr. 2 vei fi capabil să demonstrezi că ai dobândit cunoștințe suficiente pentru a înțelege:

- ce este un sistem de baze de date
- care sunt componentele unui sistem de baze de date
- care sunt tipurile de utilizatori de baze de date
- care este arhitectura internă a unui sistem de baze de date
- care este independența fizică și cea logică a datelor din baza de date
- care sunt limbajele pentru sistemele de gestiune a bazelor de date
- care sunt interfețele corespunzătoare fiecărui tip de utilizator
- câteva exemple de sisteme de baze de date

2.1. Componentele unui sistem de baze de date

Un sistem de baze de date (Database System) reprezintă un ansamblu de componente care asigură crearea, utilizarea și întreținerea uneia sau mai multor baze de date. Componentele unui sistem de baze de date sunt: hardware, software, utilizatori, date persistente.

2.1.1. Hardware.

Calculatoarele pe care sunt instalate de obicei sistemele de baze de date sunt PC standard, dar și calculatoare multiprocesor foarte puternice. Performanțele generale de operare ale calculatorului (numărul și viteza procesoarelor, dimensiunea și viteza de operare a memoriei etc) influențează în mod

corespunzător performanțele sistemului de baze de date. Cea mai importantă caracteristică a calculatorului pe care funcționează sistemul de baze de date este capacitatea harddisk-ului, utilizată pentru memorarea datelor din baza de date.

Deoarece într-un sistem de baze de date este necesar accesul rapid la oricare dintre înregistrările de date, pentru memorarea acestora se folosesc discurile magnetice (harddisk-uri). Benzile magnetice (care oferă acces secvențial la înregistrările de date) se folosesc pentru duplicarea (backup), salvarea și restaurarea datelor.

2.1.2. Software

Între baza de date (colecția de date memorate fizic în fișiere pe harddisk-uri) și utilizatorii sistemului există un nivel software, numit sistem de gestiune a bazei de date (SGBD)-(DataBase Management System). O bază de date computerizată poate fi generată și menținută fie cu ajutorul unui grup de programe de aplicație specifice acestui scop, fie cu acest SGBD.

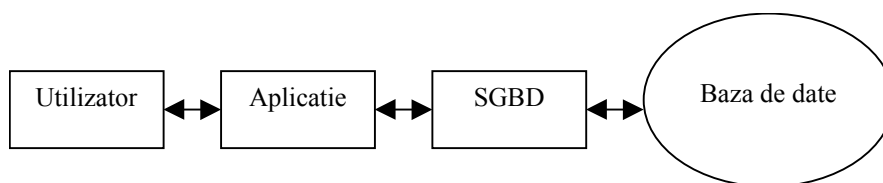


Figura 2.1. Componente ale sistemului de baze de date

Sistemul de gestiune al bazei de date (SGBD) este un interpretor de cereri, el recepționând de la utilizatori anumite cereri de acces la baza de date, le interpretează, execută operațiile respective și returnează rezultatul către utilizatori.

De fapt, SGBD este un sistem de programe general ce facilitează procesul definirii, construcției și manipulării datelor pentru diverse aplicații.

- *Definirea* bazei de date presupune specificarea tipurilor de date ce vor fi stocate în baza de date, precum și descrierea detaliată a fiecărui tip de dată.
- *Construcția* bazei de date reprezintă procesul stocării datelor însăși prin mediul controlat prin SGBD.
- Prin *manipulare* se înțeleg o serie de funcții ce facilitează implementarea cererilor pentru găsirea datelor specificate, adăugarea de noi date ce reflectă modificarea contextului, generarea de rapoarte pe baza conținutului bazei de date. În concluzie, pachetul software ce asigură manipularea

datelor, împreună cu datele însăși (conținutul bazei de date) formează ceea ce se numește sistemul de baze de date (DataBase System).

Un SGBD oferă utilizatorilor o viziune a datelor stocate în baza de date, nemaifiind necesară cunoașterea organizării particulare a sistemului, asigură o protecție a datelor față de accese neautorizate și de anumite defecte de funcționare.

2.1.3. Utilizatorii

Utilizatorii unui sistem de baze de date se împart în câteva categorii:

- *Programatorii de aplicații* sunt cei care dezvoltă aplicațiile de baze de date în anumite medii de programare. Aplicațiile pot fi aplicații desktop (stand alone) și aplicații client-server. Aplicațiile desktop sunt aplicațiile care se instalează și rulează pe un anumit calculator. Acestea sunt aplicații implementate în medii de programare cum ar fi: Visual Basic, Visual C, Java, C++, Delphi etc, iar aplicațiile client-server sunt aplicații care se instalează pe un calculator numit server și rulează de pe orice calculator aflat în aceea rețea. Aceste aplicații sunt aplicații web implementate în limbajul de scripturi php sau asp, cu interfața dezvoltată în html. Pentru ca aceste aplicații să funcționeze trebuie instalat și un server de web, cum ar fi Apache sau IIS și pe fiecare calculator de unde va fi accesată aplicația, precum și un browser de web: Internet Explorer, Netscape, Mozilla etc.
- *Utilizatorii obișnuiți* sunt acei utilizatori care accesează baza de date prin intermediul unei aplicații de baze de date. Acești utilizatori au drepturi limitate asupra accesului la datele din baza de date, ei neavând cunoștințe aprofundate asupra structurii și a datelor din acea bază de date.
- *Administratorul bazei de date* (DataBase Administrator) care este o persoană autorizată, care are ca sarcină administrarea resurselor, autorizarea accesului la baza de date, a coordonării și monitorizării utilizatorilor acelei baze de date. Administratorul bazei de date efectuează și operații periodice de salvare a datelor (backup) și de refacere a lor atunci când este necesar.

2.1.4. Date persistente

Datele memorate într-o bază de date sunt *date persistente*, adică date care rămân memorate pe suport magnetic, independent de execuția programelor de aplicații. Datele persistente ale unei baze de date se introduc, se șterg sau se actualizează în funcție de date de intrare provenite de la

tastatură. Inițial datele de intrare sunt date nepersistente, ele devenind persistente după ce au fost validate de SGBD. Datele de ieșire ale unui sistem de baze de date sunt tot date nepersistente, ele provenind din operații de interogare a bazei de date și puse la dispoziție utilizatorului sunt formă de raport, afișare etc.



Test de autoevaluare

3. 1. Enumerați pe scurt componentele unui sistem de baze de date .

2.2. Arhitectura internă a sistemelor de baze de date

Arhitectura internă a unui sistem de baze de date propusă prin standardul ANSI/X3/SPARC (1975) conține trei niveluri funcționale ce vor fi descrise mai jos.

Una din caracteristicile fundamentale a bazelor de date este dată de faptul că produce câteva niveluri de abstractizare a datelor, prin ascunderea detaliilor legate de stocarea datelor, detalii ce nu sunt utile utilizatorilor bazei de date. Se definește modelul datelor ca un set de concepte ce poate fi utilizat în descrierea structurii datelor. Prin structura bazei de date se înțelege tipul datelor, legătura dintre ele, restricțiile ce trebuie îndeplinite de date. Cele mai multe baze de date includ un set de operații ce specifică modul de acces la date.

O structură de date asociată unei baze de date poate fi reprezentată pe trei niveluri, având ca scop separarea aplicațiilor utilizatorului de baza de date fizică. Schema bazei de date pe cele trei niveluri poate fi văzută astfel:

- *Nivelul intern* constituit din schema internă ce descrie structura de stocare fizică a datelor în baza de date, utilizând un model al datelor fizice. La acest nivel se descriu detaliile complete ale stocării, precum și modul de acces la date.
- *Nivelul conceptual* sau schema conceptuală descrie structura întregii baze de date pentru o comunitate de utilizatori. La nivelul conceptual se face o descriere completă a bazei de date, ascunzând detaliile legate de stocarea fizică, concentrându-se asupra descrierii entităților, tipurilor de date, relațiilor dintre ele, precum și a restricțiilor asociate. Poate fi utilizat cu bune rezultate, la

model de nivel înalt sau un model specific de implementare.

- *Nivelul extern* sau nivelul vizual (utilizator) include o colecție de scheme externe ce descriu baza de date prin prisma diferiților utilizatori. Fiecare grup de utilizatori descrie baza de date prin prisma propriilor interese. Există tendința la acest nivel ca grupuri de utilizatori să ascundă detalii de care nu sunt interesate. Și la acest nivel se pot folosi modele de implementare sau modele de nivel înalt.

Desigur că în multe SGBD nu se poate face o distincție netă între cele trei nivele. Cu toate acestea se poate remarca la majoritatea SGBD un nivel conceptual puternic ce suplinește aparent de cele mai multe ori celelalte niveluri. De asemenea, se remarcă o contopire mai ales la dezvoltarea aplicațiilor a nivelului conceptual și extern. Se remarcă faptul că arhitectura pe trei niveluri reprezintă numai o descriere a datelor la nivel fizic. Grupurile de utilizatori se referă numai la schema externă, deci SGBD este cel ce va trebui să transforme schema externă în schemă conceptuală.

De la modelul conceptual cererile sunt adresate modelului intern pentru a fi procesate și aplicate datelor stocate. Procesul transferului cererilor și rezultatelor între nivele este numit *cartografiere* (mapping). Acest proces este mare consumator de timp pentru care multe SGBD nu posedă nivel extern.

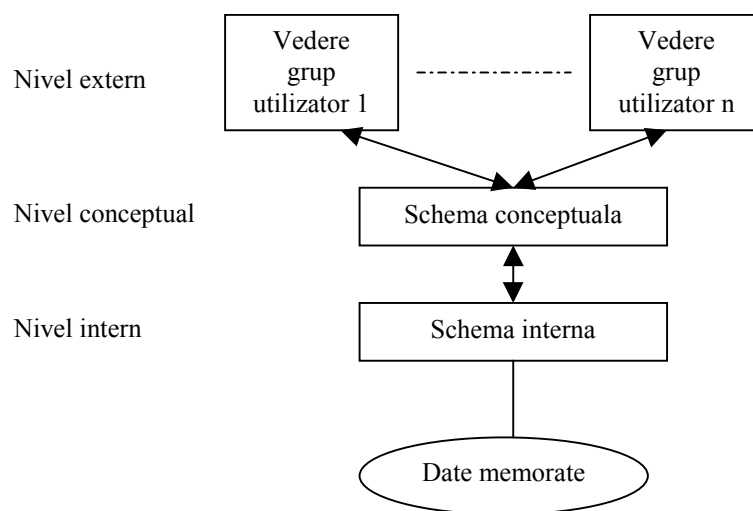


Figura 2.2. Arhitectura internă a unui sistem de baze de date



Test de autoevaluare

2. Ce reprezintă structura unei baze de date?
3. De ce este importantă împărțirea unui sistem de baze de date pe 3 niveluri?

2.3. Independența datelor

Această organizare pe trei niveluri a sistemelor de baze de date este importantă pentru că explică conceptul de independență a datelor, prin posibilitatea modificării sistemului bazei de date la orice nivel fără a influența nivelurile superioare. Independența datelor se poate defini în două moduri, aferente nivelurilor conceptual și intern.

Prin *independența logică* se înțelege capacitatea schimbării schemei conceptuale fără a atrage după sine schimbări în schema externă sau în programele de aplicație. Este posibilă schimbarea schemei conceptuale prin expandarea bazei de date ca urmare a adăugării de noi tipuri de înregistrări sau a datelor însăși, sau prin reducerea bazei de date ca urmare a reducerii înregistrărilor. Schema conceptuală după aceste operații se referă la schema conceptuală a datelor existente. Un exemplu de expandare al bazei de date este cel de adăugare a unei noi coloane la un tabel.

Independența fizică este reprezentată prin capacitatea de schimbare a schemei interne fără schimbarea schemei conceptuale sau externe. Schimbarea schemei conceptuale poate surveni ca urmare a reorganizării fizice a unor fișiere, prin crearea de noi structuri de acces menite să asigure accesul eficient la date. Dacă sistemul conține SGBD pe mai multe niveluri, catalogul trebuie să reflecte modul în care diverse cereri se implementează la fiecare nivel. Motivele prezentate mai sus pledează pentru utilizarea arhitecturii pe trei nivele.



Test de autoevaluare

4. Care este diferența dintre independența fizică și independența logică a datelor din baza de date?

2.4. Limbaje SGBD

SGBD trebuie să ofere limbajele corespunzătoare tuturor categoriilor de utilizatori. După proiectarea bazei de date și alegerea SGBD, este foarte importantă construirea schemei interne și conceptuale a bazei de date. Cum în cele mai multe situații, nu există o separație netă între cele două nivele, un limbaj numit *Data Definition Language (DDL)* este utilizat de administratorul bazei de date și de proiectantul bazei de date în definirea ambelor scheme. Un compilator DDL procesează instrucțiunile pentru identificarea descrierilor despre construcție și memorează aceasta în catalogul SGBD.

În SGBD cu o clară separație între nivelul conceptual și cel intern, DDL este utilizat pentru specificarea schemei conceptuale. Un alt limbaj numit *Storage Definition Language (SDL)* este utilizat pentru specificarea schemei interne. Legătura între cele două nivele de implementare este asigurată de unul din cele două. În general, fără specificare explicită, referirea la definirea bazei de date presupune utilizarea DDL.

Pentru o arhitectură pe trei nivele, este necesar un al treilea nivel numit *View Definition Language (VDL)* destinat utilizatorilor și legăturii acestora cu nivelul conceptual. Multe sisteme de baze de date realizează aceasta printr-un DDL ce acceptă și declarații specifice nivelului extern.

Instrucțiunile în limbaj DDL pot fi înglobate într-un limbaj general de programare sau pot fi compilate separat. Odată schema compilată și baza de date populată cu date, utilizatorul are o serie de facilități pentru manipularea datelor. Operațiile tipice includ căutarea, inserarea, ștergerea și modificare datelor. Pentru aceasta SGBD dispune de *Data Manipulation Language (DML)*. La rândul său, DML poate fi împărțit în două componente:

- un DML de nivel înalt
- un DML de nivel scăzut

Secțiunea DML de nivel înalt sau *neprocedurală* este utilizată pentru specificarea operațiilor complexe în baza de date într-o formă concisă. În general, operațiile utilizând această secțiune sunt realizate fie într-o formă interactivă de la terminal, fie prin utilizarea unui limbaj de programare universal.

Secțiunea DML de nivel scăzut sau *procedurală* este realizată utilizând un limbaj de programare general. Cu această secțiune se realizează operațiile tipice, cum sunt refacerea unei înregistrări individuale, procesarea separată a înregistrărilor bazei de date. Din motivul că operează asupra înregistrărilor individuale, această prelucrare se mai numește și înregistrare cu înregistrare. O comandă pentru DML de nivel înalt specifică o cerere de acces la date, dar nu specifică modul în care se realizează aceasta. Din

acest motiv acest limbaj se numește *declarativ*. Oricum, comenzile DML fie de nivel înalt, fie de nivel scăzut sunt implementate într-un limbaj de programare general, limbaj numit și limbaj gazdă, iar DML este numit limbaj de date. DML de nivel înalt utilizat într-o manieră interactivă formează ceea ce se numește *query language*.



Test de autoevaluare

5. Dați exemple de limbaje de sisteme de gestiune a bazelor de date.

2.5. Interfețe SGBD

SGBD trebuie să ofere interfețe corespunzătoare tuturor categoriilor de utilizatori. Aceste interfețe au ca scop facilitarea legăturii între utilizatori și sistemul de baze de date. Principalele tipuri de interfețe oferite de SGBD sunt:

- *Interfețe bazate pe meniuri.* Acestea oferă utilizatorului o listă de opțiuni, numite meniuri care îi ajută la formularea cererilor. Nu este necesară memorarea unor comenzi deoarece o comandă specifică este formată pas cu pas prin compunerea opțiunilor indicate prin meniu.
- *Interfețe grafice.* Aceste interfețe afișează utilizatorului o diagramă. Utilizatorul poate formula cererea prin manipularea acestei diagrame. În cele mai multe cazuri, interfețele grafice sunt combinate cu meniuri.
- *Interfețe bazate pe forme.* Aceste interfețe sunt acelea prin intermediul cărora utilizatorul poate completa formele cu noile date pe care le dorește să le insereze, sau folosește aceste forme pentru a cere SGBD să obțină datele de interes.
- *Interfețe în limbaj natural.* Aceste interfețe acceptă cereri scrise în limba engleză sau alte limbi de circulație internațională. O interfață în limbaj natural conține uzual o schemă proprie similară cu schema conceptuală a bazelor de date. Interpretarea cererilor se face pe baza unui set standard de cuvinte cheie ce sunt interpretate pe baza schemei interne. Dacă interpretarea se realizează cu succes, programul de interfață generează cererea de nivel înalt corespunzătoare celei în limbaj natural, ce va fi transmisă către SGBD.

- *Interfețe specializate aferente cererilor repetate.* Aceste interfețe sunt destinate unei anumite categorii de utilizatori, de exemplu utilizatorii care se ocupă de operațiile dintr-o bancă. Uzual, un mic set de comenzi prescurtate sunt implementate pentru a scurta timpul necesar introducerii comenzii, sau chiar utilizarea de chei funcționale. Aceste interfețe implementează un limbaj numit și limbaj de comandă.
- *Interfețe pentru administratorii bazelor de date.* Acestea sunt utilizate în implementarea comenzilor privilegiate ce sunt folosite de administratorii bazelor de date. Astfel de comenzi includ crearea de conturi, setarea parametrilor sistemului, autorizarea intrării într-un anumit cont, reorganizarea structurii de stocare a datelor din baza de date, precum și o serie de facilități legate de administrarea bazei de date, cum sunt: accesul la tabele și înregistrări, facilități de acces la câmpuri ale tabelelor de date.



Test de autoevaluare

6. Ce sunt interfețele sistemelor de gestiune a bazelor de date?

2.6. Exemple de SGBD

În momentul de față, pe piață există o ofertă foarte mare de sisteme de gestiune a bazelor de date, de la sisteme care se pot folosi gratuit (fără licență sau cu licență publică), până la sisteme de înaltă performanță, a căror utilizare necesită cumpărarea de licențe. Pentru aceste sisteme există pe site-urile producătorilor versiuni de test numite trial version, pentru care nu se plătește licență, durata folosirii respectivului produs fiind limitată la un număr de zile (30, 60 zile, în funcție de producător).

Microsoft SQL Server este sistemul de gestiune a bazelor de date relaționale multi-utilizator dezvoltat de firma Microsoft pentru sistemele de operare Windows. Au existat mai multe versiuni, cea actuală fiind SQLServer 2000 (SQL Sever 2003 fiind încă în faza de testare). În toate versiunile, acest sistem de baze de date suportă standardul SQL2, cu implementarea performantă a trăsăturilor avansate de stocare și prelucrare a datelor. Există o interfață grafică pentru interacțiunea cu utilizatorul, pentru folosirea tuturor opțiunilor: de export/ import date, de creare și manipulare a tabelor, pentru popularea cu date a tabelor, de

creare a interogărilor, a procedurilor stocate, a triggerelor etc. Pentru a obține gratuit o versiune de test, accesați adresa <http://www.microsoft.com>.

Microsoft Access este unul din cele mai cunoscute sisteme de gestiune a bazelor de date relaționale pe platforme de calculatoare personale. Microsoft Access dispune de un sistem de control al bazei de date (database engine) și o interfață grafică pentru interacțiunea cu utilizatorul. Aplicațiile de baze de date în MS Access se pot dezvolta cu multă ușurință datorită generatoarelor de aplicații (wizards) care permit proiectarea vizuală a bazelor de date, a formularelor (forms) pentru interfețele grafice și a rapoartelor (reports). MS Access este folosit în special pentru aplicații personale sau pentru mici afaceri și licența acestuia se cumpără odată cu cumpărarea licenței produsului Microsoft Office. Acest sistem este cel folosit de noi în capitolele următoare.

Sistemul Oracle este un sistem de gestiune al bazelor de date multi-utilizator foarte puternic, cu implementări pe toate platformele (Windows, Linux, Unix), care oferă atât performanțe de execuție ridicate, cât și un grad mare de protecție și securitate a datelor. În toate versiunile, Oracle oferă implementarea completă a caracteristicilor modelului relațional, conform standardului SQL2, iar ultimele versiuni (Oracle8i, Oracle9i etc) sunt sisteme de gestiune obiect-relaționale distribuite, implementând extensiile orientate obiect prevăzute în standardul SQL3 și oferind posibilitatea de dezvoltare a bazelor de date distribuite. De la adresa <http://www.oracle.com> se poate obține o versiune a sistemului de gestiune Oracle, dar și a diferitelor instrumente de dezvoltare a aplicațiilor de baze de date. Termenii licenței permit utilizarea gratuită a acestor sisteme în scopuri necomerciale pe o perioadă nelimitată, pentru utilizarea în scopuri comerciale trebuie plătite licențele corespunzătoare.

MySQL este un sistem de gestiune a bazelor de date relaționale cu implementări pentru sistemele de operare Linux, Unix, Windows. Acest sistem se poate utiliza gratuit, fiind open source. Ultima versiune și documentația sistemului de gestiune a bazelor de date MySQL se poate descărca de la adresa <http://www.mysql.com>. Acest sistem este compatibil cu standardul SQL2, dar unele prevederi ale standardului fiind implementate parțial.

Visual FOX PRO este un limbaj de programare complet, care acceptă un mediu interactiv și un mediu compilat la rulare. *Visual FOX PRO* este compatibil cu toate versiunile anterioare de FoxPro. Stilul de proiectare a interfeței FoxPro a fost întotdeauna orientat către flexibilitate și ușurință în utilizare. Pe de altă parte, forța și viteza brută au reprezentat dintotdeauna punctul forte al lui FoxPro. Nici un produs creat de celelalte companii axate pe baze de date, care au făcut trecerea la modelul orientat obiect nu a

putut rivaliza cu FoxPro în ce privește viteza de execuție a funcțiilor specifice bazelor de date . Și acest mediu conține vrăjitori (Wizard) pentru gestionarea mai multor taskuri.

IBM DB2 este un sistem de gestiune al bazelor de date al firmei IBM. Acest sistem asigură integritatea datelor, oferă o securitate sporită pentru date, are o interfață grafică pentru gestionarea bazei de date. Dispune, ca și Microsoft SQL Sever și Oracle de posibilitatea creării de proceduri stocate, acestea fiind niște proceduri care rulează pe server și asigură o viteză mai mare de răspuns. Are mai multe versiuni în funcție de dorințele și necesitățile utilizatorilor. Mai multe informații despre DB2 le găsiți dacă accesați www.db2mag.com, și pentru a vă comanda un CD sau pentru a vă descărca versiunea de evaluare a acestui sistem accesați www.ibm.com.



Test de autoevaluare

7. Dați exemple de sisteme de gestiune a bazelor de date.



Lucrare de verificare a cunoștințelor

Accesați paginile de web ale fiecărui sistem de gestiune a bazelor de date și precizați câteva avantaje și câteva dezavantaje ale fiecăruia.

Răspunsuri și comentarii la întrebările din testele de autoevaluare

Întrebarea 1.

Componentele unui sistem de baze de date sunt: componenta hardware, componenta software, utilizatorii și datele persistente care sunt stocate în baza de date.

Întrebarea 2.

Prin structura bazei de date se înțelege alegerea tipului de date, legăturile dintre ele, restricțiile ce trebuie îndeplinite de date.

Întrebarea 3.



Organizare pe trei niveluri a unui sistem de baze de date este importantă pentru că explică conceptul de independență a datelor, prin posibilitatea modificării sistemului bazei de date la orice nivel fără a influența nivelurile superioare. Independența datelor se poate defini în două moduri, moduri ce sunt aferente nivelurilor conceptual și intern.

Întrebarea 4.

Prin independența logică se înțelege capacitatea schimbării schemei conceptuale fără a atrage după sine schimbări în schema externă sau în programele de aplicație. Independența fizică este reprezentată prin capacitatea de schimbare a schemei interne fără schimbarea schemei conceptuale sau externe.

Întrebarea 5.

SGBD trebuie să ofere limbajele corespunzătoare tuturor categoriilor de utilizatori. Astfel:

- un limbaj numit *Data Definition Language (DDL)* este utilizat de administratorul bazei de date și de proiectantul bazei de date în definirea schemelor interne și conceptuale a bazei de date.
- un alt limbaj numit *Storage Definition Language (SDL)* este utilizat pentru specificarea schemei interne.
- un alt nivel numit *View Definition Language (VDL)* este destinat utilizatorilor și legăturii acestora cu nivelul conceptual

Întrebarea 6.

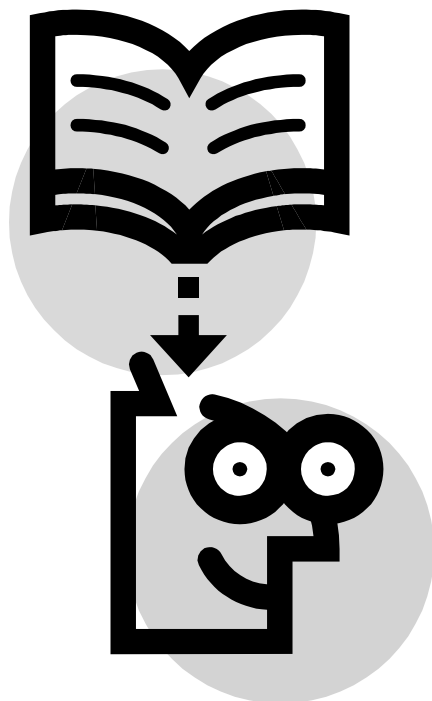
Interfețele au ca scop facilitarea legăturii între utilizatori și sistemul de baze de date.

Întrebarea 7.

Câteva dintre cele mai importante sisteme de gestiune a bazelor de date sunt: Microsoft SQL Server, Oracle, MySQL, IBM DB2, Microsoft Access, Visual FOX PRO etc.

Bibliografie:

- Cârstoiu, Dorin, *Baze de date relaționale*, Editura Printech, 1999
- Rădulescu, Florin, *Baze de date în Internet*, Editura Printech, 2000
- Ionescu, Felicia, *Baze de date relaționale și aplicații*, Editura Tehnică, 2004
- Baltac, Vasile, *ECDL-Excel, Access, PowerPoint în 20 lecții și 75 de simulări*, Editura Andreco, 2003
- Browne, Allen, Balter Alison, *Bazele Access 95*, Editura Teora, 1999
- Pribeanu, Costin, *Baze de date și aplicații*, Editura MatrixRom, 2000
- Pascu, C., Pascu A., *Totul despre SQL*, Editura Tehnică, 1994



Unitatea de învățare Nr. 3

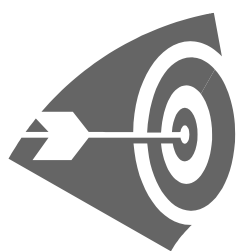
PROIECTAREA BAZELOR DE DATE

Cuprins	Pagina
Obiectivele unității de învățare nr. 3	28
3.1. Ce este proiectarea?	29
3.2. Modelul entitate-relație. Obiectele bazelor de date relaționale	32
Tabelă (relație)	
Câmp (atribut)	
Înregistrare (nuplu)	
3.3. Construcția schemelor relație	34
3.3.1. Relația unul-la-unul (1-1 sau one to one)	34
3.3.2. Relația unul-la-multe (1-N sau one to many)	35
3.3.3. Relația multe-la-multe (M-N sau many to many)	36
3.3.4. Relația unară	37
3.4. Diagrama entitate-relație	38
3.5. Constrângeri de integritate	39
3.5.1. Constrângerile de domeniu	40
3.5.2. Constrângerile referitoare la n-upluri (înregistrările din tabelă)-Cheia primară	41
3.5.3. Constrângeri între relații	41
3.6. Dependențe funcționale	44
3.7. Normalizare. Forme normale.	44
3.7.1. Forma normală de ordin 1 (FN1)	45
3.7.2. Forma normală de ordin 2 (FN2)	46
3.7.3. Forma normală de ordin 3 (FN3)	46
3.7.4. Forma normală Boyce-Codd (FNBC)	47
3.8. Structuri de indecși în tabelele de date	48
3.8.1. Indexul primar	50
3.8.2. Indexul secundar	51
3.8.3. Indexul de grup	52
3.8.4. Indexul multinivel	53
Lucrări de verificare a cunoștințelor	55
Răspunsuri și comentarii la întrebările din testele de autoevaluare	56
Bibliografie	57

Încă de la început doresc să vă felicit pentru parcurgerea cu succes a primelor două unități de învățare și să vă urez bun venit la studiul acestei noi unități de învățare. În primele două unități de învățare am realizat o scurtă introducere în acest amplu domeniu al bazelor de date și am descris mai în detaliu ce este un sistem de baze de date și în ce constă el.

Această unitate de învățare nr. 3 este foarte importantă pentru că dacă se dorește crearea unei aplicații de baze de date este necesar ca un prim pas proiectarea corectă a bazei de date.

OBIECTIVELE unității de învățare nr. 3



Principalele obiective ale unității de învățare nr. 3 sunt:

După studiul unității de învățare nr. 3 vei fi capabil să demonstrezi că ai dobândit cunoștințe suficiente pentru a înțelege:

- de ce este foarte importantă o proiectare corectă a bazei de date
- care sunt etapele care trebuie parcurse pentru o proiectare cât mai corectă a bazei de date
- ce reprezintă modelul entitate-relație
- care sunt obiectele din baza de date
- cum se construiesc schemele relație
- să descrii toate categoriile de relații care pot apărea într-o proiectare de bază de date
- să construiești o diagrama entitate-relație
- să înțelegi de ce trebuie să existe constrângeri asupra tabelor
- câte tipuri de constrângeri există și rolul lor
- ce este dependența funcțională
- ce reprezintă procesul de normalizare a tabelor
- tipurile și ce reprezintă fiecare formă normală
- la ce sunt folosiți indecșii în tabelele de date
- tipurile de indecși și o scurtă descriere a lor

3.1. Ce este proiectarea?

Proiectarea unei baze de date constă din proiectarea logică și fizice a acesteia, pentru a corespunde cerințelor utilizatorilor pentru un anumit set de aplicații.

În general, vom considera că proiectarea corectă a unei baze de date trebuie să parcurgă următoarele etape:

- Analiza cererilor și strângerea de informații referitoare la aplicație.
- Proiectarea conceptuală a bazei de date.
- Alegerea unui sistem de gestiune al bazelor de date.
- Proiectarea logică a bazei de date.
- Proiectarea fizică a bazei de date.
- Implementarea bazei de date și a aplicației.



Înainte de a se proiecta efectiv o bază de date, este necesar să se cunoască ce rezultate se așteaptă potențialii utilizatori să obțină de la baza de date respectivă și documentarea asupra informațiilor ce sunt disponibile pentru aceasta. De asemenea, este necesară și definirea cât mai exactă a aplicației (De ex.: aplicație de gestiune a stocurilor, a cărților într-o bibliotecă, aplicație contabilă, aplicație folosită la salarizare, aplicație referitoare la gestiunea studenților, a profesorilor, a cursurilor, a notelor dintr-o instituție de învățământ etc.).

Având cerințele formulate precis și concis se poate trece la elaborarea schemei conceptuale utilizând un model de nivel înalt. Schema conceptuală reprezintă o descriere concisă a datelor utilizatorului, incluzând descrierea detaliată a tipurilor de date, a relațiilor și restricțiilor acestora. Deoarece până la acest moment nu se includ detalii de implementare, rezultatele pot fi comunicate utilizatorilor, chiar dacă sunt nespecializați în domeniu, și analizate de aceștia pentru eliminarea eventualelor conflicte care pot apărea.

Următoarea etapă este cea de construcție a bazei de date ținând cont de detaliile actuale, cu ajutorul unui sistem de gestiune a bazelor de date (SGBD). Această etapă înglobează etapele de alegere a unui SGBD, de proiectarea logică a bazei de date și cea de proiectarea fizică a bazei de date.

Alegerea SGBD se face în funcție de complexitatea aplicației, de capacitatea de stocare a datelor, de posibilitatea

refacerii datelor, de numărul de utilizatori care vor folosi aplicația, dar și de costurile de achiziție ale sistemului, de costurile de întreținere etc.

Următoarea fază, cea de proiectare logică poate fi realizată în două sub-faze: transpunerea schemei conceptuale în modelul de date al sistemului SGBD ales, dar independent de sistemul de gestiune propriu-zis, sau rafinarea schemei conceptuale și a schemelor externe obținute anterior, astfel încât să se utilizeze mai multe din facilitățile oferite de sistemul SGBD ales (modul de generare a cheilor primare, definirea constrângerilor, etc.). Aceste două sub-faze se pot realiza împreună, folosind unul din instrumentele de proiectare oferite de sistemul SGBD ales. Rezultatul acestei faze de proiectare îl constituie, așadar, schema conceptuală și schemele externe ale bazei de date, dependente de sistemul SGBD ales și de modelul de date al acestuia.

Proiectarea fizică a bazei de date reprezintă procesul de alegere a structurilor de memorare și de acces la fișierele bazei de date, pentru a obține performanțe cât mai bune pentru aplicația proiectată. Ca parametri generali de alegere a opțiunilor proiectului fizic al unei baze de date relaționale se pot enumera: timpul de răspuns, utilizarea spațiului de memorare, capacitatea tranzacțională.

Deciziile de proiectare fizică se pot lua numai după o analiză a aplicațiilor care se vor executa și în principal, a interogărilor și tranzacțiilor pe care acestea le vor lansa. În urma analizei se pot sintetiza informații care să dea imaginea de ansamblu a utilizării atributelor relațiilor bazei de date: care attribute sunt actualizate cel mai frecvent, care attribute sunt folosite cel mai frecvent în selecții ale interogărilor, etc. Aceste informații se folosesc pentru stabilirea indecșilor secundari ai relațiilor.

Ultima etapă este cea de implementare efectivă a bazei de date și a aplicației. Aici se crează pe baza modelului definit obiectele bazei de date, se populează cu date baza de date, se verifică constrângerile, se crează interfețele cu utilizatorul și rapoartele necesare cu datele extrase din baza de date.

Exemplu:



Pe parcursul manualului vom proiecta și realiza o aplicație simplificată pentru gestiunea informațiilor referitoare la notele obținute de studenții dintr-o universitate la anumite materii. Baza de date se va numi Universitate. Pe parcursul acestui capitol aceasta va fi rafinată la modelul entitate-relație, un model conceptual de nivel înalt. Acest model asigură perceperea de către utilizatori fără să prezinte detaliile de stocare a datelor păstrate în calculator. În faza preliminară, după analiza cerințelor se cunosc următoarele informații :

- această universitate are în componență mai multe facultăți. Fiecare facultate având asociat un cod, o denumire, o adresă.
- studenții au stocate în baza de date informațiile personale ale fiecăruia (cnp, nume, prenume, inițiala tatălui, data nașterii etc), dar și informații legate de starea actuală a lor (grupa în care se află, facultatea de care aparține etc.)
- în această bază de date stocăm și materiile studiate în facultățile din acea universitate. Se consideră materii diferite acele materii care au aceeași denumire, dar profesor diferit.
- vom stoca și notele obținute de fiecare student la materia la care a fost evaluat prin examen.



Test de autoevaluare

1. Care sunt etapele ce trebuie parcurse pentru o proiectare corectă a bazei de date?

3.2. Modelul entitate-relație. Obiectele bazelor de date relaționale

Modelul entitate-relație este cel mai utilizat model conceptual de nivel înalt, care reprezintă schema conceptuală a bazei de date cu ajutorul entităților și a relațiilor dintre acestea. Acest model a fost introdus în anul 1976 de P.S.Chen. Elementele de bază folosite în cadrul acestui model sunt conceptele de entitate și cel de relație.

O *entitate* este un obiect al lumii reale, cu o existență independentă și poate reprezenta un obiect fizic, o activitate, un concept. O entitate este un obiect cu existență fizică, de exemplu: persoană particulară, automobil, companie, activitate, curs universitar.

Orice entitate are o serie de proprietăți numite *atribute*, ce descriu entitatea respectivă.

Cu toate că nu reprezintă același lucru, pentru denumirea de entitate se mai folosește și denumirea de tabel al bazei de date, iar pentru atribute câmpurile tabelului.

Scurte definiții a noțiunilor pe care le-am folosit și le vom folosi în continuare sunt :

- Tabelă (entitate) este o colecție de informații logice relaționale tratată ca o unitate
- Înregistrare (n-uplu). O tabelă este compusă din înregistrări sau rânduri. Fiecare înregistrare este tratată ca o simplă unitate. Fiecare înregistrare este legată de înregistrări ale altei tabele.
- Câmpuri (atribute). Înregistrările sunt constituite din câmpuri (coloane) . Un câmp este o particulă atomică a bazei de date ce reprezintă cea mai mică cantitate de informație care poate fi manipulată. Toate înregistrările dintr-o tabelă au aceleași câmpuri.

De exemplu, pentru baza de date Universitate pe care dorim să o creăm, o entitate reprezintă o Facultate cu atributele Cod Facultate, Denumire, Adresa, Nume Decan. O altă entitate ar fi Materii, descrise de atributele Cod Materie, Denumire, An, Nume Profesor.

Atributelor li se asociază valori care au ca scop identificarea entității. Această atribuire de valori pentru fiecare atribut formează o **înregistrare** a tabelului respectiv.



Unele attribute pot fi divizate în mai multe părți cu semnificație independentă. Un astfel de atribut este un **atribut complex**.

Un exemplu este cel al atributului Adresă care poate fi divizat în mai multe attribute : Oras, Cod Postal, Stradă, Numar, Bloc etc sau al atributului Nume Decan care poate fi divizat în Nume și Prenume.

Atributele care nu sunt compuse se numesc **attribute atomice**. Valoarea atributelor complexe se formează prin concatenarea valorilor atributelor atomice.

Multe attribute au valoare unică pentru o entitate particulară și sunt numite **attribute cu o singură valoare**.

De exemplu CNP-ul unei persoane. Există attribute ce pot lua mai multe valori dintr-un set dat, cum ar fi gradele didactice ale profesorilor universitari, culorile etc. Aceste attribute sunt attribute cu mai multe valori.

Atributele derivate sunt attributele ce se pot determina din alte attribute, cum ar fi vârsta unei persoane se poate calcula din data curentă minus data nașterii persoanei respective.

În anumite situații, o entitate poate să nu aibă valori pentru toate attributele asociate ei, în acest caz folosindu-se o valoare specială numită **atributul null**.

Un exemplu ar fi lipsa din atributul Adresă a numelui blocului sau a scării, etc.

Exemplu:

Revenind la exemplu nostru, să ne definim toate entitățile din baza de date Universitate.

Entitățile ar putea fi următoarele :

- Facultate cu attributele: CodFac, Denumire, Adresa, Nume Decan
- Studenti cu attributele: CodStud, An, Grupa, Media, Bursa
- StudPersonal cu attributele: CNP, Nume, Iniț, Prenume, Data Nașterii, Loc Nașterii, Tata, Mama, Adresa
- Materii cu attributele: Cod Materie, Denumire, An, Nume Profesor
- Note cu attributele Nota, Data





Test de autoevaluare

2. Care sunt obiectele bazei de date? Dați scurte definiții pentru fiecare obiect și specificați ce termeni se mai folosesc pentru fiecare dintre ele.

3.3. Construcția schemelor relație

În proiectarea bazelor de date se definesc relații sau asocieri între mulțimile de entități componente, pentru a reprezenta anumite aspecte ale realității pe care o modelează baza de date.

O *relație* este o corespondență între entități din una sau mai multe mulțimi de entități. Gradul unei relații este dat de numărul de mulțimi de entități asociate. Relațiile pot fi binare (între 2 mulțimi de entități) sau multiple (între mai mult de 2 entități).

Relațiile binare sunt împărțite în trei categorii, după numărul elementelor din fiecare din cele două mulțimi puse în corespondență de relația respectivă.

Se consideră 2 mulțimi de entități E1 și E2.

3.3.1. Relatia „unul-la-unul” (1-1 sau one to one)

Relația „unul-la-unul” este cel mai simplu tip de relație. Ea este relația prin care unui element din mulțimea E_1 îi corespunde un singur element din mulțimea E_2 și reciproc.

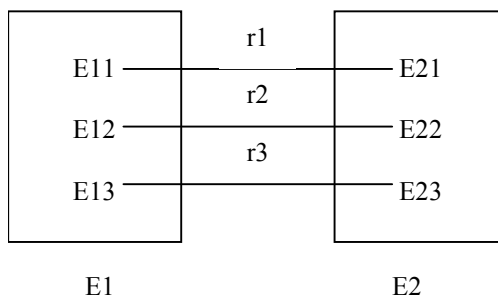


Figura 3.1. Relația „unul-la-unul”

Această relație „unul-la-unul” este foarte rar folosită în lumea reală. Cel mai des, ea este folosită pentru a reduce numărul de attribute dintr-o entitate, pentru a nu depăși numărul maxim de câmpuri asociate pentru o tabelă, acesta fiind de circa 255. Se mai poate folosi și în cazul în care dorim despărțirea elementelor fixe, a informațiilor care se modifică mai rar față de cele care se modifică destul de des.

Un exemplu ar fi la aplicația pe care o proiectăm noi, împărțirea informațiilor despre studenți în două entități: informațiile personale fiind stocate în tabelul StudPersonal și informațiile care se modifică de la an la an în tabela Studenti.

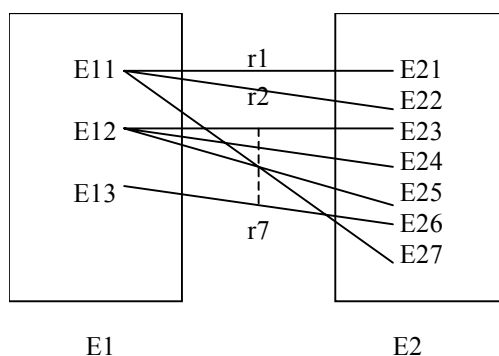
Pentru a realiza efectiv această relație trebuie introdus atributul CodStud și în tabela StudPersonal. Astfel cele două se transformă astfel:

- Studenti cu attributele: CodStud, An, Grupa, Media, Bursa
- StudPersonal cu attributele: CodStud, CNP, Nume, Init, Prenume, Data Nasterii, Loc Nașterii, Tata, Mama, Adresa

Informațiile stocate așa sunt și mai ușor de manipulat.

3.3.2. Relația „unul-la-multe” (1-N sau one to many)

Această relație este o relație prin care unui element din



mulțimea E1 îi corespund unul sau mai multe elemente din mulțimea E2, dar unui element din mulțimea E2 îi corespunde un singur element din mulțimea E1.

Figura 3.2. Relația „unul-la-multe”

O atenție sporită trebuie să avem la specificarea părților acestui tip de relație.

Un exemplu al acestui tip de relație din cadrul aplicației pe care o proiectăm noi este că într-o facultate sunt mai mulți studenți, și se precizează că un student aparține unei singure facultăți din cadrul aceleiași universități.

Pentru a realiza efectiv această relație trebuie introdus atributul CodFac și în tabela Studenți. Astfel cele două se transformă astfel:

- Facultate cu attributele: CodFac, Denumire, Adresa, Nume Decan
- Studenți cu attributele: CodStud, CodFac, An, Grupa, Media, Bursa

Observație: Dacă se adăuga atributul CodStud în tabela Facultate, se păstrau date redundante în această tabelă, adică pentru fiecare student al facultății trebuiau păstrate informații referitoare la facultate: denumirea ei, adresa, numele decanului, etc.

3.3.3. Relația „multe-la-multe” (M-N sau many to many)

Această relație este o relație prin care unui element din mulțimea E1 îi corespund unul sau mai multe elemente din mulțimea E2, și reciproc.

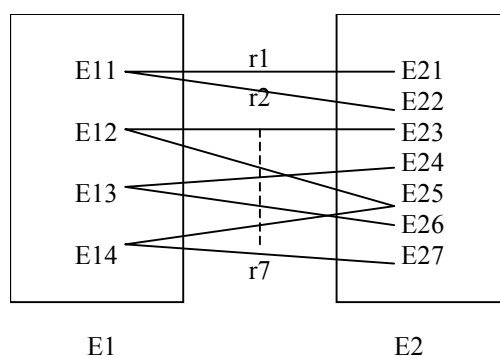


Figura 3.3. Relația „multe-la-multe”

Acest tip de relație este foarte des întâlnită, dar nu poate fi implementată în bazele de date relaționale. De fapt, pentru modelarea acestei relații se folosește o relație suplimentară, de tip unul-la-multe pentru fiecare din relațiile inițiale.

Un exemplu al acestui tip de relație din cadrul aplicația pe care o proiectăm noi este că un student participă la mai multe materii, cursuri, iar o materie este frecventată de mai mulți studenți.

Pentru a soluționa această problemă am introdus o tabelă suplimentară numită Note, care va face legătura între tabelele Materii și Studenți. Tabelele inițiale se modifică astfel:

- Studenți cu attributele: CodStud, CodFac, An, Grupa, Media, Bursa
- Materii cu attributele: CodMaterie, Denumire, An, Nume Profesor

- Note cu atributele: CodStud, CodMaterie, Nota, Data

Prin introducerea atributului Data în tabela Note s-a soluționat problema care apare atunci când un student trebuie să fie examinat de mai multe ori până la promovarea materiei respective.

3.3.4. Relația unară

Toate relațiile prezentate anterior sunt relații binare, având câte două relații implicate. Relațiile unare folosesc doar o singură relație, aceasta fiind asociată cu ea însăși.

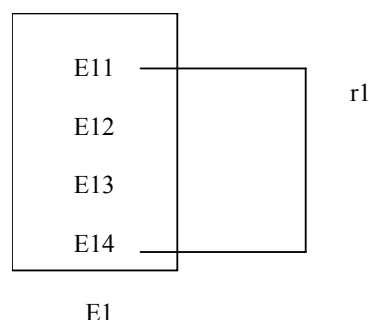


Figura 3.4. Relația unară

Exemplul clasic al acestei relații unare este cazul managerului unei companii, care la rândul său este tot un angajat al acelei companii. Relațiile unare se modelează la fel ca și relațiile binare.



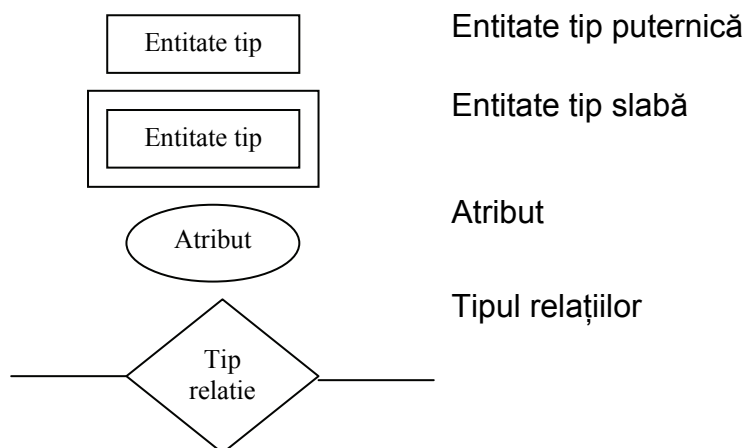
Test de autoevaluare

3. Dați exemple de relații între tabelele unei baze de date.

Pașii care trebuie urmați pentru crearea relațiilor între tabele în mediul Microsoft Access sunt descriși în capitolul 6.

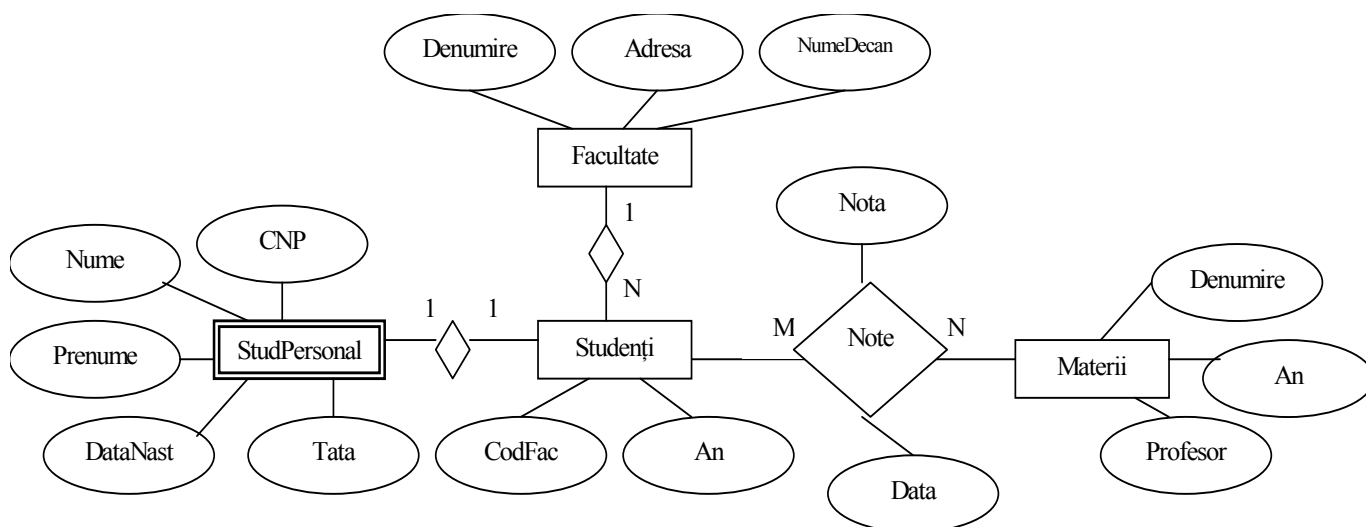
3.4. Diagrama entitate-relație

Diagrama entitate-relație este modelul entitate-relație reprezentat prin mulțimile de entități și relații dintre acestea. Există mai multe variante de notații pentru redarea acestei diagrame, astfel:



Exemplu:

În continuare, se exemplifică dezvoltarea modelului conceptual de nivel înalt al bazei de date Universitate.





9

Test de autoevaluare

4. Explicați de ce este importantă construcția diagramei entitate-relație.

3.5. Constrângeri de integritate

Constrângerile de integritate sunt reguli care se definesc la proiectarea unei baze de date și care trebuie să fie respectate de-a lungul existenței acesteia.

Entitățile unei baze de date reflectă realitatea modelată și de aceea valorile pe care le conțin trebuie să respecte anumite reguli, care să corespundă celor din realitate.

Vom folosi în continuare pentru termenul de entitate denumirea tabelă.

Constrângerile se pot clasifica astfel:

- în cadrul tabelii
- sau între tabele.

Constrângerile din cadrul unei tabele sunt reguli care se impun în cadrul unei singure tabele și asigură integritatea datelor acesteia. Ele sunt de 3 categorii:

- **constrângeri de domeniu.** Aceste constrângeri sunt condiții care se impun valorilor atributelor și asigură integritatea domeniilor atributelor.
- **constrângeri de nuplu** (de înregistrare din tabelă). Aceste constrângeri sunt condiții care se impun nuplurilor unei entități (înregistrărilor din tabelă) și asigură identificarea corectă a nuplurilor prin intermediul cheilor primare.
- **constrângeri impuse de dependențe de date** (dependențe funcționale). Acestea sunt constrângeri prin care valorile unor atribute ale unei entități (câmpuri ale tabelii) determină valorile altor atribute ale aceleiași entități.

Constrângerile între tabele sunt reguli care se impun între două sau mai multe relații. Cele mai importante sunt constrângerile de integritate referențială, care se realizează prin

intermediul cheilor străine și asigură asocierea corectă a tabelelor.

3.5.1. Constrângerile de domeniu

Constrângerile de domeniu sunt condiții impuse valorilor atributelor pentru ca acestea să corespundă semnificației pe care o au în realitatea modelată. În reprezentarea unei entități printr-un tabel, valorile atributelor sunt reprezentate pe coloane. Din această cauză aceste constrângeri se mai numesc și constrângeri de coloană.

Vom descrie 3 tipuri de constrângeri de coloană:

- *Constrângerea NOT NULL.* Valoarea NULL este o valoare particulară, care nu reprezintă valoarea 0, ci lipsă de informație. Această valoare NULL poate apărea când nu se cunosc respectivele informații, ca de exemplu, în aplicația proiectată de noi, în tabela StudPersonal nu se cunoaște numele tatălui (această informație nu este esențială). Nu orice atribut poate lua valoarea NULL, ca exemplu, numele unui student, pentru ca nu ar avea sens înregistrarea unui student al cărui nume nu se cunoaște. În astfel de situații la definirea relațiilor se impune atributului constrângerea NOT NULL, însemnând că acest atribut nu poate lua valoare NULL în orice înregistrare din tabelă
- *Constrângerea DEFAULT.* Această constrângere este folosită pentru stabilirea unei valori implicite (DEFAULT) pentru un atribut al entității. În cazul în care la inserarea unui nuplu (înregistrări) nu se specifică valoarea unui atribut (câmp), atunci acesta primește valoarea implicită (dacă a fost definită) sau valoarea NULL (dacă nu a fost definită o valoare implicită pentru atributul respectiv, dar sunt admise valori NULL). Dacă nu a fost definită o valoare implicită și nici nu sunt admise valori NULL se generează o eroare.
- *Constrângerea CHECK.* Constrângerea CHECK este după cum îi spune și numele o constrângere de verificare. În limbajul SQL, care va fi prezentat într-un capitol viitor, domeniile în care pot lua valori atributele se pot stabili ca tipuri de date predefinite. Pentru fiecare atribut se pot adăuga constrângeri de verificare la definirea tabelului.

3.5.2. Constrângerile referitoare la n-upluri (înregistrările din tabelă)-Cheia primară

O entitate este definită ca o mulțime de n-upluri. Deci, n-uplurile entității trebuie să fie distincte, acest lucru însemnând că într-o entitate nu pot exista două sau mai multe n-upluri care să conțină aceeași combinație de valori pentru fiecare atribut.

O cheie primară a unei entități (tabele) este o submulțime de attribute ale entității care are următoarele proprietăți:

- este unică, adică orice combinație de valori ale atributelor acestei chei este unică pentru orice stare a relației
- este stabilă, adică informația corespunzătoare ei nu se modifică niciodată prin operații de actualizare a datelor
- nu se admit valori NULL pentru nici unul din attributele cheii respective

În concluzie, o cheie primară reprezintă unul sau mai multe câmpuri ale tabelului care identifică unic fiecare înregistrare din tabela respectivă.

3.5.3. Constrângeri între tabele



Relațiile dintre tipurile de entități definite în modelul conceptual al unei baze de date se realizează în modelul relațional prin intermediul cheilor străine.

O cheie străină este o submulțime de attribute ale unei entități E1 care referă entitatea E2 și îndeplinește următoarele condiții: attributele cheii străine din E1 sunt definite pe domenii compatibile cu cele ale atributelor cheii din entitatea E2, și cheia din entitatea E2 este cheie primară în această relație.

Această cheie străină determină o asociere între câmpurile unor tabele cu cele ale altei tabele și creează abilitatea de realizare a unirii tabelurilor respective prin intermediul operațiilor JOIN.

Integritatea referențială este proprietatea bazei de date care garantează că oricare valoare a unei chei străine se regăsește printre valorile cheii corespunzătoare din relația referită, sau cheia străină are valoarea NULL (dacă attributele acestuia nu sunt supuse constrângerii NOT NULL).

Exemplu:

Revenind la baza de date proiectată de noi, numită Universitate vom defini cheile primare și cele străine din cadrul fiecărei tabele. Cheile primare vor fi subliniate pentru o mai clară interpretare.

- Pentru tabela Facultate vom presupune că cod facultății (CodFac) este unic pentru fiecare facultate din cadrul universității respective. În concluzie, tabela Facultate va avea următoarea structură :

Facultate (CodFac, Denumire, Adresa, Nume Decan)

- Pentru tabela StudPersonal dispunem de mai multe opțiuni în alegerea cheii primare. Ea poate fi reprezentată de CNP acesta fiind teoretic unic fiecărei persoane. Din păcate s-au întâlnit cazuri în care două persoane au același CNP și pentru a evita un posibil conflict, adăugăm la tabela un alt atribut numit CodStud și va fi un număr unic pentru fiecare înregistrare din tabela StudPersonal. În concluzie tabela StudPersonal va avea următoarea structură:

StudPersonal (CodStud, CNP, Nume, Init, Prenume, DataNasterii, LocNașt, Tata, Mama, Adresa)

- Pentru început la tabela Studenti inițială ar trebui introdus câmpul CodFac. Acesta va reprezenta o cheie străină a aceste tabele și va face legătura directă între tabela Studenti și tabela Facultate. Tabela Studenti va avea următoarea structură:

Studenti(CodFac, CodStud, An, Grupa, Media, Bursa).

Vom stabili cheia primară a acestei tabele. În acest caz, cheia primară trebuie să fie o cheie compusă din attributele CodFac și CodStud pentru a identifica unic fiecare înregistrare din tabelă. Dacă am fi ales numai câmpul CodFac nu ar fi fost bine deoarece la o facultate corespund mai mulți studenți și atunci ar apărea CodFac același pentru mai multe înregistrări și nu ar fi îndeplinite condițiile pentru ca un câmp să fie cheie primară. La fel s-ar fi întâmplat și dacă am fi optat pentru câmpul CodStud deoarece acest cod este un număr unic pentru fiecare student dintr-o facultate, același cod putând fi asociat și pentru un alt student de la o altă facultate.

În concluzie, tabela Studenti va avea următoarea structură:

Studenti(CodFac, CodStud, An, Grupa, Media, Bursa)

- Pentru tabela Materii vom presupune că codul materiei (CodMaterie) este unic pentru fiecare disciplină din cadrul universității respective. În concluzie, tabela Materii va avea următoarea structură :

Materii (CodMaterie, Denumire, An, Nume Profesor)

- În tabela Note, care are inițial două atribute Nota și Data trebuie introduse mai multe câmpuri.

Unul ar fi câmpul CodStud, care va reprezenta o cheie străină și va face legătura cu tabela Studenti pentru a cunoaște în orice moment cărui student îi aparține nota respectivă. Un alt câmp care ar trebui introdus este câmpul CodMaterie, care va reprezenta o cheie străină și va face legătura cu tabela Materii pentru a cunoaște în orice moment ce notă i-a fost acordată studentului respectiv. Tabela Studenti va avea următoarea structură:

Studenti(CodStud, CodMaterie, Nota, Data).

Vom stabili cheia primară a acestei tabele. În acest caz, cheia primară ar putea fi o cheie compusă din attributele CodStud, CodMaterie și Data pentru a identifica unic fiecare înregistrare din tabelă. Câmpul Data trebuie introdus în cheia primară deoarece un student poate să participe la un examen de mai multe ori. (fie că a fost absent, fie că nu a obținut notă de trecere, fie că dorește o mărire a notei respective). Cum este greu de manipulat o astfel de cheie primară, propunem introducerea unui alt câmp în tabela numit CodNota, care va fi unic pentru fiecare înregistrare din tabelă. Astfel, în concluzie, tabela Note va avea următoarea structură:

Note (CodNota, CodStud, CodMaterie, Nota, Data).



Test de autoevaluare

5. Ce sunt constrângerile? Dați exemple de tipuri de constrângere.

3.6. Dependente funcționale

Dependența funcțională definește relația dintre un atribut sau un grup de atribute ale unui tabel și un alt atribut sau grup de atribute ale altuia. După cum am specificat, atributele se refră la câmpurile tabelii. Prin urmare trebuie să vedeți ce câmpuri depind de alte câmpuri.

În orice tabelă pot exista două categorii de dependente funcționale:



- Dependente funcționale determinate de cheile tabelii; astfel de dependente funcționale nu produc redundanța datelor și nici anomalii de actualizare a relației
- Dependente funcționale în care atributul determinat nu este o cheie a tabelii; astfel de dependente funcționale produc redundanța datelor și anomalii de actualizare a tabelii.

Constrângerile de cheie sunt constrângeri implicite, conținute în definiția relației și sunt verificate și impuse automat de sistemul de gestiune; proiectantul bazei de date nu trebuie să prevadă nimic suplimentar pentru ca aceste constrângeri să fie satisfăcute de orice stare a relației.

În schimb, dependențele funcționale în care atributul determinant nu este o cheie a relației sunt constrângeri explicite, care nu sunt verificate și nici impuse de sistemul de gestiune. Verificarea și impunerea acestor dependente funcționale se poate face numai procedural, prin trigger, proceduri stocate sau funcții impuse în programele de aplicație.

3.7. Normalizare. Forme normale.

La proiectarea bazelor de date relaționale se stabilesc entitățile din realitatea modelată. Modul în care se pot stabili entitățile unei baze de date nu este unic și de aceea este necesar să existe criterii de evaluare a calității entităților, astfel încât acestea să asigure integritatea datelor.

În acest capitol se tratează procesul normalizării și primele trei forme normale pentru o tabelă.

Procesul de normalizare propus de E.F. Codd în 1970 urmărește execuția asupra unei tabeli a unor serii de teste pentru a cerceta apartenența la forma normală.

Codd propune trei forme normale (3NF), cea mai bună definiție fiind dată mai târziu de Boyce și Codd, fiind cunoscută sub numele de forma normală Boyce-Codd.

Normalizarea datelor poate fi privită ca un proces în timpul căruia schemele tabelă nesatisfăcătoare sunt descompuse prin împărțirea atributelor în tabele cu attribute mai puține ce posedă proprietățile dorite.

În fond, unul din obiectivele procesului de normalizare este asigurarea faptului că tabela posedă o bună construcție asigurând posibilități de modificare cu eliminarea anomaliilor care pot apărea.

Forma normală oferă proiectantului bazei de date :

- un schelet formal pentru analiza relațiilor bazat pe chei și pe dependența funcțională între attribute
- serie de teste ce pot elimina tabelele individuale astfel încât baza de date relațională poate fi normalizată în orice grad. Când un test nu este trecut, tabela va fi descompusă în tabele ce trec testele de normalitate



3.7.1. Forma normală de ordin 1 (FN1)

Forma normală de ordin 1 este considerată ca fiind parte a definiției formale a unei tabele.

Ea nu permite attribute cu mai multe valori, attribute compuse sau combinații ale lor. Aceasta stabilește ca domeniul atributelor trebuie să includă numai valori atomice și valoarea oricărui atribut într-un nuplu este o valoare unică în domeniul atributului respectiv.



Deci, FN1 nu permite un set de valori, un nuplu de valori sau o combinație a acestora ca valoare a unui atribut pentru un nuplu. Cu alte cuvinte, FN1 nu permite tabele în tabele sau tabele ca attribute ale nuplurilor. Valorile permise de FN1 sunt atomice sau indivizibile, pentru un domeniu specificat de valori.

Exemplu:

Considerăm că în tabela Materii (CodMaterie, Denumire, An, NumeProfesor), unde cheia primară este CodMaterie este introdusă o înregistrare de tipul:



CodMaterie	Denumire	An	NumeProfesor
1	Analiză matematică	1	O.Stanășilă, P.Flondor, M.Olteanu

Această înregistrare reprezintă o disciplină care este predată de trei profesori diferiți.

Acest nuplu (înregistrare) nu îndeplinește FN1, deoarece la atributul NumeProfesor nu sunt valori atomice, ci un set de valori. Pentru a rezolva această problemă vom introduce mai multe înregistrări, care vor îndeplini cerințele FN1, considerând trei materii diferite astfel:

CodMaterie	Denumire	An	NumeProfesor
1	Analiză matematică	1	O.Stanășilă
2	Analiză matematică	1	P.Flondor
3	Analiză matematică	1	M.Olteanu

3.7.2. Forma normală de ordin 2 (FN2)

A doua formă normală impune ca fiecare atribut (coloană) să fie dependent de fiecare parte a cheii principale.

Mai exact, o tabelă îndeplinește FN2 dacă îndeplinește FN1 și conține numai atribute care dau informații despre cheia tabeli.

Exemplu:



Considerând că în tabela Materii (CodMaterie, Denumire, An, NumeProfesor), unde cheia primară este CodMaterie că ar mai exista și alte câmpuri cum ar fi: Nume Student, Nota, Data Examinării. Acea structură nu ar fi proiectată bine, neîndeplinind FN2. Pentru a soluționa această problemă trebuie împărțită acea tabelă în mai multe, astfel:

Studenti(CodStud, Nume,)
 Materii (CodMaterie, Denumire, An, NumeProfesor),
 Note(Nota, Data, CodStud, Cod Materie)

3.7.3. Forma normală de ordin 3 (FN3)

Pentru a ajunge la a treia formă normală, tabelul trebuie să fie deja în prima și a doua formă normală. Pentru a fi în a treia formă normală, trebuie ca toate câmpurile non-primare să depindă numai de câmpurile primare.

Deși nu face parte în mod riguros din normalizare, de obicei nu este recomandabil să includeți câmpuri care pot fi derivate din alte câmpuri situate în același tabel sau în tabelele aflate în relație.



Exemplu:

Să considerăm baza de date Universitate, mai exact tabela StudPersonal, care are structura:

StudPersonal (CodStud, CNP, Nume, Init, Prenume, DataNasterii, LocNașt, Tata, Mama, Adresa)

Nu are rost să stocăm un alt câmp numit Vârsta, care se poate calcula din DataNasterii.

3.7.4. Forma normală Boyce-Codd (FNBC)

Forma normală Boyce-Codd este o formă strictă FN3, înțelegând prin aceasta că fiecare tabelă FNBC este în același timp o tabelă FN3, cu toate că o tabelă FN3 nu este în mod necesar și o tabelă FNBC. Cele două forme sunt asemănătoare, ambele impunând condiția ca atributul care determină funcțional alte atribute să fie o cheie a tabelii. Forma normală Boyce-Codd este mai restrictivă decât FN3, deoarece în FNBC se impune această condiție tuturor atributelor, prime sau neprime, pe când în FN3 condiția se impune numai atributelor neprime. Atributele prime sunt atributele care aparțin unei chei, iar celelalte se numesc atribute neprime.

Orice tabelă formată din două atribute este FNBC, FN2 și FN3.

Această tabelă compusă din două atribute este FN2, deoarece, fie cheia este formată din ambele atribute și atunci nu există atribute neprime, fie cheia este formată dintr-unul din atribute, iar dependența funcțională a celuilalt atribut (care este atribut neprim) față de cheie este totală.

Această tabelă compusă din două atribute este FN3 deoarece este FN2 și nu poate exista nici un atribut neprim care să determine funcțional un alt atribut neprim, deoarece o tabelă cu două atribute nu poate avea decât cel mult un atribut neprim.



Test de autoevaluare

6. Ce reprezintă procesul de normalizare și specificați formele normale.

S-ar putea să vă simțiți copleșiți de aceste reguli. De fapt, pe măsură ce veți dobândi experiență, veți începe să creați fișiere normalizate de la bun început.

3.8. Structuri de indecși în tabelele de date

Un index reprezintă o cale rapidă de localizare a înregistrărilor dintr-o tabelă, prin gruparea tuturor înregistrărilor pentru un anumit atribut sau grup de atribute.

Indexarea este utilizată în două scopuri principale:

- accelerarea căutărilor în baza de date
- asigurarea unicității înregistrărilor

Vom privi o relație ca o colecție de date (o mulțime) în care nu sunt admise elemente duplicate. În cazul unei mulțimi reprezentate printr-o colecție neordonată de elemente, timpul de căutare a unui element crește proporțional cu numărul de elemente ale mulțimii, deoarece în cazul cel mai rău trebuie parcurse toate elementele mulțimii pentru a găsi elementul dorit. Timpul de căutare a unui element poate fi micșorat considerabil dacă elementele mulțimii sunt ordonate.

Un exemplu este cel utilizat uzual în cărți. Într-o carte găsim la sfârșit termenii importanți aranjați în ordine alfabetică. La fiecare termen din această listă este furnizată un număr de pagină în care apare și este explicat termenul. Utilizând această listă se găsește imediat un termen căutat. Fără o astfel de listă, neexistând o ordine de ghidare a căutării, singura alternativă este explorarea completă a întregului material pentru a găsi termenul dorit.

În general, operațiile de căutare, inserare și ștergere a elementelor într-o mulțime (tabelă) se execută mai rapid dacă elementele mulțimii (înregistrările) sunt reprezentate printr-o colecție ordonată. În tehnologia bazelor de date, ordonarea colecțiilor de date se face prin indexarea datelor.

Indexul unei tabele este o structură de date adițională memorată în baza de date care permite accesul rapid la înregistrările tabelului prin ordonarea acestora.

De fapt, indexul poate fi gândit ca o tabelă cu două atribute: primul atribut conține valorile atributelor tabelului bazei de date pentru care se crează indexul, iar al doilea conține un pointer la locația nuplurilor corespunzătoare. Valorile sunt aranjate fie în ordine descendentă cheii de indexare, fie în ordine ascendentă.



O reprezentare generică a structurii de index este:

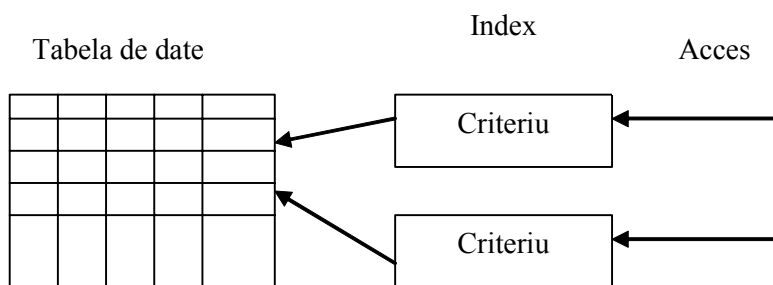


Figura 3.1. O structură generică de index

Indecșii se clasifică după tipul de câmp sau după nivel și după modul de organizare a tabeli. O clasificare a acestora este următoarea:

1. **Indexul primar** este un index asociat unei tabeli ordonate după câmpul cheie al tabeli, iar în structura de index se utilizează câmpul cheie.
2. **Indexul secundar** este un index construit tot pe baza unui câmp cheie, dar tabela nu este ordonată după câmpul cheie.
3. **Indexul de grup** (cluster) este un index construit după câmpuri ce nu sunt câmpuri cheie (criteriu de acces este diferit de câmpul cheie), iar tabela poate fi ordonată sau nu relativ la criteriul de acces.
4. **Indexul multinivel** (se mai numesc și indecși de blocuri) se aplică oricăror tabeli. Principiul de bază este de a construi niveluri de indexare până când structura adițională de date corespunde indexului de cel mai mare nivel poate fi memorată într-un singur bloc. Acești indecși se construiesc în două forme: bazați pe arbori B și bazați pe arbori B⁺.

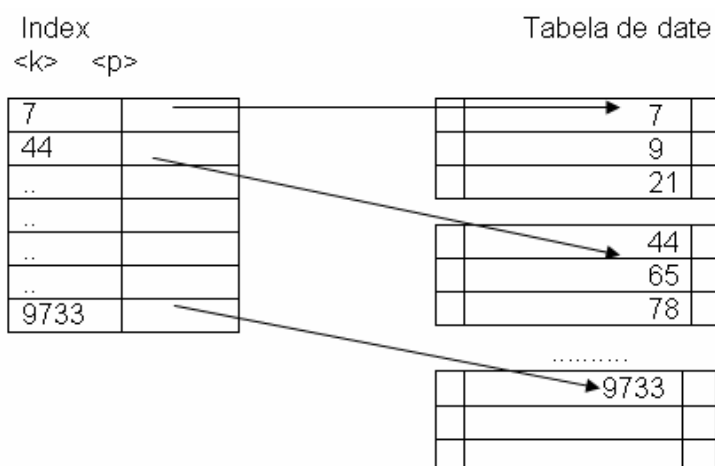


3.8.1. Indexul primar

Un index primar este un fișier ordonat cu înregistrări de lungime fixă având două câmpuri. Primul câmp al indexului este de același tip cu un câmp cheie ordonat al tabelului de date, iar al doilea câmp este un pointer către un bloc (o adresă a unui bloc).

Câmpul cheie de ordonare se mai numește și cheie primară a tabelului de date. Asociația celor două câmpuri formează intrarea index sau înregistrarea index pentru fiecare bloc al tabelului de date. Cum tabela de date este ordonată după valorile câmpului index, în fișierul index valoarea primului câmp este dată de valoarea câmpului index de la prima înregistrare a blocului. Al doilea câmp, cel ce semnifică un pointer este de tip întreg și indică adresa blocului.

Volumul datelor în index este mai mic datorită faptului că în index avem o singură intrare pentru un bloc, cât și datorită faptului că un index este similar cu o tabelă, dar are numai două câmpuri. Ca efect, căutarea într-un fișier index este mult mai rapidă decât într-o tabelă de date, putând fi utilizate metode de căutare binare.



<k> reprezintă cheia de ancorare bloc

<p> reprezintă pointerul la bloc

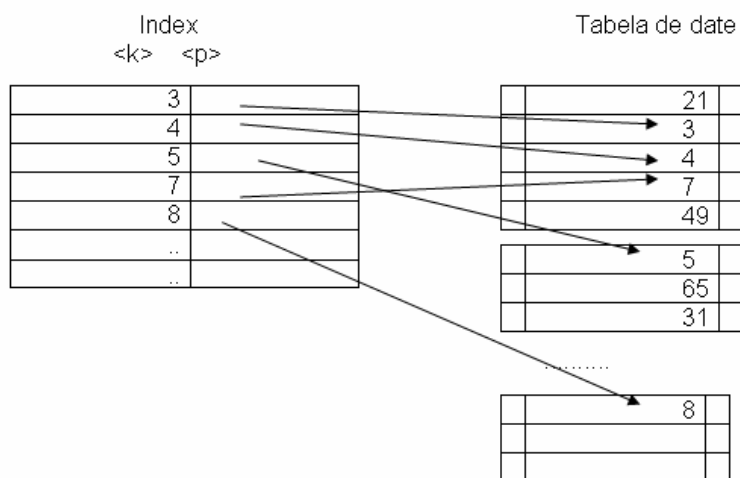
Figura 3.2. Indexul primar

3.8.2. Indexul secundar

Metoda de indexare secundară se aplică la tabele neordonate, indiferent dacă valorile câmpului după care se face indexarea în tabela de date sunt sau nu distincte.

Indexul secundar este un fișier ordonat cu două câmpuri ca și la alți indecși, în care primul câmp este identic cu cel al tabelului de date, iar al doilea câmp este un pointer. Câmpul pentru care indexul este construit se numește și câmp de indexare.

În concluzie, orice câmp al unei tabeli poate fi câmp de indexare secundar.



<k> reprezintă cheia de ancorare bloc

<p> reprezintă pointerul la bloc

Figura 3.3. Indexul secundar

3.8.3. Indexul de grup

Astfel de indecși sunt folosiți când înregistrările tabelului de date sunt ordonate fizic după un câmp care nu este cheie (noncheie), deci un câmp ce nu are valori distincte la fiecare înregistrare. Un astfel de câmp identifică un grup de înregistrări (clustering field). În această situație se poate crea un index ce facilitează găsirea înregistrărilor ce aparțin unui câmp.

Un index de grup este deci un fișier ordonat cu două câmpuri, primul câmp conținând aceeași informație cu cea a câmpului noncheie de ordonare, al doilea fiind destinat unui pointer către un bloc de date. În acest mod, fișierul index conține câte o intrare pentru fiecare valoare distinctă a câmpului de ordonare. Al doilea câmp al înregistrării index conține un pointer către blocul în care apare pentru prima oară valoarea câmpului de ordonare din primul câmp al indexului.

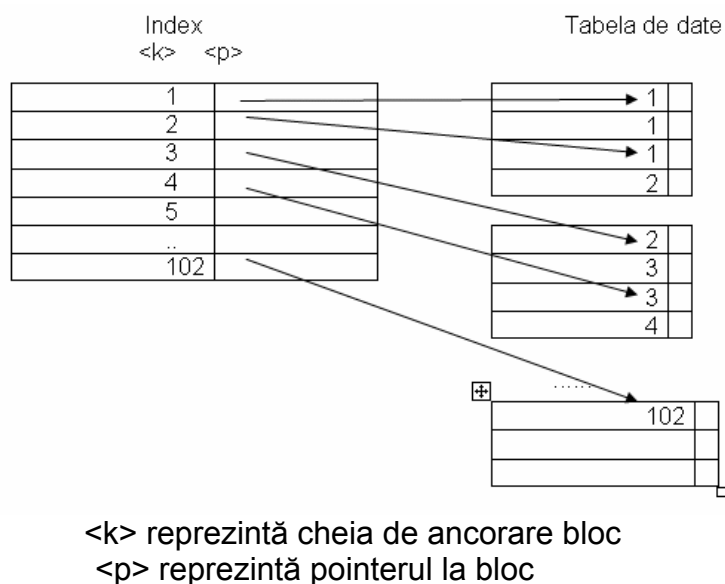
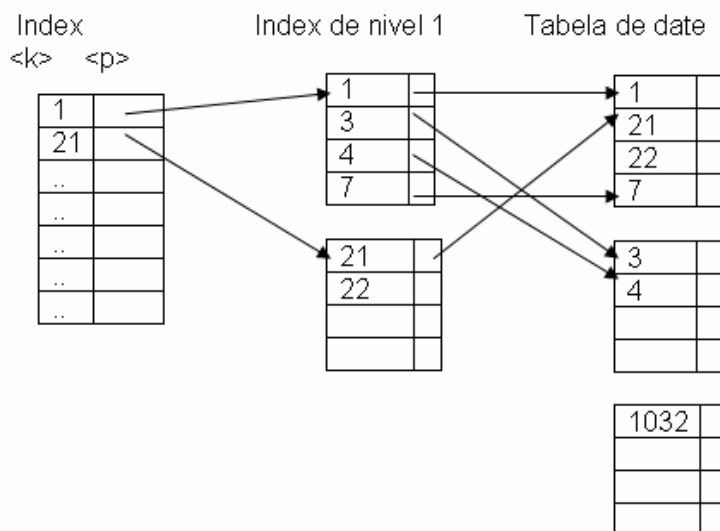


Figura 3.4. Indexul de grup

3.8.4. Indexul multinivel

Metodele de indexare descrise până acum operează cu un fișier index ordonat. Asupra fișierului index se aplică metode de căutare binară pentru localizarea înregistrărilor cu valoarea specificată în câmpul index.

Pentru un index multinivel, fișierul index este văzut ca un nou fișier la care se construiește un nou index și așa mai departe. Primul fișier index conține câte o valoare distinctă pentru fiecare cheie de indexare. Se poate crea un index primar pentru primul nivel, nivel numit și nivel secund al indexului multinivel. Cum al doilea nivel este un index primar se poate folosi metoda de ancorare a blocurilor, așa că al doilea nivel are câte o intrare pentru fiecare bloc al primului nivel, întrucât este în esență un index primar.



<k> reprezintă cheia de ancorare bloc
<p> reprezintă pointerul la bloc

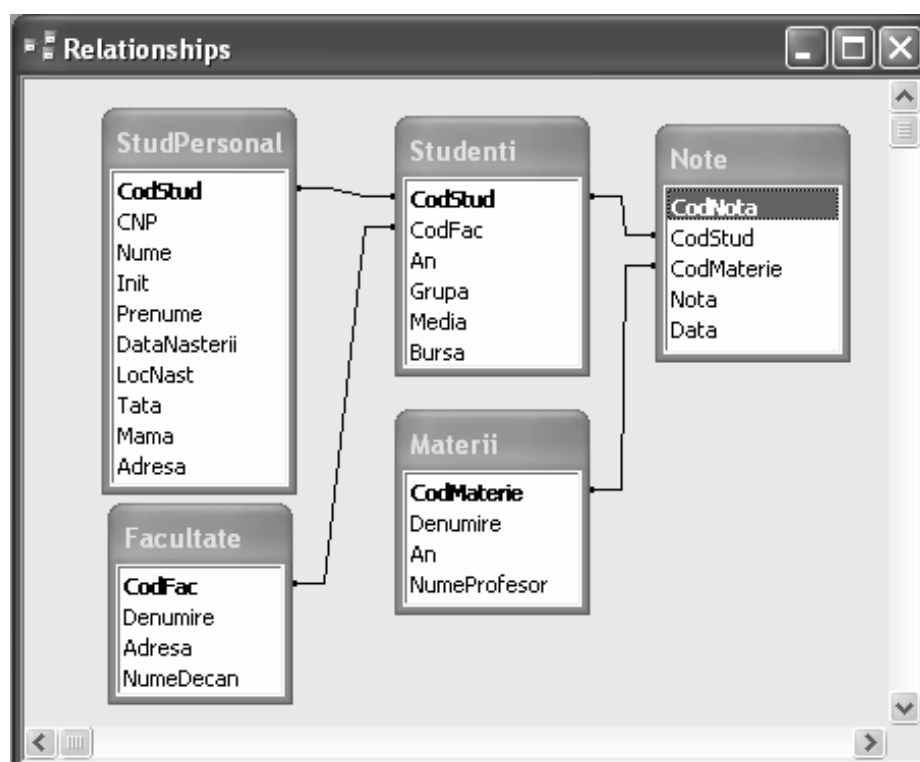
Figura 3.5. Indexul cu două niveluri



Test de autoevaluare

7. Ce sunt indecșii? Dați câteva exemple de indecși.

În imaginea alăturată este prezentată baza de date, după ce s-a realizat proiectarea ei, crearea tabelor și a relațiilor dintre ele și după ce s-au impus constrângerile.



Câmpurile scrise îngroșat sunt cheile primare din cadrul fiecărei tabele.

Liniile dintre tabele arată relațiile care le-am creat prin proiectare între fiecare două tabele. La capetele liniilor sunt două câmpuri: câmpul scris neîngroșat reprezintă cheia străină a tablei și după cum spune și definiția, face legătura cu cheia primară a celeilalte tabele.



Lucrare de verificare a cunoștințelor

1. Propun pe lângă aplicația pe care am proiectat-o și realizat-o împreună să încercați să aplicați aceleași etape și pentru o altă două aplicație descrisă mai jos.

Aplicație pentru gestiunea angajaților dintr-o companie.

Se dau mai jos cerințele pentru construcția unei baze de date aferente unei companii, baza de date numită COMPANIE, ce are ca scop ilustrarea procesului descris. În faza preliminară se cunoaște faptul că compania are un număr de angajați organizați pe departamente și urmărește realizarea unor proiecte. Pentru simplitate, se presupune că după analiza cerințelor s-a decis următoarea descriere, ca elemente primare, pentru baza de date:

- Compania este organizată în departamente, fiecare departament are un nume, un număr de cod, un număr de angajați. Compania poate avea mai multe sedii
- Un departament este implicat în mai multe proiecte, fiecare din ele are un nume, un număr de cod și o singură locație
- Se păstrează pentru fiecare angajat numele, CNP, adresa salariul, sex, data nașterii. Fiecare angajat este afiliat la un departament, însă poate lucra la mai multe proiecte ce nu sunt neapărat coordonate de același departament. Trebuie stocat și numărul de ore alocate săptămânal pentru fiecare proiect. De asemenea, fiecare angajat are un șef direct
- Lista persoanelor din întreținerea fiecărui angajat este importantă întrucât este utilizată și calculul impozitului, lista conținând numele, sexul și data nașterii fiecărui întreținut.

2. Dați exemple de entități care ar putea exista pentru baza de date Companie. Scrieți entitățile propuse de voi alături de attributele asociate fiecăreia din ele.

3. Dați exemple de relații care ar putea exista pentru baza de date Companie. Descrieți relațiile dintre toate entitățile.

4. Realizați diagrama entitate-relație asociată bazei de date Companie.

5. Definiți cheile primare și străine asociate fiecărui tabel din baza de date Companie și scrieți structura finală pentru fiecare tabelă.

6. Explicați dacă tabelele din baza de date Companie sunt normalizate. Dați exemple cum sunt îndeplinite fiecare formă normală.

Răspunsuri și comentarii la întrebările din testele de autoevaluare

Întrebarea 1.

O proiectare corectă a unei baze de date trebuie să parcurgă următoarele etape:



- Analiza cererilor și strângerea de informații referitoare la aplicație.
- Proiectarea conceptuală a bazei de date.
- Alegerea unui sistem de gestiune al bazelor de date.
- Proiectarea logică a bazei de date.
- Proiectarea fizică a bazei de date.
- Implementarea aplicației.

Întrebarea 2.

Obiectele bazei de date sunt următoarele:

- Tabelă (entitate) este o colecție de informații logice relaționale tratată ca o unitate
- Înregistrare (n-uplu). O tabelă este compusă din înregistrări sau rânduri. Fiecare înregistrare este tratată ca o simplă unitate. Fiecare înregistrare este legată de înregistrări ale altei tabelă.
- Câmpuri (attribute). Înregistrările sunt constituite din câmpuri (coloane) . Un câmp este o particulă atomică a bazei de date ce reprezintă cea mai mică cantitate de informație care poate fi manipulată. Toate înregistrările dintr-o tabelă au aceleași câmpuri.

Întrebarea 3.

Relațiile între tabele pot fi de mai multe tipuri: relație unară (în cadrul aceleiași tabelă), relația unul-la-unul, relația unul-la-multe și relația multe-la-multe (aceste 3 tipuri de relații fiind relații între două tabele).

Întrebarea 4.

Construirea diagramei entitate-relație este importantă pentru o vizualizare cât mai clară a ansamblului de obiecte din baza de date.

Întrebarea 5.

Constrângerile sunt reguli care se definesc la proiectarea unei baze de date și care trebuie să fie respectate de-a lungul existenței acesteia.

Constrângerile sunt de mai multe feluri:

- În cadrul tabelii
 - Constrângeri de domeniu
 - Constrângeri de nuplu
 - Constrângeri impuse de dependențe de date
- Între tabele
 - Constângeri de integritate referențială

Întrebarea 6.

Procesul de normalizare propus de Codd (1972) urmărește execuția asupra unei tabele a unor serii de teste pentru a cerceta apartenența la forma normală. Codd propune trei forme normale (1NF, 2NF, 3NF), cea mai bună definiție fiind dată mai târziu de Boyce și Codd, fiind cunoscută sub numele de forma normală Boyce-Codd.

Întrebarea 7.

Indexul unei tabele este o structură de date adițională memorată în baza de date care permite accesul rapid la înregistrările tabelei prin ordonarea acestora. Indecșii sunt de mai multe tipuri:

- Index primar
- Index secundar
- Index de grup
- Index multinivel

Indicații la lucrarea de verificare

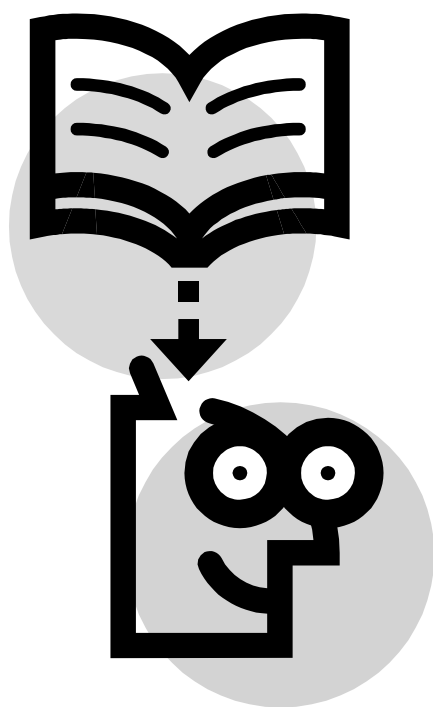
Problemele propuse în lucrarea de verificare se fac după modelele de exemple prezentate în unitatea de învățare sau după tipicul acestora, astfel:

- problema 1 ca exemplul de la pagina 31
- problema 2 ca exemplul de la pagina 33
- problema 3 ca exemplul de la pagina 34-37
- problema 4 ca exemplul de la pagina 38
- problema 5 ca exemplul de la pagina 42
- problema 6 ca exemplul de la pagina 45-47

Bibliografie:

- Cârstoiu, Dorin, *Baze de date relaționale*, Editura Printech, 1999
- Rădulescu, Florin, *Baze de date în Internet*, Editura Printech, 2000
- Ionescu, Felicia, *Baze de date relaționale și aplicații*, Editura Tehnică, 2004
- Baltac, Vasile, *ECDL-Excel, Access, PowerPoint în 20 lecții și 75 de simulări*, Editura Andreco, 2003

- Browne, Allen, Balter Alison, *Bazele Access 95*, Editura Teora, 1999
- Pribeanu, Costin, *Baze de date și aplicații*, Editura MatrixRom, 2000
- Pascu, C., Pascu A., *Totul despre SQL*, Editura Tehnică, 1994



Unitatea de învățare Nr. 4

UN LIMBAJ PENTRU BAZELE DE DATE RELAȚIONALE (SQL)

Cuprins	Pagina
Obiectivele unității de învățare nr. 4	61
4.1. Introducere	62
4.1.1. Deschiderea și închiderea aplicației Microsoft Access	62
4.1.2. Crearea unei baze de date noi	63
4.1.3. Închiderea unei baze de date	63
4.2. Tipuri de date MICROSOFT Access	64
4.3. Operatorii logici	66
4.4. Limbajul standard SQL	68
4.4.1. Scurt istoric al limbajului SQL	68
4.4.2. Crearea unei tabele	69
4.4.3. Salvarea unei tabele	72
4.4.4. Ștergerea unei tabele	73
4.4.5. Modificarea structurii unei tabele	74
4.5. Modificarea datelor în SQL	74
4.5.1. Inserarea de noi linii într-o tabelă	74
4.5.2. Ștergerea unor linii dintr-o tabelă	75
4.5.3. Modificarea unor linii dintr-o tabelă	77
4.6. Limbajul de cereri în SQL	79
4.6.1. Cereri simple	79
Expresii aritmetice	
Alias de coloană	
Constante (literali)	
4.6.2. Clauza DISTINCT	84
4.6.3. Clauza ORDER BY	85
4.6.4. Clauza WHERE	86
Operatorul BETWEEN	
Operatorul IN	
Operatorul IS NULL	
Operatorul LIKE	
4.6.5. Funcții de grup	90
4.6.6. Clauza GROUP BY	92
4.6.7. Clauza HAVING	93
4.6.8. Cereri conținând mai multe tabele	94
Lucrări de verificare a cunoștințelor	99

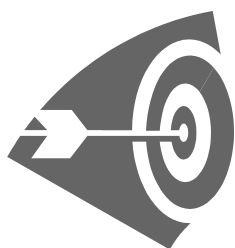
Răspunsuri și comentarii la întrebările din testele de autoevaluare	101
Bibliografie	102



Încă de la început doresc să vă felicit pentru parcurgerea cu succes a primelor trei unități de învățare și să vă urez bun venit la studiul acestei noi unități de învățare. În primele trei unități de învățare am realizat o scurtă introducere în acest amplu domeniu al bazelor de date, am descris mai în detaliu ce este un sistem de baze de date și în ce constă el, dar am învățat și cum se crează, analizează și se proiectează corect o bază de date.

În această unitate de învățare nr. 4 vom studia cum se crează, se manipulează și se șterge o bază de date și datele stocate în ea.

OBIECTIVELE unității de învățare nr. 4



Principalele obiective ale unității de învățare nr. 4 sunt:

După studiul unității de învățare nr. 4 vei fi capabil să demonstrezi că ai dobândit cunoștințe suficiente pentru a înțelege:

- cum se folosește sistemul de baze de date Microsoft Access
- cum se creează și cum se închide o bază de date Access
- care sunt tipurile de date disponibile în Access
- care sunt operatorii folosiți în algebra relațională
- ce este limbajul SQL și care sunt operațiile care se pot efectua în acest limbaj
- cum se face o interogare către baza de date cu ajutorul limbajului SQL
- cum se face grafic o interogare în Microsoft Access
- care sunt tipurile de cereri SQL
- cum se crează o tabelă, cum se modifică tabela și informațiile din tabelă, cum se inserează date în tabelă, cum se șterg datele dintr-o tabelă și cum se șterge o tabelă

4.1. Introducere

Pe parcursul acestui capitol vom face referire la baza de date analizată și proiectată în capitolele anterioare. Această bază de date se numește **Universitate** și conține următoarele tabele:

Facultate(CodFac, Denumire, Adresa, NumeDecan)

Studenti(CodStud, CodFac, An, Grupa, Media, Bursa)

StudPers(CodStud, CNP, Nume, Init, Prenume, DataNasterii, LocNast, Tata, Mama, Adresa)

Materii(CodMaterie, Denumire, An, NumeProfesor)

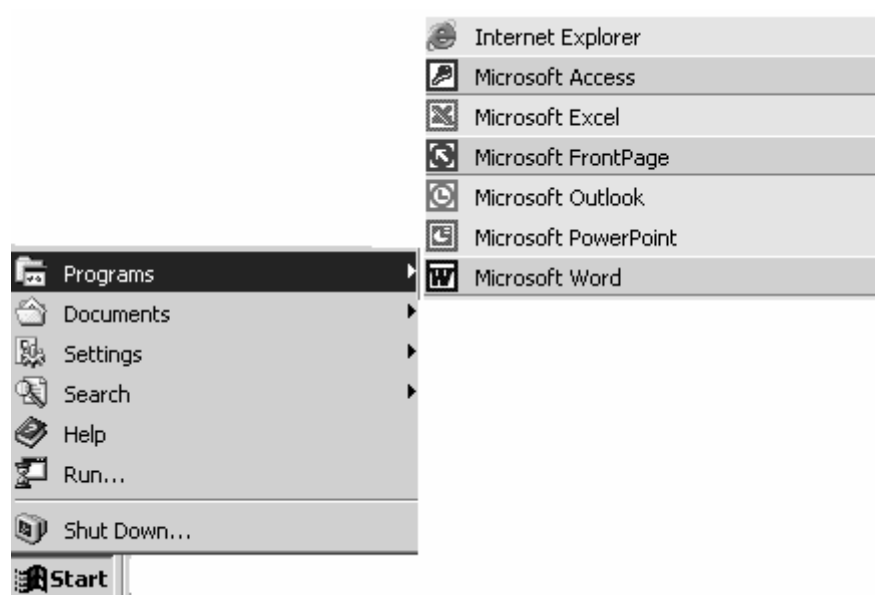
Note(CodNota, CodStud, CodMaterie, Nota, Data)

Observație: Câmpurile subliniate sunt chei primare în fiecare tabelă.

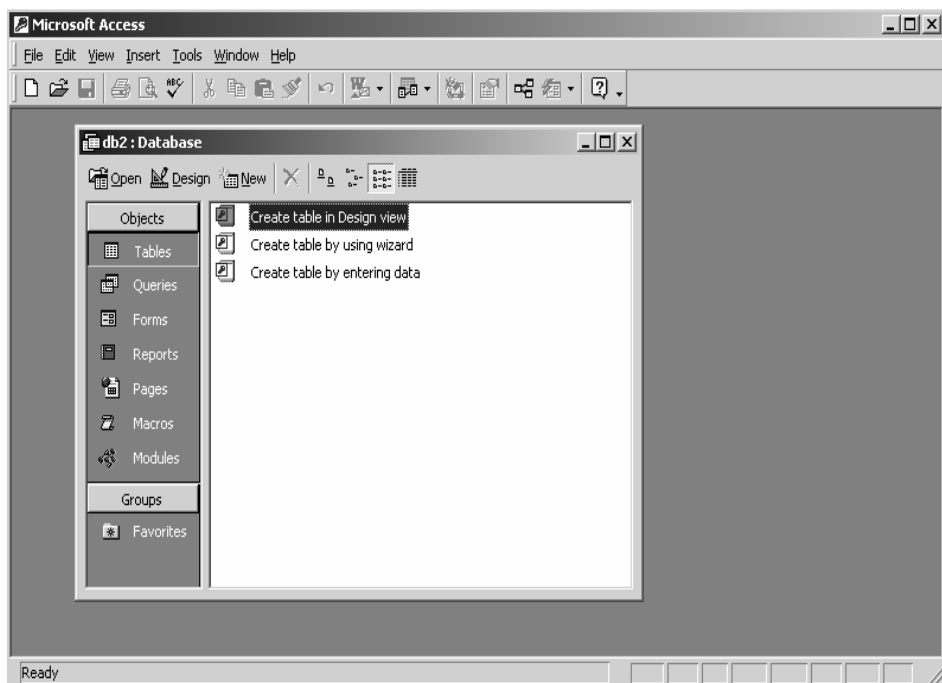
În acest capitol vor fi prezentate o serie de exemple pentru înțelegerea mai exactă a limbajului de interogări SQL. Toate exemplele au fost dezvoltate în sistemul de baze de date Microsoft Access.

4.1.1. Deschiderea și închiderea aplicației Microsoft Access

Pentru a deschide această aplicație folosim meniul **Start-Programs-Microsoft Access**.



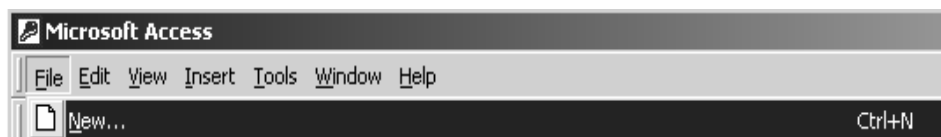
În momentul apăsării Microsoft Access, pe ecranul monitorului apare imaginea specifică acestei aplicații, ce conține: o bară de meniu, o bară de instrumente și o fereastră cu toate tipurile de obiecte care pot fi utilizate în acest program (Tables, Queries, Forms, Reports, etc).



Pentru a închide această aplicație se va alege opțiunea **Exit** din meniul **File** sau printr-un clic pe butonul din dreapta sus a ferestrei:

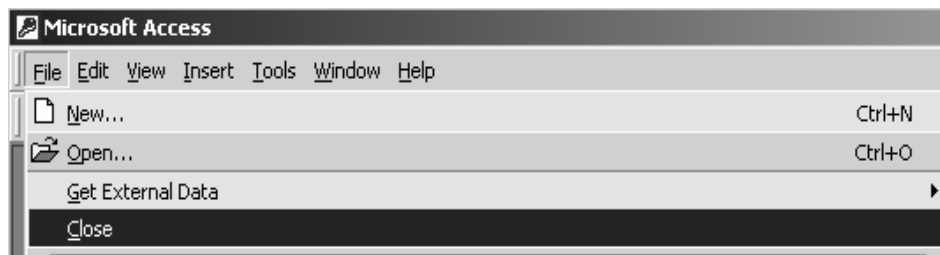
4.1.2. Crearea unei baze de date noi


O bază de date nouă se poate deschide folosind opțiunea **New** din meniul **File**, sau printr-un clic pe simbolul din bara de instrumente.



4.1.3. Închiderea unei baze de date

Pentru a închide o bază de date, fără a închide și aplicația Access ce este necesară pentru utilizări ulterioare, se utilizează comanda următoare:



sau se apasă pe butonul  din colțul din dreapta sus al ferestrei cu baza de date.

O bază de date conține unul sau mai multe tabele.



Teste de autoevaluare

1. Deschideți și închideți aplicația Microsoft Access. Explicați în câteva cuvinte.

4.2. Tipuri de date MICROSOFT Access

Vom vorbi în continuare despre crearea de tabele în mediul Microsoft Access și despre alocarea unui tip de date corespunzător pentru câmpurile din tabele.

Tabelele reprezintă obiectele din cadrul bazei de date în care se stochează datele. Un tabel este constituit din câmpuri, care sunt coloane ale acelui tabel și cărora li se atribuie câte un nume, fiecare având un tip de date și o dimensiune bine precizată. Această abordare structurată a datelor asigură bazelor de date puterea și viteza de lucru de care au nevoie.

Vom alege personal numele fiecărui câmp, tipul și proprietățile acestora, în funcție de dorințe. Înțelegerea corectă a tipurilor de câmpuri disponibile și a proprietăților acestora ne oferă posibilitatea creării unor structuri bune și eficiente pentru stocarea și gestionarea datelor.

Microsoft Access permite folosirea numelor pentru câmpuri cu o lungime de până la 64 de caractere inclusiv spațiile înglobate. Este indicat ca numele să fie scurt, fără spații și denumirea lui să fie cât mai sugestivă. Trebuie precizat tipul de date pe care îl conține fiecare câmp.

Aceste tipuri de date disponibile în Access sunt:

- **Text**

Majoritatea câmpurilor folosite în baza de date sunt de tipul text. Pe lângă câmpurile care au tipul evident text, cum ar fi:

Nume, Adresa, câmpurile text se mai pot folosi și pentru numerele care nu au funcție matematică. De exemplu, se folosesc câmpurile text pentru stocarea numerelor de telefon, a codului poștal etc. Dimensiunea prestabilită a unui câmp text în Access este de 50 caractere, dar putem alege orice dimensiunea între 1 și 255.

- **Memo**

Dimensiunea unui câmp de tipul Memo este de 64KB (kiloocteți), asta însemnând că putem stoca aproximativ 16 pagini de text scris la un rând pentru fiecare înregistrare. Câmpurile memo constau numai din text. Pot fi incluse secvențe de salt la început de rând sau salt la rând nou, dar nu se acceptă opțiuni de formatare a textului sau indentarea unui paragraf.

- **Număr**

Tipul număr (Number) include mai multe tipuri care diferă prin modul de stocare și viteza de răspuns. Acestea sunt:

1. tipul **Byte** (Octet), care acceptă numai valori întregi pozitive până la 255
2. tipul **Integer** (întreg), care acoperă domeniul numerelor întregi de la -32768 la 32768
3. tipul **Long Integer** (Întreg lung), care reprezintă numerele întregi până dincolo de limitele de plus și minus 2 miliarde
4. tipul **Single**, care acoperă domeniul numerelor fracționare cu până la 7 cifre semnificative
5. tipul **Double**, care acoperă domeniul numerelor fracționare cu până la 14 cifre semnificative
6. tipul **ReplicationID** (IdentificatorDuplicare), care este reprezentată pe 16 octeți și a fost creată cu scopul de a asigura un identificator unic global (GUID: Globally Unique Identifier) pentru bazele de date în cazul cărora trebuie realizată sincronizarea unor mari cantități de date suplimentare, de la mai multe surse.

- **Data calendaristică/Oră.**

Microsoft Access stochează intern datele calendaristice sub forma unor numere în virgulă mobilă pe 8 octeți, ora fiind reprezentată ca o fracțiune dintr-o zi. În general este suficientă în majoritatea situațiilor opțiunea Short Date (Forma scurtă a datei), deși s-ar putea să dorim ca anul să fie reprezentat pe formatul de 4 cifre.

- **Valută**

Tipul valută (Currency) acoperă valorile în dolari întregi până la 15 cifre și valorile zecimale până la sutimi de cent. Ca reprezentare internă, tipul valută este un număr în virgulă fixă.

Acest format asigură o precizie sporită calculelor financiare, dar este mai lent decât tipurile de numere întregi sau în virgulă mobilă.

- **Număr cu incrementare automată**

În Microsoft Access, datele de tip număr cu incrementare automată (AutoNumber) pot fi secvențiale sau aleatoare. Microsoft Access permite stabilirea dimensiunii unui număr cu incrementare automată, prestabilit fiind întreg lung. Numerele cu incrementare automată sunt atribuite chiar în momentul în care începem să adăugăm o înregistrare nouă. În cazul ștergerii unei înregistrări existente, nu există nici o posibilitate de creare a unei alte înregistrări cu același număr.

- **Da/Nu**

Tipul Da/Nu (Yes/No) stochează numai valori true sau false (adevărat sau fals). O astfel de valoare poate fi afișată în una din formele True/False, Yes/No sau On/Off.

- **Obiect OLE**

Tipul de câmp OLE este destinat păstrării datelor provenite de la alte programe, care s-au înregistrat ele însele ca servere OLE în Windows. Aceasta permite bazei de date să stocheze documentele create de programe de prelucrare a textelor, seturi de foi de calcul, ilustrații, sunete, videoclipuri, etc.

- **Program wizard de căutare**

Programul wizard de căutare (Lookup Wizard) nu este câtuși de puțin un alt tip de câmp. El reprezintă o metodă convenabilă de crearea a unui câmp care îndeplinește funcția de căutare într-un alt tabel.

Vom reveni la limbajul SQL.

Vom prezenta ca o introducere pentru limbajul SQL, operatorii logici pe care îi vom folosi la interogări.

4.3. Operatorii logici

Majoritatea operațiilor algebrei relaționale implică folosirea operatorilor logici, operatori care de obicei întorc un rezultat boolean – rezultat care are valoarea *true* (adevărat) sau *false* (fals). Spunem de obicei deoarece dacă adăugăm valoarea *null* la modelul relațional, lucrurile se complică puțin. *Null* adaugă o a treia valoare la setul de valori boolene. Această valoare este prezentată pe larg în capitolul anterior. În concluzie, în algebra relațională se folosesc trei valori: true, false, null.

Operatorii logici folosiți în algebra relațională sunt: AND, OR, XOR.

Tabelele de adevăr pentru operatorii logici standard sunt:

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	NULL
FALSE	TRUE	FALSE	NULL
NULL	NULL	NULL	NULL

XOR	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL
FALSE	TRUE	FALSE	NULL
NULL	NULL	NULL	NULL

După cum se vede, null *op* orice, unde *op* este un operator logic, întoarce valoarea null.

Alți operatori (=, ≠) sunt prezentați mai jos:

=	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	TRUE	NULL
NULL	NULL	NULL	NULL

≠	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL
FALSE	TRUE	FALSE	NULL
NULL	NULL	NULL	NULL

SQL ne mai pune la dispoziție încă 2 operatori unari- IS NULL și IS NOT NULL pentru manipularea valorilor nule.

	IS NULL	IS NOT NULL
<valoare>	FALSE	TRUE
TRUE	FALSE	TRUE
FALSE	FALSE	TRUE
NULL	TRUE	FALSE

<valoare> reprezintă orice în afară de valoarea nulă.

4.4. Limbajul standard SQL

4.4.1. Scurt istoric al limbajului SQL

SQL a fost conceput ca un limbaj standard de descriere a datelor și acces la informațiile din bazele de date, ulterior dezvoltându-se ca o adevărată tehnologie dedicată arhitecturilor client-server.

Utilizat inițial de către firma IBM pentru produsul DB2, limbajul de interogare al bazelor de date relaționale SQL a devenit la mijlocul deceniului trecut un standard în domeniu. De atunci și până în prezent au fost dezvoltate un număr de 7 versiuni ale standardului SQL, trei dintre acestea aparținând Institutului Național American de Standarde (ANSI), celelalte fiind concepute de firme de prestigiu ca IBM, Microsoft, Borland, sau de către consorțiile industriale SAG (The SQL Access Group) și X/Open, primul format din sute de firme ce comercializează software pentru baze de date, iar cel din urmă orientat spre activități de promovare a standardelor în domeniul sistemelor deschise. Din păcate, lipsa unui standard unic SQL are drept consecințe creșterea costurilor programelor de gestiune a bazelor de date și îngreunează întreținerea arhitecturilor client/server.

Termenul SQL reprezintă o prescurtare a Structured Query Language.

Comenzile principale în cazul limbajului SQL se referă la cele cinci operații de bază care se pot efectua într-un limbaj relațional:

- Crearea/ștergerea unei tabele
- Inserarea de noi linii într-o tabelă
- Ștergerea unor linii dintr-o tabelă
- Modificarea unor linii dintr-o tabelă
- Listarea selectivă a datelor din una sau mai multe tabele

În acest mediu există două moduri pentru crearea interogărilor: modul de scriere efectivă a cererilor în partea de Queries sau un mod grafic mult mai prietenos utilizatorului numit Design View. Vom prezenta interogările și în primul mod: SQL VIEW, dar și în modul grafic.



Teste de autoevaluare

2. Care sunt operatorii algebrei relaționale?
3. Ce reprezintă prescurtarea SQL?

4.4.2. Crearea unei tabele

Comanda de creare de noi tabele în baza de date curentă în limbajul SQL standard este CREATE TABLE.

Sintaxa simplificată pentru această comandă este următoarea:


```
CREATE TABLE nume_tabela (
    coloana_1 descriere_1,
    coloana_2 descriere_2,
    .....,
    coloana_n descriere_n,
    [alte_descrieri]
)
```

unde coloana_x este numele coloanei, iar descriere_x conține tipul valorilor acelei coloane și alte elemente de descriere pentru ea. În descrierea unei coloane se poate specifica, pe lângă tipul valorilor sale și alte constrângeri de integritate ca:

- NOT NULL indică faptul că valorile aferente coloanei respective nu pot avea valori de tip null, care nu înseamnă zero, ci lipsă de informație.
- PRIMARY KEY indică faptul că coloana specificată cu această constrângere va fi cheie primară pentru acest tabel.
- FOREIGN KEY necesită ca fiecare valoare din coloană să existe într-o coloană corespondentă dintr-o tabelă referită. Constrângerea FOREIGN KEY poate face referire doar la

coloane care sunt PRIMARY KEY sau UNIQUE în tabela referită.

- DEFAULT indică o valoare implicită care îl ia un câmp al unei table.

Interogările prezentate mai departe vor fi făcute în modul SQL VIEW. Pentru a intra în acest mod ne poziționăm pe obiectul Queries și apăsăm butonul . Va apărea o fereastră pentru alegerea tipului de creare a interogării. Facem opțiunea Design View și vom face opțiunea View – SQL View din noul meniu sau apăsăm clic dreapta de la mouse pe fereastra Query și facem opțiunea SQL View.

Exemplu:



Crearea tabelor cu structura prezentată la începutul capitolului.


```
create table Facultate(  
    CodFac integer primary key,  
    Denumire text(50),  
    Adresa text(50),  
    NumeDecan text(20));  
  
create table StudPersonal(  
    CodStud integer primary key,  
    CNP integer,  
    Nume Text(25),  
    Init text(3),  
    Prenume text(20),  
    DataNasterii date,  
    LocNast text(50),  
    Tata text(30),  
    Mama text(30),  
    Adresa text(50));  
  
create table Studenti(  
    CodStud integer primary key,  
    CodFac integer,  
    An byte,  
    Grupa text(6),  
    Media double,  
    Bursa integer);  
  
create table Materii(  
    CodMaterie integer primary key,  
    Denumire text(30),  
    An byte,  
    NumeProfesor text(50));
```

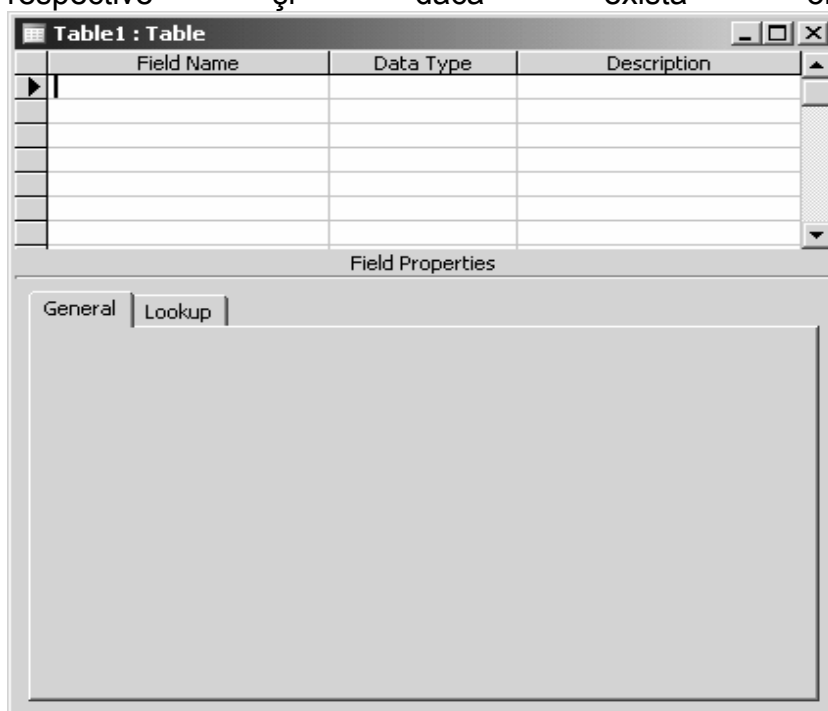


```
create table Note (
    CodNota integer autonumber primary key,
    CodStud integer,
    CodMaterie integer,
    Nota byte,
    Data date);
```

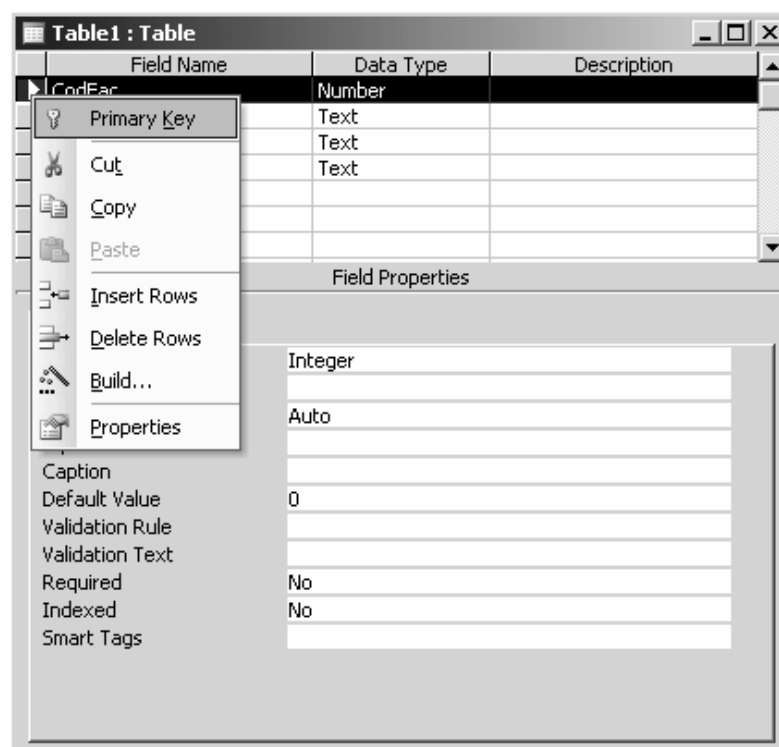
După rularea acestor fraze SQL vor apărea la obiectul *Tables* cele 5 tabele noi create.

Cea de-a doua metodă de creare a tabelelor, mai exact metoda grafică este și cea mai des folosită de toți utilizatorii.

Pentru a crea o tabelă în modul Design se face opțiunea  Create table in Design view . Va apărea o fereastră în care trebuie completată denumirea câmpului, tipul de date asociat câmpului respective și dacă există observații.




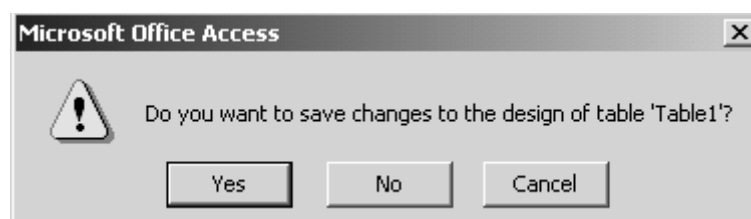
Vom exemplifica decât crearea tabelii Facultate. După completarea denumirii câmpurilor în zona Field Name și a tipurilor de date în zona Data Type vom seta cheia primară. Acest lucru se face poziționându-ne cu cursorul de la mouse pe partea din stânga câmpului corespunzător cheii primare (în cazul nostru CodFac), se apasă clic dreapta și se face opțiunea Primary Key.



Se salvează tabela cu un nume dat de utilizator.

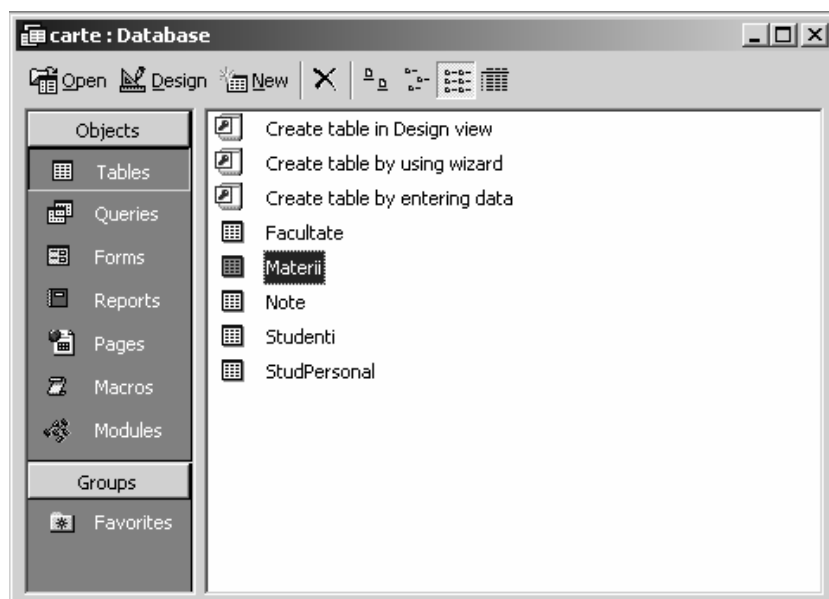
4.4.3. Salvarea unei tabele

O tabelă se salvează cu opțiunea Save din meniul File, sau printr-un clic pe simbolul  din bara de instrumente. Va apărea un mesaj pentru confirmarea salvării tablei.



Aceeași pași trebuie urmați și pentru celelalte tabele.

În final, partea de obiecteTables va arăta astfel:



4.4.4. Ștergerea unei tabele

Ștergerea unei tabele se face cu comanda DROP TABLE. Sintaxa acestei comenzi în limbajul SQL standard este:

DROP TABLE nume_tabelă

Exemplu:

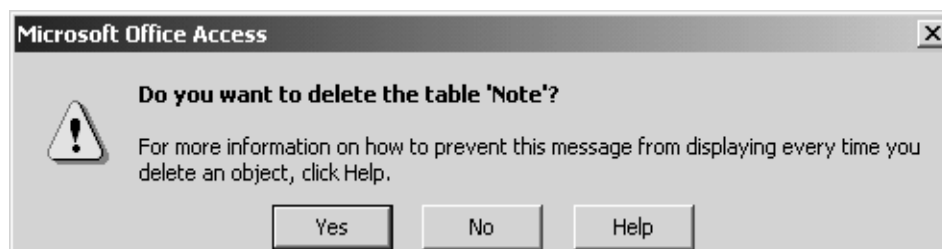
Ștergerea tablei Note se face astfel:

drop table Note;



Ștergerea unei tabele în modul grafic se face astfel: ne poziționăm în obiectul Tables, unde sunt afișate toate tabelele din baza de date, apăsăm clic dreapta de la mouse pe tabelul dorit pentru ștergere și facem opțiunea Delete.

Va apărea un mesaj de confirmare:



Se va apăsa butonul Yes dacă se dorește într-adevăr ștergerea tablei sau se apasă butonul No dacă se dorește revenirea asupra operației de ștergere.

4.4.5. Modificarea structurii unei tabele



Modificarea structurii unei tabele în limbajul SQL standard se face cu comanda ALTER TABLE. Această comandă este folosită pentru a adăuga coloane la tabele de bază din baza de date sau pentru a șterge anumite constrângeri.

O nouă coloană adăugată prin această comandă va avea valoarea null în toate înregistrările care existau în tabelă.

Sintaxa acestei comenzi este:

```
ALTER TABLE nume_tabela  
ADD coloana_n descriere_n [DROP constrangere]
```

Exemplu:

Adăugarea câmpului Ora de tip integer tablei Note se face astfel:

```
alter table Note add Ora integer;
```

Modificarea structurii unei tabele în modul grafic se face astfel: ne poziționăm în obiectul Tables, unde sunt afișate toate tabelele din baza de date, apăsăm clic dreapta de la mouse pe tabelul dorit pentru modificare și facem opțiunea Design. Va apărea o fereastră cu structura tablei. Se vor face modificările dorite și apoi se salvează tabelul.



4.5. Modificări ale datelor în SQL

4.5.1. Inserarea de noi linii într-o tabelă

Comanda INSERT care permite inserarea de noi linii într-o tabelă are următoarea sintaxă simplificată în limbajul SQL standard:

```
INSERT INTO nume_tabela [(nume_coloana, ...)]  
VALUES (valoarea_coloana_1, valoare_coloana_2,...)
```

Această comandă ne permite inserarea manuală de noi înregistrări. Dacă este prezentă lista de coloane (nume_coloana, ...) înseamnă că se dau doar valori pentru aceste coloane, pentru celelalte asignându-se valori de null.

Exemplu:

Introducerea unor înregistrări în tabela Facultate se face astfel:

```
insert into Facultate values  
(1,'Electrotehnica','Noul Local',''),  
(2,'Energetica','Noul Local',''),
```

```
(3,'Automatica','Noul Local','Dumitru Popescu'),
(4,'Electronica','Leu',''),
(5,'Aeronave','Polizu',''),
(6,'Mecanica','Noul Local',''),
(7,'Transporturi','Noul Local','');
```

După rularea aceste fraze SQL, tabela Facultate va arăta astfel:

Facultate : Table				
	CodFac	Denumire	Adresa	NumeDecan
	1	Electrotehnica	Noul Local	
	2	Energetica	Noul Local	
	3	Automatica	Noul Local	Dumitru Popescu
	4	Electronica	Leu	
	5	Aeronave	Polizu	
	6	Mecanica	Noul Local	
	7	Transporturi	Noul Local	

Dacă un câmp de tip integer nu conține nici o valoare se va scrie NULL, iar dacă este de tip text se va lăsa spațiu.

Inserarea datelor într-un tabel în modul grafic se face astfel: ne poziționăm în obiectul Tables, unde sunt afișate toate tabelele din baza de date, apăsăm clic dreapta de la mouse pe tabelul în care dorim să introducem date și facem opțiunea Open. Va apărea o fereastră cu datele deja existente în tabelă. Se vor face inserările de date dorite și apoi se salvează tabelul.

4.5.2. Ștergerea unor linii dintr-o tabelă

Sintaxa simplificată a comenzii SQL în limbajul standard care șterge liniile dintr-o tabelă este următoarea:

```
DELETE FROM nume_tabela
[WHERE conditie] [LIMIT numar_linii]
```



Efectul acestei comenzi este de ștergere a liniilor care îndeplinesc condiția din clauza WHERE. LIMIT se folosește pentru a specifica numărul maxim de linii care se pot șterge cu acea comandă. În cazul în care clauza WHERE lipsește, toate liniile tablei vor fi eliminate.

Exemplu:

Ștergerea studenților din tabela Studenti care au media mai mică de 7.

delete from Studenti where medie<7;

Înainte:

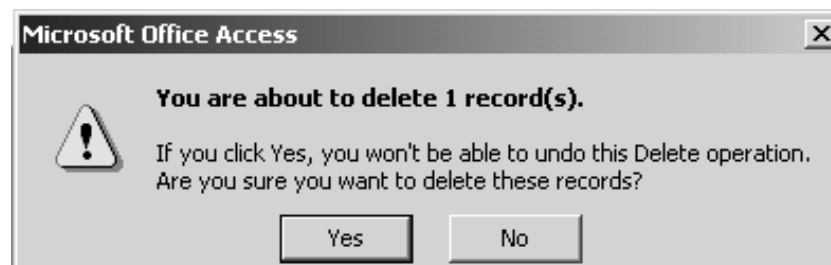
	CodStud	CodFac	An	Grupa	Media	Bursa
	1	3	1	312AA	8.12	3000000
	2	3	4	344AB	9.25	4000000
	3	1	1	313BB	7.34	
	4	3	3	333AC	6.45	
	5	2	5	1212AC	7.77	
	6	3	1	314AA	9.13	4000000
		0	0		0	0

Record: 7 of 7

După rularea acestei interogări se va șterge înregistrarea a-4-a, aceasta îndeplinind condiția media<7.

Ștergerea datelor dintr-o tabelă în modul grafic se face astfel: ne poziționăm în obiectul Tables, unde sunt afișate toate tabelele din baza de date, apăsăm clic dreapta de la mouse pe tabelul din care dorim să ștergem anumite date și facem opțiunea Open. Va apărea o fereastră cu datele existente în tabela respectivă. Pentru a șterge o înregistrare ne poziționăm în partea stângă a înregistrării dorite pentru ștergere, apăsăm clic dreapta al mouse-ului și facem opțiunea Delete Record.

Va apărea un mesaj de confirmare:



Se va apăsa butonul Yes dacă se dorește într-adevăr ștergerea înregistrării respective sau sa apasă butonul No dacă se dorește revenirea asupra operației de ștergere.

4.5.3. Modificarea unor linii dintr-o tabelă

Sintaxa simplificată a comenzii SQL în limbajul standard care modifică liniile dintr-o tabelă este următoarea:

```
UPDATE nume_tabela SET colana1=valoare1,
coloana2=valoare2, ....
[WHERE conditie] [LIMIT numar_linii]
```

Efectul acestei comenzi este de actualizare a tuturor liniilor care îndeplinesc condiția din clauza WHERE, sau a tuturor liniilor din tabelă, în cazul în care lipsește această clauză. Noile valori sunt date de clauza SET.

Exemplu:

1. Modificarea numelui studentului cu CNP=333333, din Cornel în Vasilescu.



```
update StudPersonal set nume='Vasilescu' where CNP=333333;
```

Tabela StudPers înainte:

	CodStud	CNP	Nume	Init	Prenume	DataNasterii	LocNast	Tata	Mama	Adresa
▶	1	111111	Ion	A.	Adrian	11/04/1987	Bucuresti	Ion	Maria	Buc
	2	222222	Vasile	I.	Alexandru	01/05/1086	Timisoara	Iancu	Mioara	Timis
	3	333333	Cornel	A.	Rodica	12/09/1999	Brasov			
	4	444444	Alecu	M.	Petre	13/08/1943	Bacau			
	5	555555	Mihai	L.	Dumitru	08/08/1977	Teleorman	Ion	Ioana	Teleorman
	6	666666	Bucur	V.	Valeriu	11/07/1987	Bucuresti	Gabriel	Ana	Bucuresti

Record: 1 of 6

Tabela StudPers după:

	CodStud	CNP	Nume	Init	Prenume	DataNasterii	LocNast	Tata	Mama	Adresa
	1	111111	Ion	A.	Adrian	11/04/1987	Bucuresti	Ion	Maria	Buc
	2	222222	Vasile	I.	Alexandru	01/05/1086	Timisoara	Iancu	Mioara	Timis
	3	333333	Vasilescu	A.	Rodica	12/09/1999	Brasov			
	4	444444	Alecu	M.	Petre	13/08/1943	Bacau			
	5	555555	Mihai	L.	Dumitru	08/08/1977	Teleorman	Ion	Ioana	Teleorman
	6	666666	Bucur	V.	Valeriu	11/07/1987	Bucuresti	Gabriel	Ana	Bucuresti

Record: 7 of 7

2. Mărirea tuturor burselor studenților cu 10%.

update Stumenti set bursa=bursa*1,1;

Tabela Stumenti înainte:

Stumenti : Table						
	CodStud	CodFac	An	Grupa	Media	Bursa
▶	1	3	1	312AA	8.12	3000000
	2	3	4	344AB	9.25	4000000
	3	1	1	313BB	7.34	
	5	2	5	1212AC	7.77	
	6	3	1	314AA	9.13	4000000

Record: 1 of 5

Tabela Stumenti după:

Stumenti : Table						
	CodStud	CodFac	An	Grupa	Media	Bursa
▶	1	3	1	312AA	8.12	3300000
	2	3	4	344AB	9.25	4400000
	3	1	1	313BB	7.34	
	5	2	5	1212AC	7.77	
	6	3	1	314AA	9.13	4400000

Record: 1 of 5

Modificarea datelor dintr-un tabel în modul grafic se face astfel: ne poziționăm în obiectul Tables, unde sunt afișate toate tabelele din baza de date, apăsăm clic dreapta de la mouse pe tabelul dorit pentru modificare și facem opțiunea Open. Va apărea o fereastră cu datele deja existente în tabelă. Se vor face modificările asupra datelor dorite și apoi se salvează tabelul.



Teste de autoevaluare

4. Care sunt operațiunile care se pot efectua în cadrul unei baze de date?

4.6. Limbajul de cereri în SQL

Regăsirea datelor din una sau mai multe tabele se face cu comanda SELECT. Sintaxa simplificată a acesteia este:

```
SELECT [DISTINCT] lista_rezultat
FROM tabela sau tabele
[WHERE conditie]
[GROUP BY coloana1, coloana2....]
[HAVING conditie_de_grup]
[ORDER BY coloana1 [ASC|DESC],...]
```

După cum se observă doar clauzele SELECT și FROM sunt obligatorii, celelalte reprezentând opțiuni.

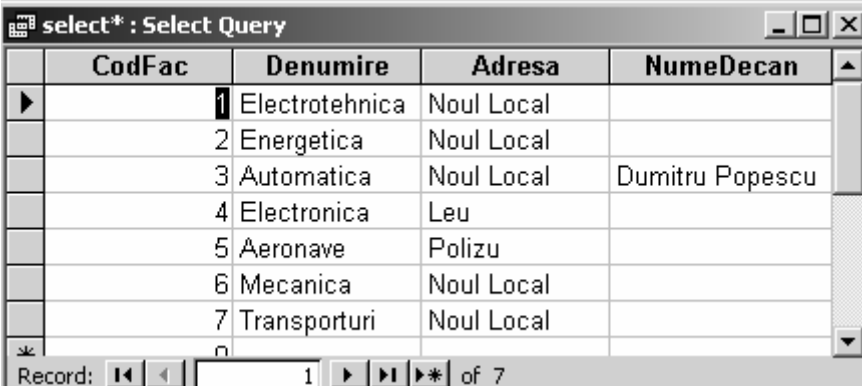
4.6.1. Cereri simple

Cea mai simplă cerere este cea prin care se regăsesc toate informațiile dintr-o tabelă. În acest caz, simbolul * plasat lângă clauza SELECT ține loc de lista tuturor coloanelor unei tabele. Clauza FROM conține numele tablei.

Exemplu:

1. Selectarea tuturor informațiilor din tabela Facultati.



select * from Facultate;




	CodFac	Denumire	Adresa	NumeDecan
▶	1	Electrotehnica	Noul Local	
	2	Energetica	Noul Local	
	3	Automatica	Noul Local	Dumitru Popescu
	4	Electronica	Leu	
	5	Aeronave	Polizu	
	6	Mecanica	Noul Local	
	7	Transporturi	Noul Local	

Record: 1 of 7

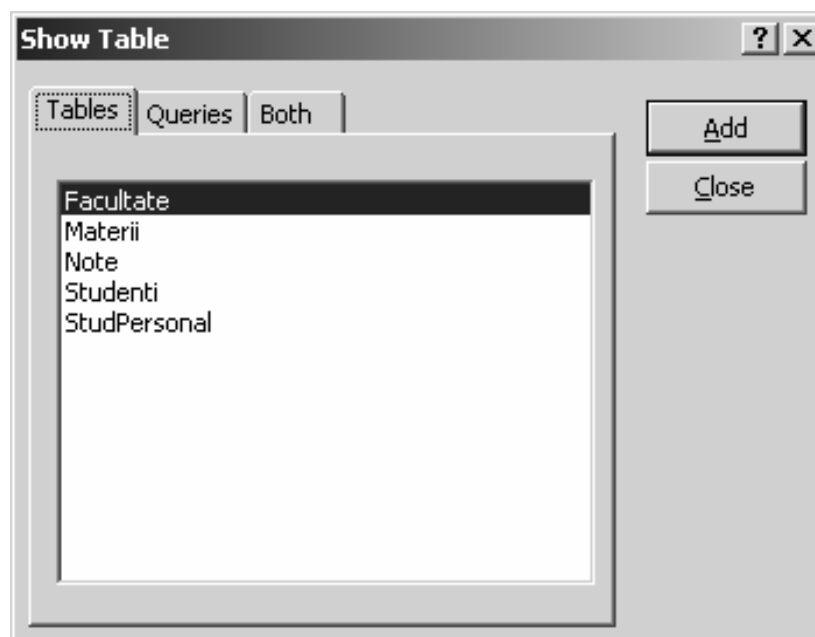
Această interogare se face în modul grafic astfel: ne poziționăm în partea de Queries și facem opțiunea

-  Create query in Design view pentru crearea manuală de interogări sau
-  Create query by using wizard pentru utilizarea wizardului.

Vom exemplifica realizarea unei interogări în Design View, acest lucru fiind posibil apăsând dublu clic pe opțiunea

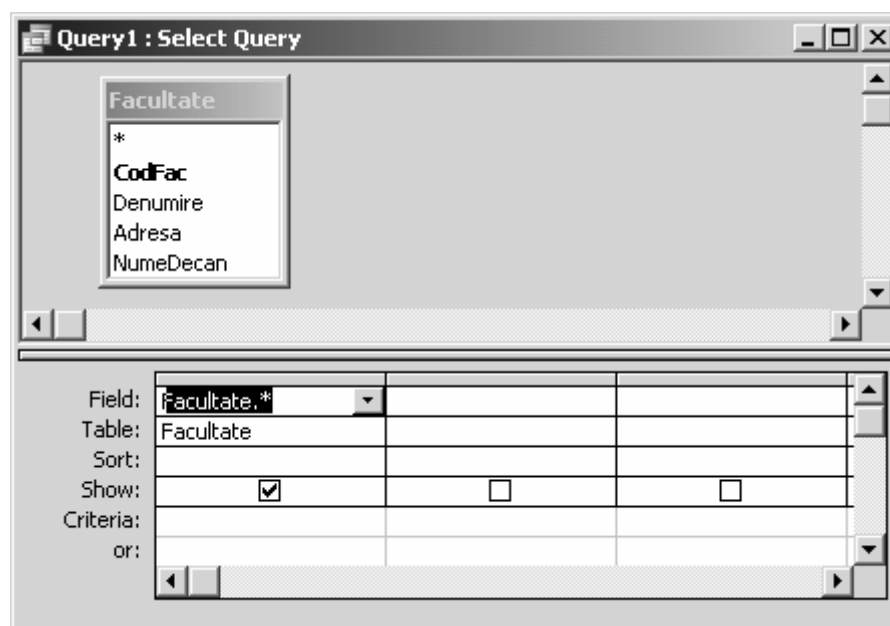
-  Create query in Design view


Va apărea o fereastră în care sunt afișate toate tabelele din baza de date.

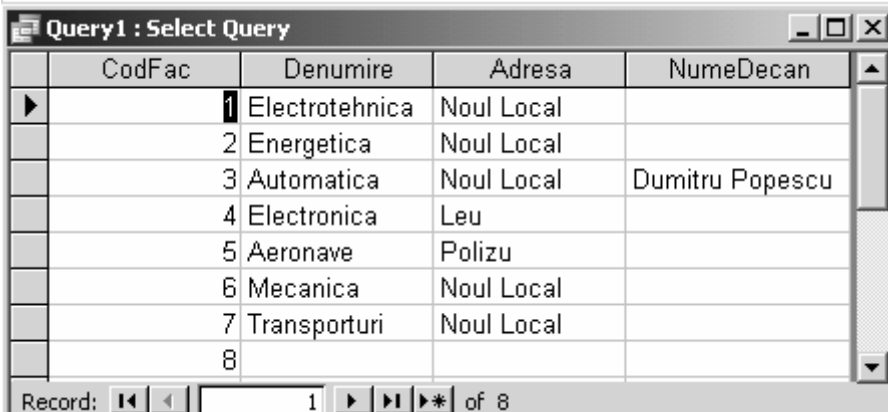


Vom alege acele tabele necesare pentru interogare. De exemplu, vom realiza o interogare care după rulare îmi va afișa toate înregistrările din tabela *Facultate*. Vom alege tabela *Facultate* și în partea de jos alegem ce câmpuri doresc să fie afișate după rularea interogării.

Caracterul * reprezintă toate câmpurile dintr-o tabelă.



Pentru rularea efectivă a interogării se apasă butonul Run . Rezultatele obținute în urma rulării vor apărea într-o fereastră astfel:



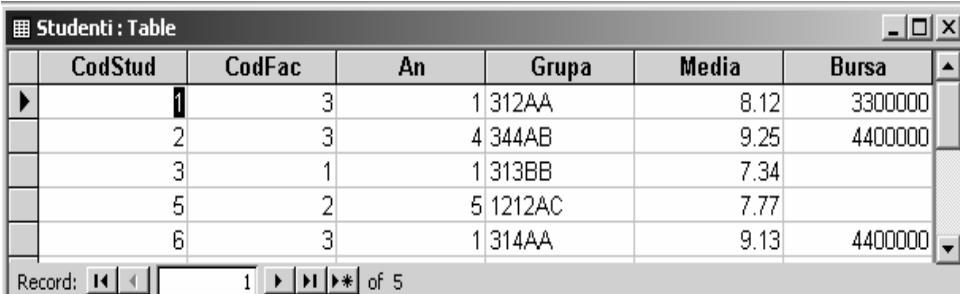
	CodFac	Denumire	Adresa	NumeDecan
▶	1	Electrotehnica	Noul Local	
	2	Energetica	Noul Local	
	3	Automatica	Noul Local	Dumitru Popescu
	4	Electronica	Leu	
	5	Aeronave	Polizu	
	6	Mecanica	Noul Local	
	7	Transporturi	Noul Local	
	8			

Record: 1 of 8

Pentru orice tip de interogare pașii care trebuie urmați de utilizatori sunt aceiași. Cei prezentați mai sus.

2. Selectarea tuturor informațiilor din tabela Studenti.

`select * from Studenti;`



	CodStud	CodFac	An	Grupa	Media	Bursa
▶	1	3	1	312AA	8.12	3300000
	2	3	4	344AB	9.25	4400000
	3	1	1	313BB	7.34	
	5	2	5	1212AC	7.77	
	6	3	1	314AA	9.13	4400000

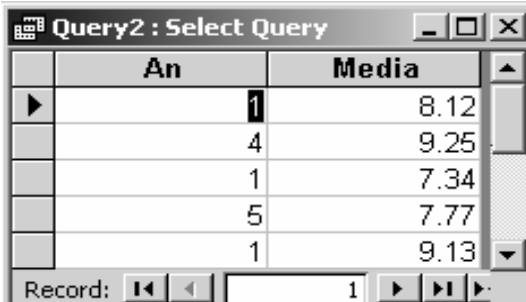
Record: 1 of 5

În cazul în care se doresc regăsite doar o parte a coloanelor unei tabelă acestea sunt enumerate în clauza SELECT.

Exemplu:

Selectarea anului și a mediei din tabela Studenti.

`select An, Media from Studenti;`



	An	Media
▶	1	8.12
	4	9.25
	1	7.34
	5	7.77
	1	9.13

Record: 1



În clauza SELECT pot fi prezente, pe lângă nume de coloane, o serie de alte construcții, cum ar fi:

a. Expresii aritmetice

Operatorii care pot fi utilizați sunt cei uzuali: +,-,*,/ paranteze. De remarcat că o expresie care conține o valoare nulă se evaluează la NULL.

Exemplu:

Afișarea mediei, bursa, a bursei+1 și a (bursesi+1)*100 din tabela Studenti.

```
select Media, bursa, bursa+1,(bursa +1)*100 from Studenti;
```



Query2 : Select Query				
	Media	bursa	Expr1002	Expr1003
▶	3.12	3300000	3300001	330000100
	9.25	4400000	4400001	440000100
	7.34			
	7.77			
	9.13	4400000	4400001	440000100
*	0	0		
Record: 1 of 5				

b. Alias de colană

În cazul în care clauza SELECT conține expresii, numele coloanelor din rezultat sunt date de acestea. Dacă se dorește ca în rezultat coloana respectivă să aibă alt nume, acesta se poate specifica cu AS nume_nou.

Exemplu:

Afișarea numelui studentului cu denumirea Nume_Student, a prenumelui cu denumirea Prenume_Student din tabela StudPersonal.

```
select nume as Nume_Student, prenume as Prenume_Student
from StudPersonal;
```



Query2 : Select Query

	Nume_Student	Prenume_Student
▶	Ion	Adrian
	Vasile	Alexandru
	Vasilescu	Rodica
	Alecu	Petre
	Mihai	Dumitru
	Bucur	Valeriu
*		

Record: 1 of 6

Dacă se dorește ca aliasul să conțină spații el trebuie pus între apostrofi:

```
select CNP, nume as 'Nume Student', prenume as 'Prenume Student' from StudPersonal;
```

Query2 : Select Query

	CNP	'Nume Student'	'Prenume Student'
▶	111111	Ion	Adrian
	222222	Vasile	Alexandru
	333333	Vasilescu	Rodica
	444444	Alecu	Petre
	555555	Mihai	Dumitru
	666666	Bucur	Valeriu
*	0		

Record: 1 of 6

c. Constante (literali)

Dacă în lista SELECT se găsesc și constante atunci pe acele coloane toate liniile rezultatului vor conține valorile respective:

Exemplu:

Selectarea numelui studentului, textul 'este născut la data de' și DataNasterii din tabela StudPersonal.

```
select nume, 'este nascut la date de', DataNasterii from StudPersonal;
```



Query2 : Select Query			
	nume	textul	DataNasterii
▶	Ion	este nascut la date de	11/04/1987
	Vasile	este nascut la date de	01/05/1086
	Vasilescu	este nascut la date de	12/09/1999
	Alecu	este nascut la date de	13/08/1943
	Mihai	este nascut la date de	08/08/1977
	Bucur	este nascut la date de	11/07/1987
*			
Record: 1 of 6			


4.6.2. Clauza DISTINCT

În cazul cererilor de până acum din fiecare linie a tabelului rezultă o linie a rezultatului, chiar dacă uneori unele linii sunt identice, ca în cazul următor:

Exemplu:

Afișarea codului numeric al facultății pentru care există studenți în tabela Studenti;

`select CodFac from Studenti;`



Query...	
	CodFac
	3
	3
	1
	2
	3
▶	
Record: 1 of 6	

Pentru a elimina liniile duplicat ale unui rezultat se folosește clauza DISTINCT care apare între cuvântul cheie SELECT și lista de elemente ale rezultatului.

`select distinct CodFac from Studenti;`

Query...	
	CodFac
▶	1
	2
	3
Record: 1 of 3	

4.6.3. Clauza ORDER BY

Ordinea în care apar liniile unui rezultat este dată de modul în care un sistem de gestiune stochează și regăsește informația în tabele. Din această cauză în limbajul SQL există posibilitatea de a sorta liniile unui rezultat în funcție de necesitățile utilizatorului.

Criteriile de sortare se definesc cu ajutorul clauzei ORDER BY. Aceasta este în mod normal ultima clauză care apare într-o cerere și poate conține nume de coloane, aliasuri de coloane sau numărul de ordine al coloanei în rezultat. Când sunt specificate mai multe criterii ele se aplică de la stânga la dreapta.

Exemplu:

Afișarea CNP, nume, prenume, data nașterii din tabela StudPersonal ordonați după nume și prenume.

```
select CNP, Nume, Prenume, DataNasterii from StudPersonal
order by nume, prenume;
```



	CNP	Nume	Prenume	DataNasterii
▶	444444	Alecu	Petre	13/08/1943
	666666	Bucur	Valeriu	11/07/1987
	111111	Ion	Adrian	11/04/1987
	555555	Mihai	Dumitru	08/08/1977
	222222	Vasile	Alexandru	01/05/1086
	333333	Vasilescu	Rodica	12/09/1999
*	0			
Record: ◀ ◻ ▶ ▶▶ * of 6				

După cum se poate observa ordinea de sortare implicită este cea ascendentă (crescătoare):

- Pentru numere - de la valoarea cea mai mică la cea mai mare
- Pentru șiruri de caractere - ordinea este cea lexicografică, din dicționar
- Pentru date calendaristice - de la cea mai veche dată la cea mai nouă

Inversarea ordinii implicite se poate face pentru fiecare criteriu de sortare în parte folosind cuvântul cheie DESC (descendent) plasat după criteriul respectiv.

O problemă importantă este tratarea valorilor nule (NULL) de clauza ORDER BY. Aceste valori sunt considerate a fi mai mici decât orice altă valoare, deci vor apărea primele pentru sortarea implicită și ultimele pentru sortarea descendentă.

Exemplu:

Afișarea în ordine ascendentă a rezultatului după valoarea bursei și o sortare descendentă după codul studentului în tabela Studenți.



`select CodStud, Bursa from Studenti order by Bursa, CodStud desc;`

CodStud	Bursa
5	
3	
1	3300000
6	4400000
2	4400000
	0

Record: 6

4.6.4. Clauza WHERE

Până acum, în afara cazurilor în care se utilizează DISTINCT, din fiecare linie a tabelului rezultă o linie a rezultatului. Prin folosirea clauzei WHERE se poate specifica o condiție care indică liniile din tabela care vor avea asociată câte o linie din rezultat.

Exemplu:

Afișarea media și valoarea bursei pentru studenții de la facultatea cu codul 3 din tabela Studenți.



`select Media, Bursa from Studenti where CodFac=3;`

Media	Bursa
8.12	3300000
9.25	4400000
9.13	4400000
*	0

Record: 1

Operatorii care se pot folosi sunt cei obișnuiți:

- Egal: =
- Mai mic, mai mic sau egal: <, ≤
- Mai mare, mai mare sau egal: >, ≥
- Diferit: ≠, !=

De asemenea se pot folosi paranteze și conectori logici:

- AND sau && (pentru și)
- OR sau || (pentru sau)
- NOT sau ! (pentru negare, inversarea condiției)

Exemplu:



Afișarea codului facultății, media, grupa, anul și valoarea bursei pentru studenții care au bursa egală cu 4400000 și media mai mare sau egală cu 9 sau sunt la facultatea cu codul 3.

```
select CodFac, Media, Grupa, An, Bursa from Studenti where  
(Bursa=4400000 and media>=9) or CodFac=3;
```

	Media	Grupa	An	Bursa
▶	8.12	312AA	1	3300000
	9.25	344AB	4	4400000
	9.13	314AA	1	4400000
	9.03	544AB	4	4400000
*	∅		∅	∅

Record: 1 of 4

Pe lângă operatorii de mai sus care sunt prezenți în majoritatea limbajelor de programare există însă în SQL patru operatori specifici. Aceștia au fost introduși pentru a simplifica anumite categorii de cereri sau pentru a specifica condiționări care nu pot fi exprimate prin operatori obișnuiți.

Acești operatori sunt:

a. Operatorul BETWEEN

Sintaxa: **between** valoare_iniciala **and** valoare_finala

Operatorul between indică o plajă de valori incluzând valorile din capetele acestuia, cele indicate. Este un operator derivat, astfel de condiții putând fi scrise folosind >=, AND, <= și a fost introdus pentru ca cererile să fie mai apropiate de exprimarea în limba engleză.

Exemplu:

Afișarea codului facultății, media, grupa pentru studenții care au media cuprinsă între 8 și 9 inclusiv.

```
select CodFac, Media, Grupa from Studenti where media between 8 and 9;
```

	CodFac	Media	Grupa
▶	3	8.12	312AA
*	0	0	

Record: 1 of 1

b. Operatorul IN

Sintaxa: IN (v1,v2,...vk)

Operatorul IN indică apartenența la o mulțime de valori (v1,v2,...vk). Este de asemenea un operator derivat, dar este foarte util pentru simplificarea scrierii cererilor în cazul în care mulțimea conține un număr mare de valori.

Exemplu:

Afișarea codului facultății, codul studentului, grupa, bursa pentru studenții care au codul facultății 1 sau 3.

```
select CodFac, CodStud, Grupa, Bursa from Studenti where CodFac in (1,3);
```

	CodFac	CodStud	Grupa	Bursa
▶	3	1	312AA	3300000
	3	2	344AB	4400000
	1	3	313BB	
	3	6	314AA	4400000
*	0	0		0

Record: 1 of 4

c. Operatorul IS NULL

Operatorul IS NULL a fost introdus pentru a se putea testa dacă o valoare a unei expresii este nulă, deoarece valorile nule nu pot fi detectate cu operatori de comparație obișnuiți.

Exemplu:

Afișarea codului facultății, codul studentului, grupa pentru studenții care nu au bursa.

```
select CodFac, CodStud, Grupa from Studenti where bursa is null;
```

	CodFac	CodStud	Grupa	Bursa
▶	1	3	313BB	
	2	5	1212AC	
	4	8	766	
*	0	0		0

Record: 1 of 3

Pentru negarea acestui operator s-a făcut și o excepție de la sintaxa standard a expresiilor. Astfel în loc de “not is null”, sintaxa SQL prevede forma IS NOT NULL.

Exemplu:

Afișarea codului facultății, codul studentului, grupa pentru studenții care au bursa.

```
select CodFac, CodStud, Grupa from Studenti where bursa is not null;
```

	CodFac	CodStud	Grupa	Bursa
▶	3	1	312AA	3300000
	3	2	344AB	4400000
	3	6	314AA	4400000
	2	7	544AB	4400000
*	0	0		0

Record: 1 of 4

d. Operatorul LIKE

Sintaxa: **LIKE** 'șablon'

Operatorul LIKE a fost introdus pentru a se putea testa potrivirea valorii unei expresii cu un șablon. Șablonul poate conține caractere care vor fi căutate așa cum sunt și caracterele speciale:

- % însemnând orice șir de caractere (inclusiv unul vid)
- _ însemnând orice caracter

De exemplu 'I_S%L' este un șablon pentru un șir de cel puțin 4 caractere care începe cu litera I, are al treilea caracter S și

ultimul caracter este L. Cu acest șablon se potrivesc de exemplu șirurile: INSTABIL, IXSTL, IOSL.

Operatorul se poate folosi inclusiv pentru expresii de alt tip decât șirurile de caractere, valoarea acestora fiind întâi convertită la șir de caractere și apoi verificată potrivirea cu șablonul.



Exemplu:

1. Afișarea codul studentului, numelui, prenumelui, data nașterii pentru studenții care au numele format din 5 litere din tabela StudPersonal.

```
select CodStud, Nume, Prenume, DataNasterii from StudPersonal  
where nume like '_____';
```

2. Afișarea codul studentului, numelui, prenumelui, data nașterii pentru studenții care au prenumele care începe cu litera A din tabela StudPersonal.

```
select CodStud, Nume, Prenume, DataNasterii from StudPersonal  
where Prenume like 'A%';
```

4.6.5. Funcții de grup. Clauzele GROUP BY și HAVING

Există multe cazuri în care se dorește obținerea de date statistice din informațiile conținute în baza de date. Pentru aceasta se folosesc funcțiile de grup, care pe baza înregistrărilor din întreaga tabelă sau a celor care fac parte dintr-un grup - în cazul existenței clauzei GROUP BY - calculează valoarea statistică respectivă.

Principalele funcții de grup sunt:

a. Funcția COUNT (numărare) având mai multe forme:

COUNT(*) întoarce numărul de înregistrări din grup

COUNT(expr) întoarce numărul de valori nenule pentru expresia argument

COUNT(DISTINCT expr) întoarce numărul de valori distincte pentru expresia argument

b. Funcția AVG (medie)

AVG(expr) întoarce media aritmetică a valorilor unei expresii

c. Funcția MIN (valoarea minimă)

MIN(expr) întoarce valoarea minimă a unei expresii

d. Funcția MAX (valoarea maximă)

MAX(expr) întoarce valoarea maximă a unei expresii

Funcțiile Min și MAX se pot aplica și șirurilor de caractere, în acest caz se folosește ordinea lexicografică.

e. Funcția SUM (suma valorilor)

SUM(expr) întoarce suma valorilor unei expresii sau NULL în cazul în care în grupul pentru care se calculează suma este vid. Valorile nule nu sunt luate în considerare la calcularea sumei.

Așa cum am menționat, în cazul în care cererea nu conține clauza GROUP BY valoarea funcțiilor este calculată pentru întreaga tabelă specificată în clauza FROM.

Exemplu:

Afișarea valorii minime, maxime și media pe coloana media, suma pe coloana bursa și numărul de înregistrări din tabela Studenti.

```
select min(Media), max(Media), avg(Media), sum(Bursa), count(*)
from Studenti;
```



Query2 : Select Query					
	Expr1000	Expr1001	Expr1002	Expr1003	Expr1004
▶	7.34	9.25	8.33	16500000	7
Record:	1 of 1				

Funcțiile de grup se pot folosi și în conjuncție cu celelalte clauze discutate anterior.

Exemplu:

Afișarea mediei mediilor studenților de la facultatea cu codul 3.

```
select avg(Media) from Studenti where CodFac=3;
```



Query2...	
	Expr1000
▶	3.8333333333
Record:	

4.6.6. Clauza GROUP BY


Clauza GROUP BY se folosește pentru a grupa înregistrările pe baza unor criterii în scopul calculării de valori statistice pentru fiecare grup în parte. În acest caz rezultatul cererii va conține câte o linie pentru fiecare grup identificat.

Sintaxa: GROUP BY coloana1 [, coloana2...]

Exemplu:

Afișarea grupată a înregistrărilor din tabela Studenti după valoarea coloanei bursa și obținerea unui rezultat conținând valoarea bursei, numărul de studenți având acea bursă și suma burselor din fiecare grup.

```
select Bursa,count(*),sum(Bursa) from Studenti group by Bursa;
```




Bursa	Expr1001	Expr1002
	3	
3300000	1	3300000
4400000	3	13200000

Record: 1 of 3

În cazul în care în GROUP BY apar mai multe coloane, un grup va fi construit din toate înregistrările care au valori comune pe toate coloanele specificate.

Exemplu:

```
select CodFac,Bursa,sum(Bursa) from Studenti where CodFac in (2, 3) group by CodFac,Bursa;
```



CodFac	Bursa	Expr1002
2		
2	4400000	4400000
3	3300000	3300000
3	4400000	8800000

Record: 1 of 4

Atenție: în cazul folosirii funcțiilor de grup în clauza SELECT nu pot apare alături de acestea decât valori care sunt constante pentru fiecare grup în parte- în principal numele coloanelor după care s-a făcut gruparea.

De exemplu, cererile următoare sunt greșite:

```
select CodFac,max(Media), avg(Media) from Studenti;
```

```
select CodStud, Bursa, sum(Bursa) from Studenti group by CodFac;
```

4.6.7. Clauza HAVING

Dacă WHERE introduce o condiție de filtrare a înregistrărilor, HAVING face același lucru pentru grupuri: doar grupurile care îndeplinesc condiția conținută în această clauză vor avea o linie în rezultatul cererii.

Condiția specificată prin HAVING este o expresie logică incluzând funcții de grup sau constante la nivel de grup.

Exemplu:



O cerere în două variante: fără și apoi cu o clauză HAVING care elimină o parte din grupuri.

```
select CodFac, count(*) as NumarStudenti, avg(Media) as MediaFacultatii from Studenti group by CodFac;
```

	CodFac	NumarStudenti	MediaFacultatii
▶	1	1	7.34
	2	2	8.4
	3	3	8.83333333333
	4	1	7.67

Record: 1 of 4

```
select CodFac, count(*) as NumarStudenti, avg(Media) as MediaFacultatii from Studenti group by CodFac having avg(Media)>8;
```

	CodFac	NumarStudenti	MediaFacultatii
▶	2	2	8.4
	3	3	8.83333333333

Record: 1 of 2

4.6.8. Cereri conținând mai multe tabele

Operația prin care se obține un rezultat pe baza datelor din mai multe tabele se numește JOIN. Pentru a se putea efectua un JOIN este în general necesar ca tabelele să aibă coloane comune, cum este cazul CodFac care se găsește atât în tabela Studenti, cât și în tabela Facultate.

În cazul în care clauza FROM sunt specificate mai multe tabele toate celelalte clauze sunt evaluate pornind de la produsul cartezian al tabelor. Acesta este obținut (teoretic) prin concatenarea fiecărei linii a unei tabele cu fiecare dintre liniile celorlalte tabele.

De exemplu, în tabelele de mai sus, produsul cartezian va conține $7 \times 7 = 49$ linii, deoarece sunt 7 linii în tabela Studenti și 7 linii în tabela Facultate.

Pentru eliminarea liniilor inconsistente – cum este concatenarea înregistrării unui student al facultății 1 cu înregistrarea facultății 2 – este necesar ca în clauza WHERE să existe așa-numita *condiție de join*. În cazul existenței de coloane comune între tabele, aceasta este o condiție de egalitate a valorilor acelor coloane.

Exemplu:



1. Afișarea numelui și prenumelui studentului (din tabela StudPers) și a grupei din care face parte, anul și media (informații aflate în tabela Studenti).

```
select Nume,Prenume,Grupa,An,Media
from Studenti,StudPersonal
where Studenti.CodStud=StudPersonal.CodStud;
```

Query2 : Select Query					
	Nume	Prenume	Grupa	An	Media
▶	Ion	Adrian	312AA	1	8.12
	Vasile	Alexandru	344AB	4	9.25
	Vasilescu	Rodica	313BB	1	7.34
	Mihai	Dumitru	1212AC	5	7.77
	Bucur	Valeriu	314AA	1	9.13
Record: 1 of 5					

Condiția de JOIN este `Studenti.CodStud=StudPersonal.CodStud`, unde construcția `tabela.coloana` a fost folosită deoarece câmpul `CodStud` este în ambele tabele cu același nume.

2. Afișarea numelui facultății (din tabela `Facultate`), a numelui și a prenumelui studentului (din tabela `StudPers`) și a grupei din care face parte, anul și media (informații aflate în tabela `Studenti`).

```
select Denumire,Nume,Prenume,Grupa,An,Media
from Facultate,Studenti,StudPersonal
where Studenti.CodStud=StudPersonal.CodStud and
Facultate.CodFac=Studenti.CodFac;
```

Denumire	Nume	Prenume	Grupa	An	Media
Automatica	Ion	Adrian	312AA	1	8.12
Automatica	Vasile	Alexandru	344AB	4	9.25
Electrotehnica	Vasilescu	Rodica	313BB	1	7.34
Energetica	Mihai	Dumitru	1212AC	5	7.77
Automatica	Bucur	Valeriu	314AA	1	9.13

Record: 1 of 5

2. Afișarea numelui facultății (din tabela `Facultate`), numelui și prenumelui studentului (din tabela `StudPers`) și a grupei din care face parte, anul și media (informații aflate în tabela `Studenti`) grupate pe facultăți.

Vom folosi aliași pentru numele tabelelor.

```
select Denumire,Nume,Prenume,Grupa,An,Media
from Facultate F,Studenti S,StudPersonal P
where S.CodStud=P.CodStud and F.CodFac=S.CodFac
order by Denumire;
```

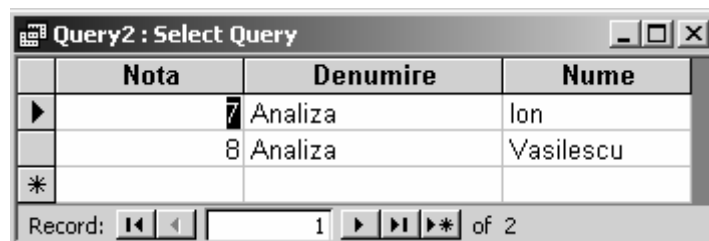
Denumire	Nume	Prenume	Grupa	An	Media
Automatica	Bucur	Valeriu	314AA	1	9.13
Automatica	Vasile	Alexandru	344AB	4	9.25
Automatica	Ion	Adrian	312AA	1	8.12
Electrotehnica	Vasilescu	Rodica	313BB	1	7.34
Energetica	Mihai	Dumitru	1212AC	5	7.77

Record: 1 of 5

`F`, `S`, `P` sunt aliașii tabelelor (nu este obligatorie folosirea cuvântului cheie `as`). Folosirea numelui de tabelă sau a aliasului ca prefix pentru un nume de coloană este obligatorie doar în cazul în care pot apărea confuzii (coloane cu același nume în mai multe tabele), dar nu va fi semnalată eroare dacă se folosesc și pentru celelalte coloane.

3. Afișarea numelui studenților care au nota la materia cu numele Analiză.

```
select Nume, Denumire, Nota
from Studenti S, Materii M, Note N
where S.CodStud=N.CodStud and M.CodMaterie=N.CodMaterie
and M.Denumire='Analiza';
```

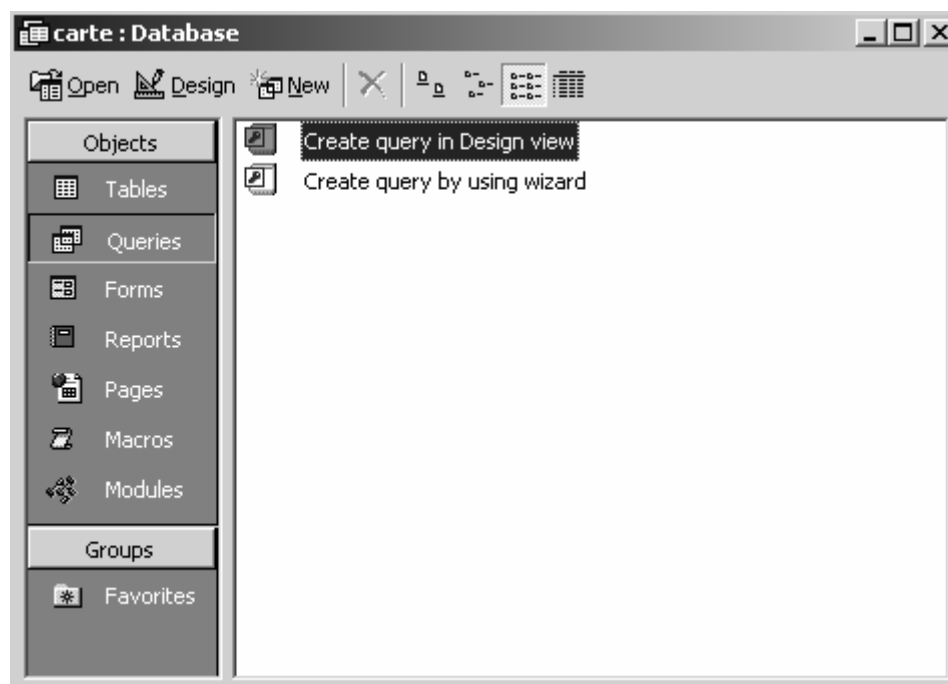


Nota	Denumire	Nume
7	Analiza	Ion
8	Analiza	Vasilescu

Pentru acest exemplu vom utiliza și modul grafic care va genera rezultatul de mai sus.

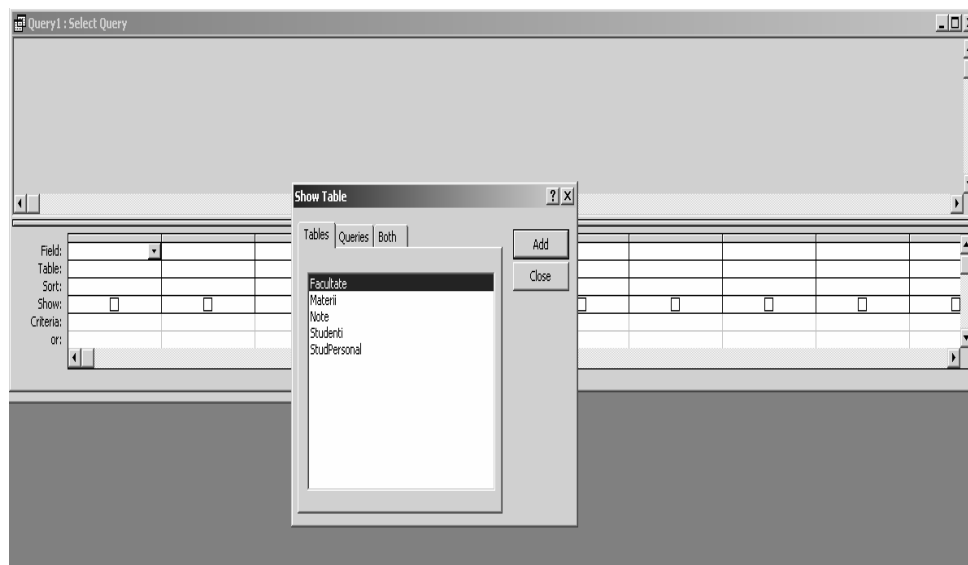
Pasul 1:

Ne poziționăm pe obiectul Queries și se va deschide următoarea fereastră.



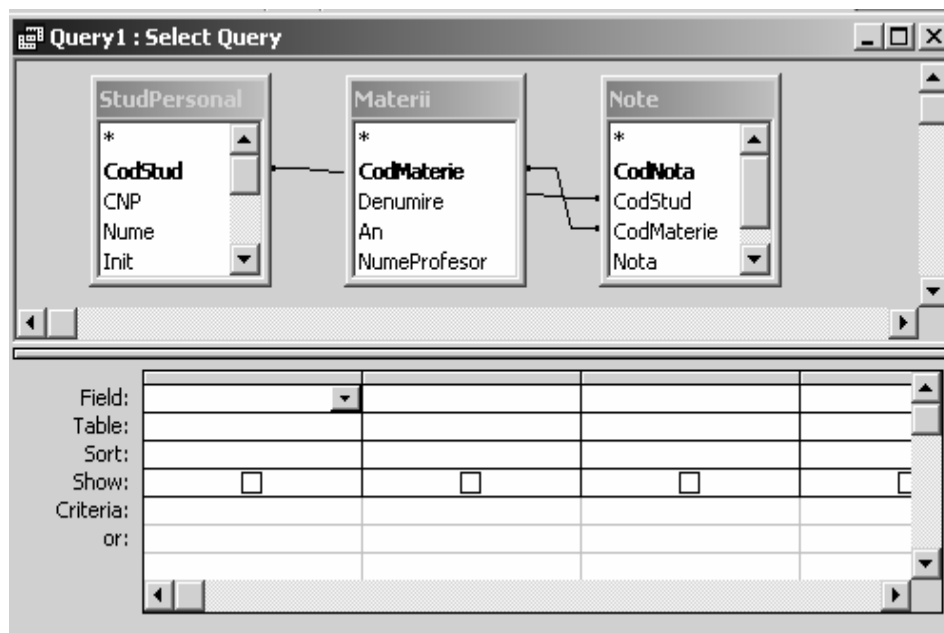
Pasul 2:

Apăsăm dublu clic pe opțiunea Create query in Design view și se va deschide fereastra care conține toate tabelele din baza noastră de date:



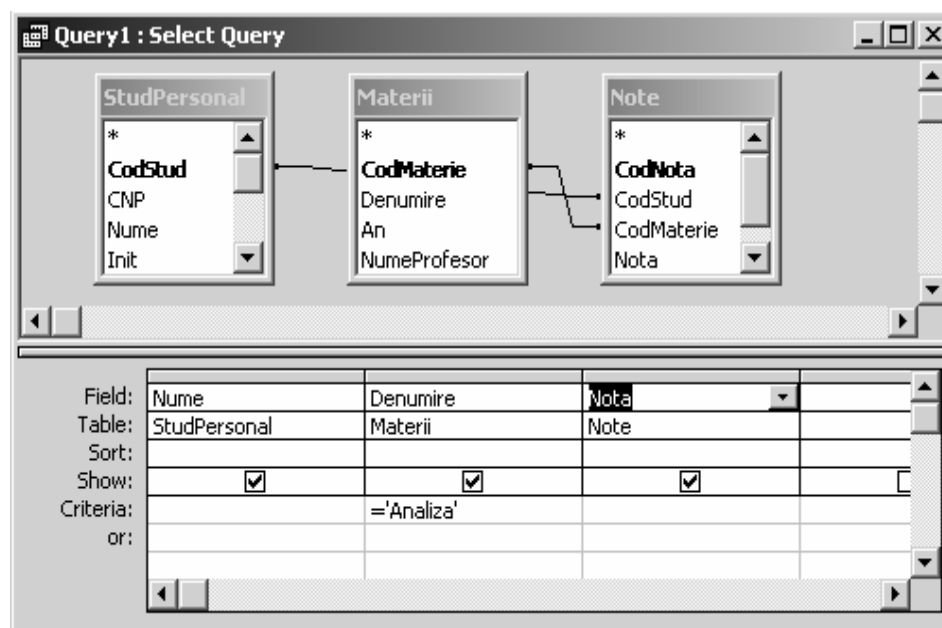
Pasul 3:

Alegem tabelele care ne sunt necesare pentru interogare. În cazul nostru StudPersonal, Materii și Note, apăsând pe rând butonul Add.



Pasul 4:

Se alege pentru fiecare tabelă câmpurile care trebuie afișate și se setează un criteriu dacă acesta există.



Pasul 5:

Se apasă ! (Run) și se va afișa rezultatul frazei SQL.

	Nume	Denumire	Nota
▶	on	Analiza	7
	Vasilescu	Analiza	8
*			

Record: 1 of 2



Lucrarea de verificare a cunoștințelor 1

1. Creați baza de date Universitate.
2. Creați baza de date Companie

Lucrarea de verificare a cunoștințelor 2

1. Creați tabelele din baza de date Universitate în modul grafic.
2. Creați tabelele din baza de date Companie. Structura bazei de date Companie este cea proiectată în capitolul anterior.

Lucrarea de verificare a cunoștințelor 3

1. Creați în baza de date Universitate o tabelă suplimentară numită Test cu un câmp numit testare cu tipul de date text (în modul grafic).
2. Modificați structura tabelii Test adăugând un câmp numit Nou cu tipul de date asociat Integer (în modul grafic).
3. Ștergeți tabela Test din baza de date Universitate (în modul grafic).

Lucrarea de verificare a cunoștințelor 4

Introduceți date în tabelele din baza de date Companie, folosind modul grafic, dar și modul SQL view.

Lucrarea de verificare a cunoștințelor 5

Realizați în modul SQL View și în modul grafic următoarele interogări:

1. Ștergeți din baza de date Universitate materia care este predată de profesorul M.Olteanu.
2. Ștergeți din baza de date Universitate toți studenții care s-au născut în orașul Constanța.
3. Ștergeți dintr-un tabel al bazei de date Companie, 2 înregistrări, după un criteriu ales de Dvs.

Lucrarea de verificare a cunoștințelor 6

Realizați în modul SQL View și în modul grafic următoarele interogări:

1. Modificați în baza de date Universitate Adresa la facultăți cu denumirea "Splaiul Independenței".
2. Modificați anul de studiu al tuturor studenților care au media

>5, aceștia fiind considerați promovați în anul de studiu curent.
3. Modificați un tabel al bazei de date Companie, după un criteriu ales de Dvs.

Lucrarea de verificare a cunoștințelor 7

Realizați în modul SQL View și în modul grafic următoarea interogare:

1. Afișați toate datele stocate în tabela StudPers.

Lucrarea de verificare a cunoștințelor 8

Realizați în modul SQL View și în modul grafic următoarele interogări:

1. Afișați notele și data efectuării testării pentru toate înregistrările din tabela Note.
2. Afișați numele facultății și adresa din tabela Facultati folosind aliașii de coloană 'Nume facultate' și 'Adresă facultate'.
3. Afișați câmpurile nume facultate, nume decan și între acestea textul 'are decanul' din tabela Facultate.

Lucrarea de verificare a cunoștințelor 9

Realizați în modul SQL View și în modul grafic următoarele interogări:

1. Afișați notele distincte care există salvate în baza de date.
2. Afișați în ordine ascendentă în funcție de data nașterii toți studenții din tabelul StudPers.
3. Afișați în ordine descendentă toate înregistrările din tabela Note, ordonați după ID-ul facultății.
4. Afișați toate materiile din anul 1.
5. Afișați toți studenții care sunt născuți în București sau în județul Teleorman.
6. Afișați toate grupele cu studenții care au media între 7 și 9 sau aparțin de facultățile care au codul facultății 1 sau 2.
7. Afișați toate numele studenților care începe cu A și prenumele cu I.

Lucrarea de verificare a cunoștințelor 10

Realizați în modul SQL View și în modul grafic următoarele interogări:

1. Afișați numărul de facultăți din universitate.
2. Afișați care este cea mai mică și cea mai mare notă obținută de studenții de la facultatea cu id-ul 3.
3. Afișați nota obținută de studentul cu numele Ionescu la materia Analiză.
4. Afișați grupele care există în fiecare facultate grupate pe ani.

Răspunsuri și comentarii la întrebările din testele de autoevaluare

Întrebarea 2.



Operatorii din algebra relațională sunt AND, OR, XOR, =, \neq , IS NULL, IS NOT NULL. În limbajul standard SQL mai există și sunt folosiți și alți operatori ca: BETWEEN, IN etc.

Întrebarea 3.

Termenul SQL reprezintă o prescurtare a Structured Query Language. El a fost utilizat inițial de către firma IBM pentru produsul DB2, limbajul de interogare al bazelor de date relaționale SQL a devenit la mijlocul deceniului trecut un standard în domeniu.

Întrebarea 4.

Operații de bază care se pot efectua în cadrul unei baze de date sunt:

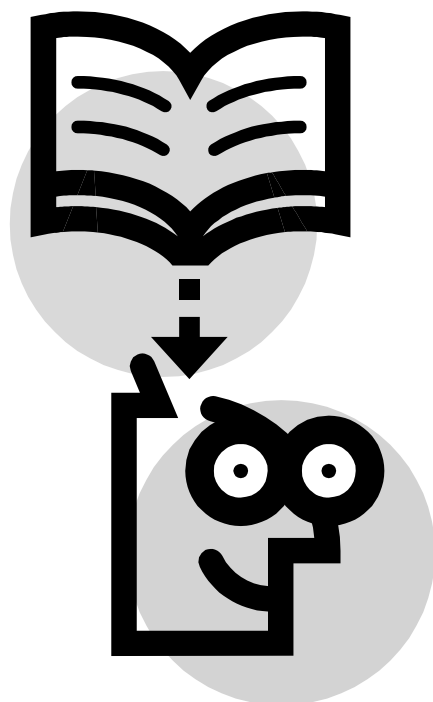
- Crearea/ștergerea unei tabele
- Inserarea de noi linii într-o tabelă
- Ștergerea unor linii dintr-o tabelă
- Modificarea unor linii dintr-o tabelă
- Listarea selectivă a datelor din una sau mai multe tabele

Indicații la problemele propuse

Problemele propuse în lucrările de verificare se fac după modelele de exemple prezentate în unitatea de învățare sau după tipicul acestora.

Bibliografie:

- Cârstoiu, Dorin, *Baze de date relaționale*, Editura Printech, 1999
- Rădulescu, Florin, *Baze de date în Internet*, Editura Printech, 2000
- Ionescu, Felicia, *Baze de date relaționale și aplicații*, Editura Tehnică, 2004
- Baltac, Vasile, *ECDL-Excel, Access, PowerPoint în 20 lecții și 75 de simulări*, Editura Andreco, 2003
- Browne, Allen, Balter Alison, *Bazele Access 95*, Editura Teora, 1999
- Pribeanu, Costin, *Baze de date și aplicații*, Editura MatrixRom, 2000
- Pascu, C., Pascu A., *Totul despre SQL*, Editura Tehnică, 1994



Unitatea de învățare Nr. 5

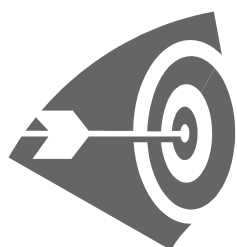
CONSTRUIREA INTERFEȚELOR CU AJUTORUL FORMULARELOR ÎN MICROSOFT ACCESS

Cuprins	Pagina
Obiectivele unității de învățare nr. 5	104
5.1. Ce este un formular?	105
5.2. Lucrul cu formularele	105
5.2.1. Deschiderea unui formular	105
5.2.2. Crearea unui formular	105
5.2.4. Utilizarea unui formular	109
pentru a introduce și a modifica date în tabelă	
5.2.4. Parcurgerea înregistrărilor utilizând formularele	110
5.2.5. Adăugarea și modificarea textului în antet și subsol	110
5.2.6. Ștergerea unui formular	112
5.3. Salvarea și închiderea unui formular	112
Lucrare de verificare a cunoștințelor	113
Răspunsuri și comentarii la întrebările din testele de autoevaluare	114
Bibliografie	114

Încă de la început doresc să vă felicit pentru parcurgerea cu succes a primelor patru unități de învățare și să vă urez bun venit la studiul acestei noi unități de învățare. În primele patru unități de învățare am realizat o scurtă introducere în acest amplu domeniu al bazelor de date, am descris mai în detaliu ce este un sistem de baze de date și în ce constă el, am studiat și cum se crează, analizează și se proiectează corect o bază de date, dar am studiat și cum se gestionează datele stocate în baza de date.

În această unitate de învățare nr. 5 vom studia cum se crează, se manipulează și se șterge un formular pentru introducerea datelor în baza de date.

OBIECTIVELE unității de învățare nr. 5



Principalele obiective ale unității de învățare nr. 5 sunt:

După studiul unității de învățare nr. 5 vei fi capabil să demonstrezi că ai dobândit cunoștințe suficiente pentru a înțelege:

- ce este un formular
- la ce se folosesc formularele din Access
- cum se creează un formular pentru o tabelă a bazei de date Access
- cum se utilizează un formular pentru introducerea datelor într-o tabelă și pentru modificarea datelor într-o tabelă
- cum se salvează un formular
- cum se șterge un formular

5.1. Ce este un formular?

Formularele (Forms) reprezintă ferestrele primare folosite pentru introducerea și afișarea datelor în Access. Formularele vă permit să prezentați datele într-o formă care îl scutește pe utilizator de preocupările legate de modul de stocare al acestora. Se pot crea formulare diferite pentru utilizări diferite: introducerea unor înregistrări noi, editarea celor existente, numai pentru afișare sau formulare care funcționează pur și simplu ca niște casete de dialog.



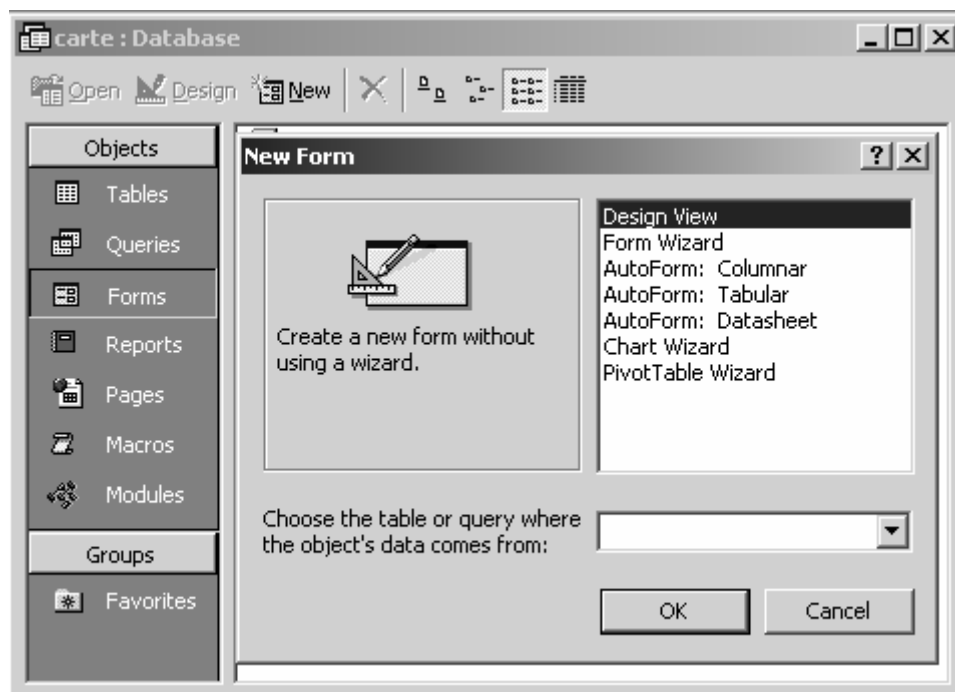
5.2. Lucrul cu formularele

5.2.1. Deschiderea unui formular

Dacă există deja creat un formular, acesta se poate deschide prin apăsarea dublu clic pe respectivul formular din obiectul Forms. Toate formularele create într-o bază de date se vor găsi în obiectul Forms.

5.2.2. Crearea unui formular

Pentru a crea un formular nou vom alege obiectul Forms din fereastra Database și se apelează meniul Insert cu opțiunea Forms. Va apărea o fereastră cu mai multe tipuri de programe wizard pentru formulare:



- *Design View* (modul proiectare) permite crearea tuturor elementelor manual
- *Form Wizard* oferă posibilitatea controlării fiecărei etape a procesului

- *AutoForm: Columnar* (Configurare automată a formularului: așezare în coloană) crează un formular cu controalele aliniate unul sub celălalt (configurație corespunzătoare pentru un formular principal)
- *AutoForm: Tabular* (Configurare automată a formularului: așezare tabelară) crează un formular cu controalele aliniate unul lângă celălalt (configurație corespunzătoare pentru un subformular)
- *AutoForm: DataSheet* (Configurare automată a formularului: modul DataSheet) crează un formular având modul de afișare prestabilit DataSheet (configurație corespunzătoare pentru afișarea numărului maxim posibil de înregistrări deodată)
- *Chart Wizard* (program pentru reprezentări grafice) crează un grafic. Pentru a obține un grafic tipărit, se folosește un raport în loc de formular
- *PivotTable Wizard* (program pentru tabele pivot) crează un formular pentru afișarea datelor din Excel.

Sub lista programelor wizard se află o casetă cu listă derulantă în care trebuie să selectați tabelul sau interogarea care va servi ca sursă de date pentru formular. În cazul unui formular casetă de dialog, caseta pentru precizarea sursei de date poate fi lăsată necompletată.



Exemplu:

Vom exemplifica crearea unui formular nou pentru introducerea datelor în tabela Facultate, folosind opțiunea **Create by using wizard**, urmând pașii necesari pentru a crea formularul dorit.

Pas 1:

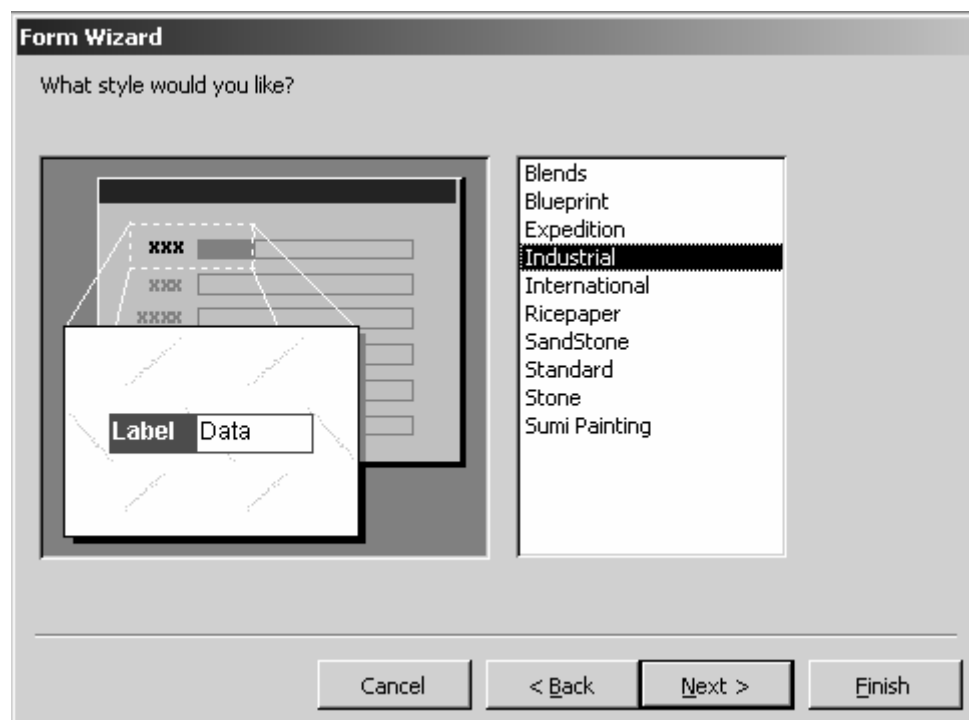
Se alege tabela căreia doriți să îi creați formularul și se apasă butonul **Next**.

Pas 2:

În această etapă se alege modul în care să fie afișate atributele în formular și se apasă **Next**

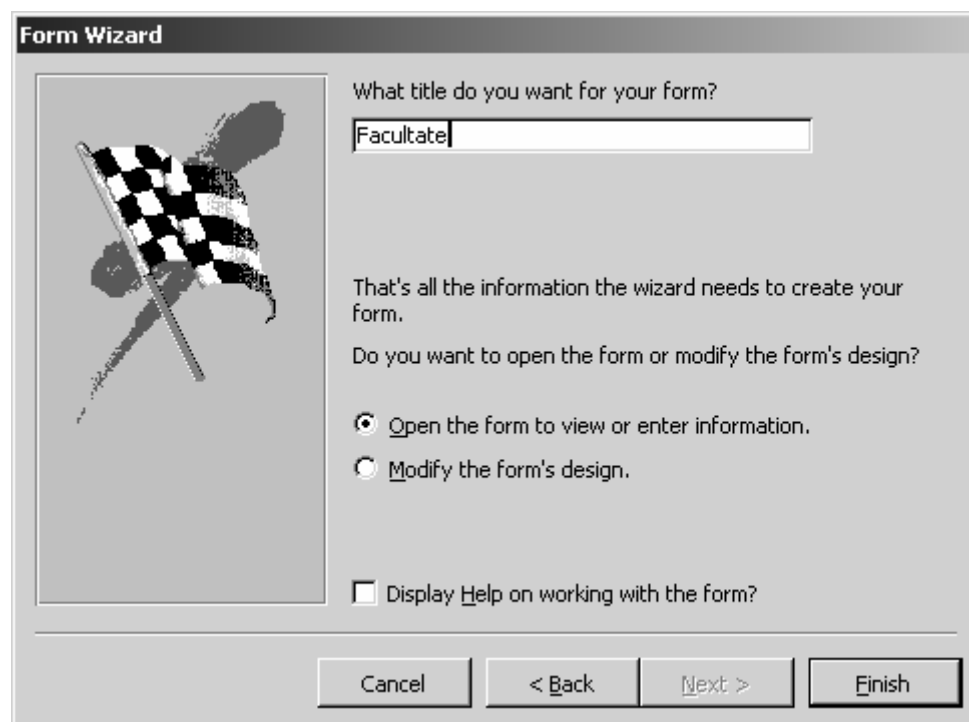
Pas 3:

În această etapă se pot alege din stilurile predefinite, stilul pe care îl doriți să îl conțină formularul.



Pas 4:

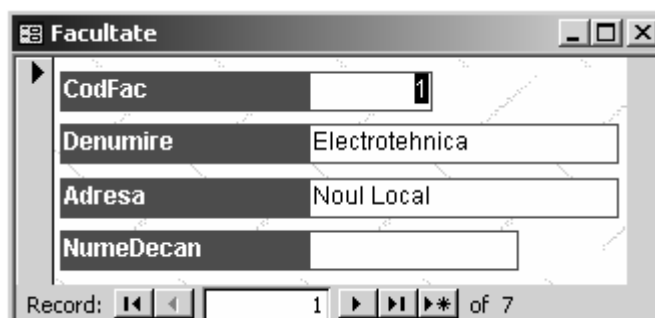
Aceasta este etapa finală, în care vă alageți numele dorit pentru formular și dacă doriți să introduceți date în tabelă cu ajutorul formularului.




Pentru a salva un formular se face opțiunea Save din meniul File.

5.2.3. Utilizarea unui formular pentru a introduce și a modifica date în tabelă

Având deschisă forma, puteți introduce și modifica date în tabela corespunzătoare formularului.



Pentru a introduce date în tabelă cu ajutorul formularului trebuie apăsat butonul . Acesta ne va poziționa pe o nouă înregistrare și vom putea introduce noile date. Ne vom poziționa cu cursorul în câmpurile de editare și vom introduce datele corespunzătoare noii înregistrări.

Pentru a modifica date în tabelă cu ajutorul formularului trebuie să ne poziționăm pe înregistrarea dorită cu ajutorul mouse-ului și apoi se modifică datele dorite.

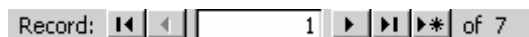
Exemplu:

De exemplu, dorim ca la înregistrarea de mai sus să completăm numele decanului. Pentru aceasta ne poziționăm pe respectiva înregistrare și apoi în dreptul atributului NumeDecan introducem numele dorit. După realizarea modificărilor, prin închiderea formularului, modificările vor fi actualizate în tabelă.



5.2.4. Parcurgerea înregistrărilor utilizând formularele

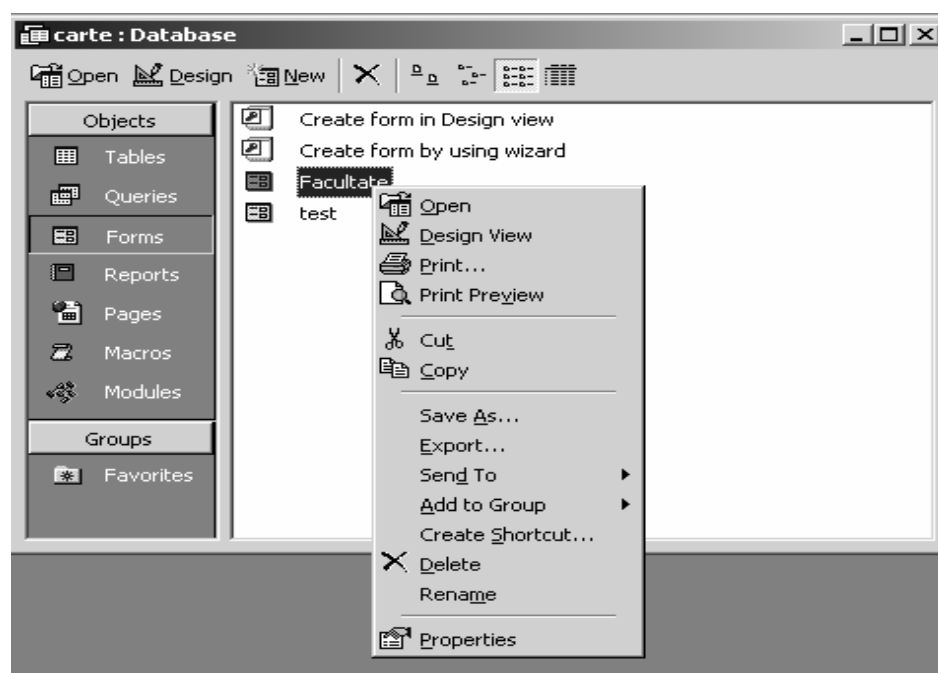
Cu ajutorul formularelor putem parcurge toate înregistrările unei table. Acest lucru îl putem realiza cu ajutorul butoanelor existente în partea de jos a formularului.



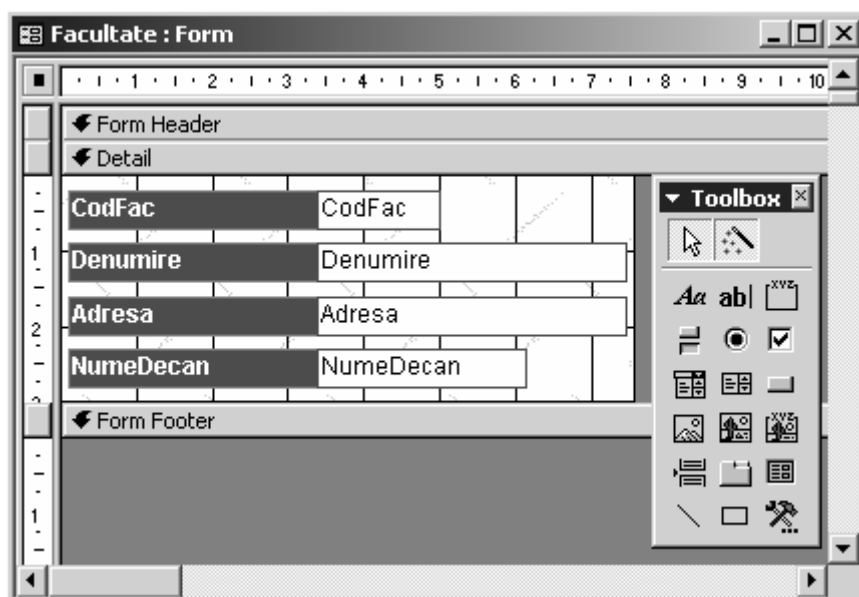
- Butonul se folosește pentru poziționarea pe prima înregistrare a tablei
- Butonul se folosește pentru reîntoarcerea la înregistrarea anterioară
- Butonul se folosește pentru trecerea la înregistrarea următoare
- Butonul se folosește pentru poziționarea pe ultima înregistrare a tablei
- Butonul se folosește pentru a adăuga o înregistrare nouă.

5.2.5. Adăugarea și modificarea textului în antet și subsol

Pentru a putea face modificări în cadrul unui formular, mai exact dacă dorim adăugarea unui text sau a unei imagini în antetul sau subsolului unui formular este necesar să deschidem formularul în modul Design al formularului apăsând butonul sau făcând opțiunea Design View, care apare la apăsarea clic dreapta pe respectivul formular.



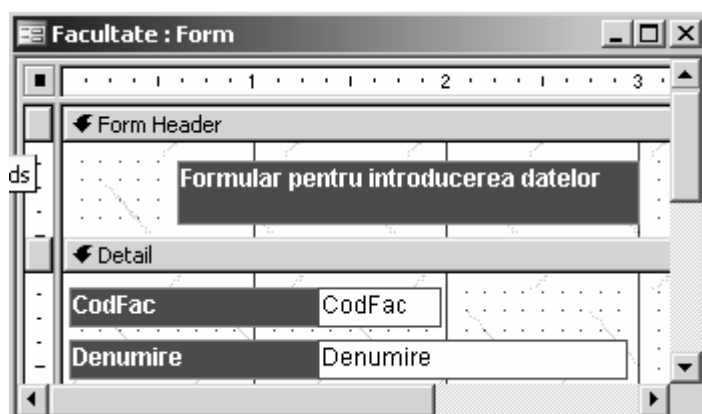
Pe ecran va apărea constructorul de formulare în care puteți modifica ceea ce doriți. Va apărea și o bară de instrumente corespunzătoare constructorului de formulare, prin intermediul căreia putem realiza diferite modificări în formular.



Exemplu:



De exemplu, dorim să scriem în antet “Formular pentru introducerea datelor”. Pentru asta va trebui să facem loc în partea antetului căsuței de text, trăgând efectiv cu mouse-ul antetul și apoi în spațiul creat introducem textul dorit. Pentru a introduce un text trebuie apăsat butonul **Aa** de pe bara de instrumente. După apăsarea acestui buton selectați locul unde dorim poziționarea textului și apoi vom introduce textul.



După introducerea textului se salvează forma pentru a vedea modificările făcute asupra ei.

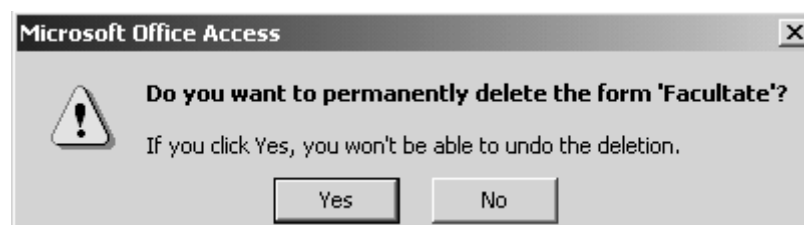
Forma finală a formularului este:

Asemănător se poate modifica și subsolul formularului, sau putem introduce imagini în formulare folosind comanda Insert-Picture.

5.2.6. Ștergerea unui formular

Putem șterge un formular prin mai multe metode. Primul pas pentru orice metodă este selectarea formularului dorit pentru ștergere și apoi fie apăsăm butonul Delete din tastatură, fie apăsând butonul existent în parte de sus a ferestrei, fie apăsând clic dreapta și făcând opțiunea Delete.

Pe ecran va apărea o casetă de dialog în care se cere confirmarea ștergerii formularului.



Apăsând butonul Yes se confirmă ștergerea formularului, iar apăsând No se renunță la operația de ștergere.

5.3. Salvarea și închiderea unui formular

Pentru salvarea formularului sau a oricărei modificări făcute asupra lui, se apasă butonul Save existent pe bara de instrumente a aplicației, sau făcând opțiunea Save din meniu.

Pentru închiderea formularului sau se apasă butonul din partea dreaptă a ecranului sau făcând opțiunea Close din meniu.



Lucrare de verificare a cunoștințelor

1. Creați un formular pentru introducerea datelor personale despre studenți pentru baza de date Universitate cu titlul „Date personale studenți” și salvați-l cu numele Personal.
2. Introduceți două înregistrări în tabela StudPersonal cu ajutorul formularului creat la exercițiul 1.
3. Modificați o înregistrare care deja există în tabela StudPersonal cu ajutorul formularului creat la exercițiul 1 și apoi închideți formularul.



Teste de autoevaluare

1. Ce este un formular?
2. Care sunt operațiile care se pot realiza asupra unor formulare?
3. Descrieți tipurile de programe wizard disponibile pentru crearea de formulare.

↑

Răspunsuri și comentarii la întrebările din testele de autoevaluare



Întrebarea 1.

Formularele (Forms) reprezintă ferestrele primare folosite pentru introducerea și afișarea datelor în Access.

Întrebarea 2.

Operațiile care se pot efectua asupra unor formulare sunt de creare, de modificare, de salvare și de ștergere.

Întrebarea 3.

Răspunsul la această întrebare este mai amplu și este descris cu amănuntul la începutul acestui capitol.

Indicații la problemele propuse

Problemele propuse sunt așezate după un exemplu. Fiecare problemă propusă se face după modelul de exemplu prezentat înaintea ei sau după tipicul acesteia.

Bibliografie:

- Cârstoiu, Dorin, *Baze de date relaționale*, Editura Printech, 1999
- Rădulescu, Florin, *Baze de date în Internet*, Editura Printech, 2000
- Ionescu, Felicia, *Baze de date relaționale și aplicații*, Editura Tehnică, 2004
- Baltac, Vasile, *ECDL-Excel, Access, PowerPoint în 20 lecții și 75 de simulări*, Editura Andreco, 2003
- Browne, Allen, Balter Alison, *Bazele Access 95*, Editura Teora, 1999
- Pribeanu, Costin, *Baze de date și aplicații*, Editura MatrixRom, 2000
- Pascu, C., Pascu A., *Totul despre SQL*, Editura Tehnică, 1994

Unitatea de învățare Nr. 6

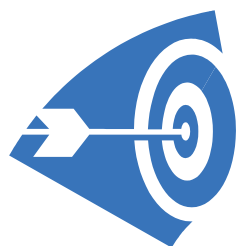
RAPOARTE ÎN MICROSOFT ACCESS

Cuprins	Pagina
Obiectivele unității de învățare nr. 6	116
6.1. Ce este un raport?	117
6.2. Lucrul cu rapoarte	117
6.2.1. Deschiderea unui raport	117
6.2.2. Crearea unui raport	117
6.2.3. Adăugarea și modificarea textului în antet și subsol	123
6.2.5. Ștergerea unui raport	125
6.3. Salvarea și închiderea unui raport	125
Lucrare de verificare a cunoștințelor	125
Răspunsuri și comentarii la întrebările din testele de autoevaluare	127
Bibliografie	127

Încă de la început doresc să vă felicit pentru parcurgerea cu succes a primelor cinci unități de învățare și să vă urez bun venit la studiul acestei noi unități de învățare. În primele cinci unități de învățare am realizat o scurtă introducere în acest amplu domeniu al bazelor de date, am descris mai în detaliu ce este un sistem de baze de date și în ce constă el, am învățat și cum se crează, analizează și se proiectează corect o bază de date, dar am studiat și cum se gestionează datele stocate în baza de date. În unitatea de învățare anterioară am învățat cum se crează, se manipulează și se șterge un formular pentru introducerea datelor în baza de date.

În această unitate de învățare vom studia ce sunt rapoartele, cum se crează și cum se manipulează ele.

OBIECTIVELE unității de învățare nr. 6



Principalele obiective ale unității de învățare nr. 6 sunt:

După studiul unității de învățare nr. 6 vei fi capabil să demonstrezi că ai dobândit cunoștințe suficiente pentru a înțelege:

- la ce se folosesc rapoartele din Access
- cum se creează un raport de date stocate în Access
- cum se salvează un raport
- cum se șterge un raport

6.1. Ce este un raport?

În timp ce formularul este proiectat pentru lucrul pe ecran (deși și el poate fi tipărit), raportul este proiectat în primul rând pentru tipărire (cu toate că și el poate fi afișat pe ecran).

Pe lângă această deosebire de destinație, există și o diferență conceptuală majoră între formular și raport. Formularul este proiectat pentru accesul aleator la date. După ce obține accesul la o anumită înregistrare, utilizatorul poate trece la sfârșitul formularului pentru a adăuga o înregistrare nouă sau se poziționează la începutul formularului pentru căutarea unei alte înregistrări. În concluzie, formularul se bazează pe un set dinamic de date, date care trebuie actualizate cu toate modificările efectuate de alți utilizatori în timpul utilizării formularului.

Pe de altă parte, raportul nu modifică niciodată datele, fiind necesară parcurgerea secvențială a înregistrărilor pentru a genera subtotaluri și rezumate. În locul unui set dinamic de date, Microsoft Access folosește o copie protejată la scriere a datelor care este citită în secvență, numită snapshot.


Un raport trebuie creat pentru orice intenționăm să tipărim cu regularitate, fie că este vorba de un listing simplu, o listă de etichete poștale, o colecție de grafice sau un rezumat sau o analiză financiară complexă. Raportul ne oferă de asemenea posibilitatea prelucrării datelor în scopul obținerii unor rezultate sintetice: totaluri, subtotaluri etc. Datele pot fi grupate pe un număr de până la zece niveluri diferite, fiecare cu propriile sale informații sintetice.

6.2. Lucrul cu rapoarte

6.2.1. Deschiderea unui raport

Dacă există deja creat un raport, acesta se poate deschide prin apăsarea dublu clic pe respectivul raport din obiectul Reports. Toate rapoartele create într-o bază de date se vor găsi în obiectul Reports.

6.2.2. Crearea unui raport





Pentru a crea un raport nou vom alege obiectul Reports din fereastra Database și se apasă dublu clic pe opțiunea  Create report by using wizard pentru crearea de rapoarte cu ajutorul Wizard-ului. Va trebui să parcurgem pașii necesari pentru a crea raportul dorit.

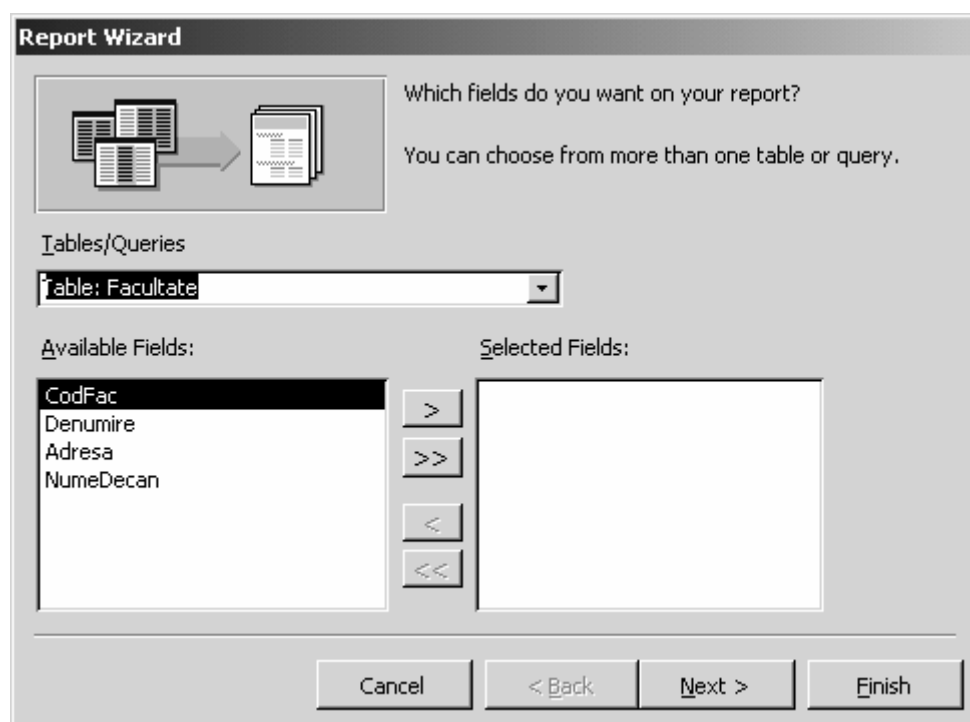


Exemplu:

Vom exemplifica crearea unui raport nou pentru afișarea datelor în tabela Facultate, folosind opțiunea **Create report by using wizard**, urmând pașii necesari pentru a crea raportul dorit.

Pas 1:

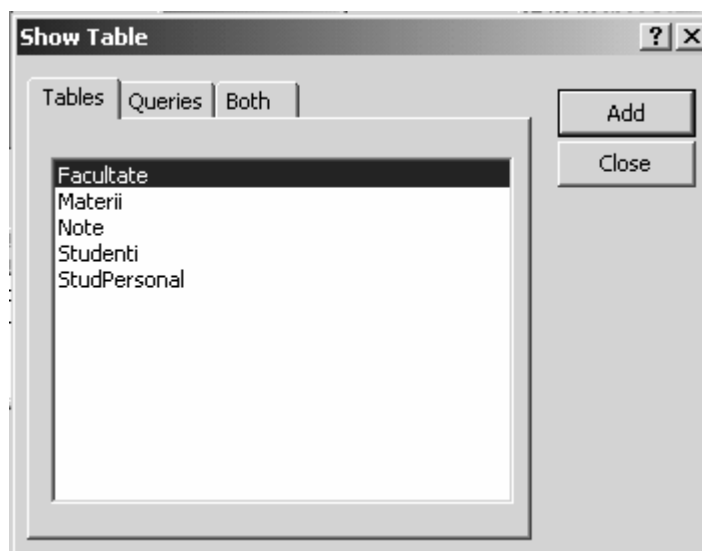
Se alege tabela (sau tabelele) care conține câmpurile care trebuie să fie conținute în raport și se apasă Next. Mai exact, putem crea un raport care să conțină numai anumite câmpuri dintr-o tabelă și alte câmpuri din alte tabele, acest lucru făcându-se alegând tabela corespunzătoare și selectând câmpurile dorite. Selectarea câmpurilor dintr-o tabelă se face astfel: se poziționează pe respectivul câmp și se apasă butonul . Dacă dorim ca toate câmpurile dintr-o tabelă să apară în raport se apasă butonul . Pentru deselectarea câmpurilor ne poziționăm pe câmpul respectiv și se apasă butonul , iar pentru deselectarea tuturor câmpurilor se apasă butonul .



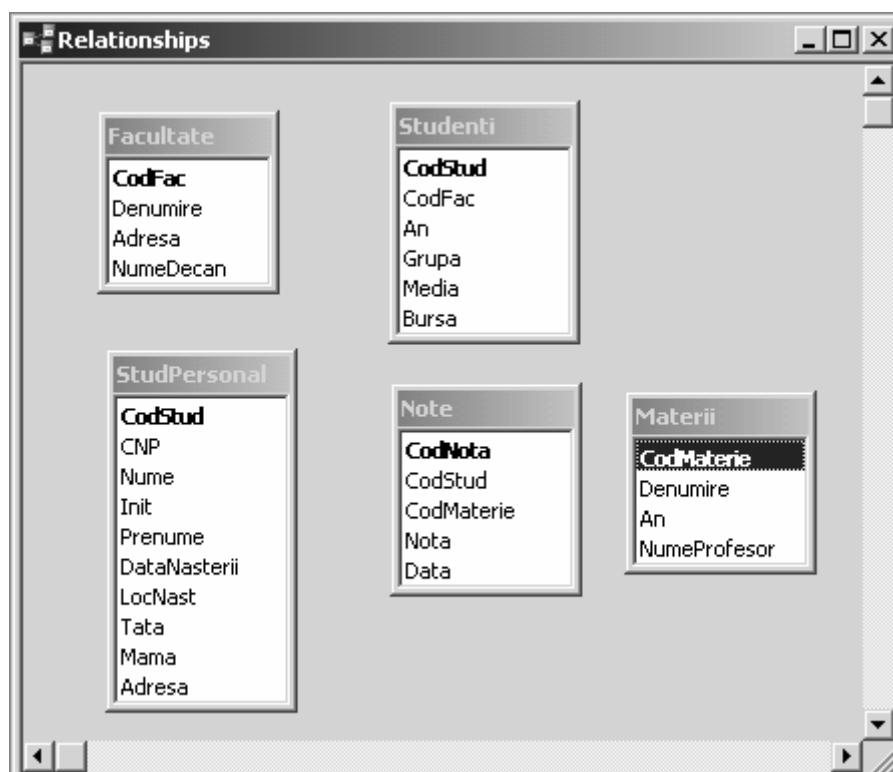
Observație: Pentru a putea genera rapoarte cu date din mai multe tabele, trebuie înainte create relațiile între tabele.

Pentru a crea aceste relații dintre tabele se face opțiunea Tools din meniu și se alege Relations. Va apărea o fereastră pentru alegerea tabelelor între care dorim crearea de relații.

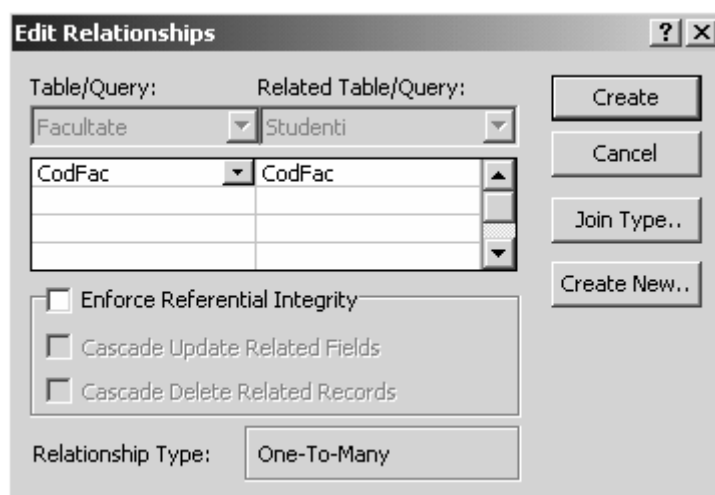




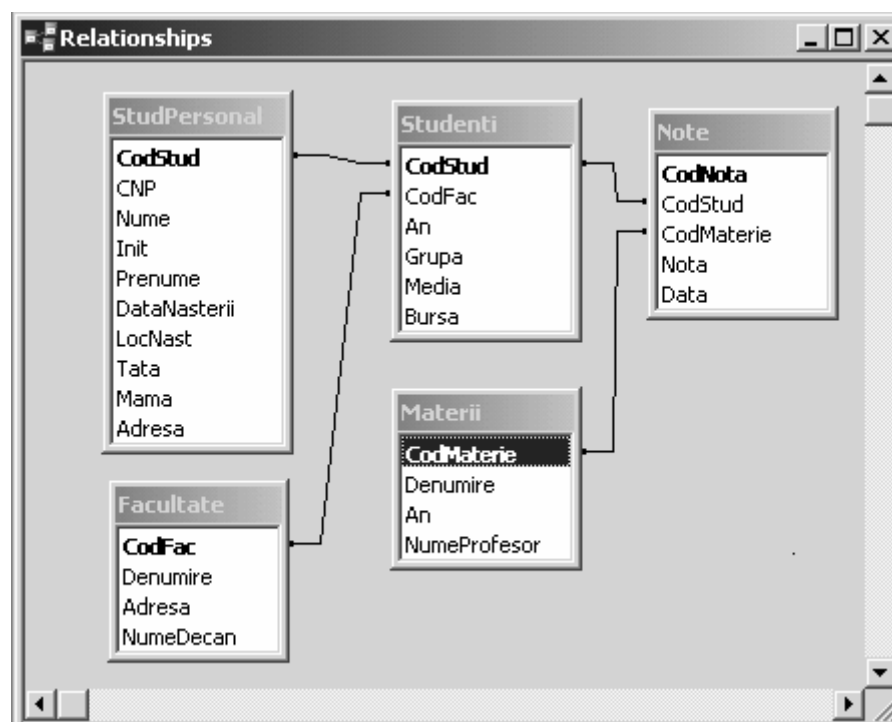
Pentru a crea relații între toate tabelele ne poziționăm pe fiecare dintre ele și apăsăm butonul Add. Va apărea o fereastră în care sunt afișate toate tabelele între care vom crea relații (pe care le-am ales noi din lista de tabele existente în baza de date).



Pentru a crea efectiv relațiile, se poziționăm pe un câmp al unei tabele și ținem clicul de la mouse apăsat până la câmpul corespunzător relației. Va apărea o fereastră pentru confirmarea creării relației.



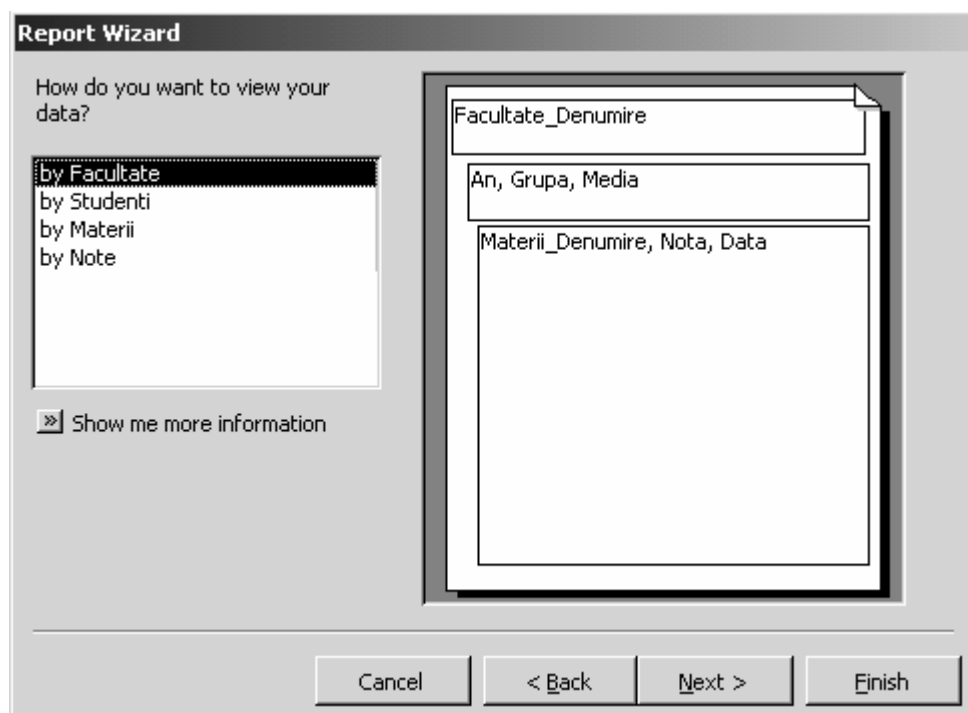
După ce au fost create toate relațiilor, structura bazei de date ca arăta astfel:



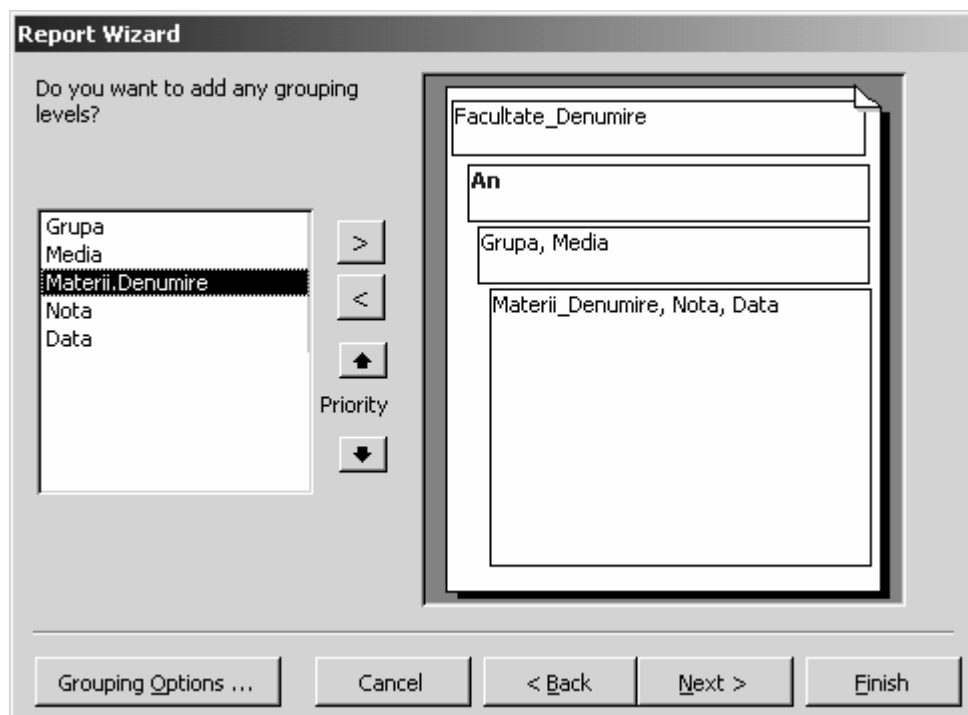
Revenim la crearea raportului despre notele studenților de la fiecare facultate de la fiecare materie.

Pas2:

În această etapă se alege modul în care vor fi afișate atributele în raport și se apasă Next.

**Pas3:**

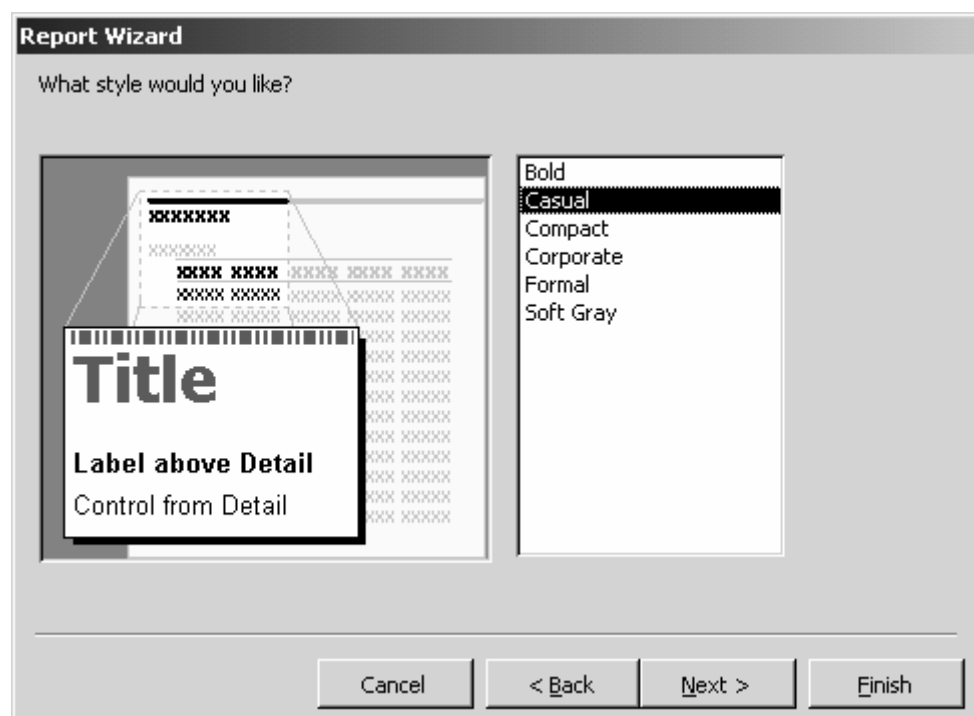
La acest pas se pot grupa rezultatele care vor fi afișate în raport după anumite câmpuri folosind săgețile și apoi se apasă Next.



Pasul 4 și **pasul 5** reprezintă alegerea modului de afișare a informațiilor în raport.

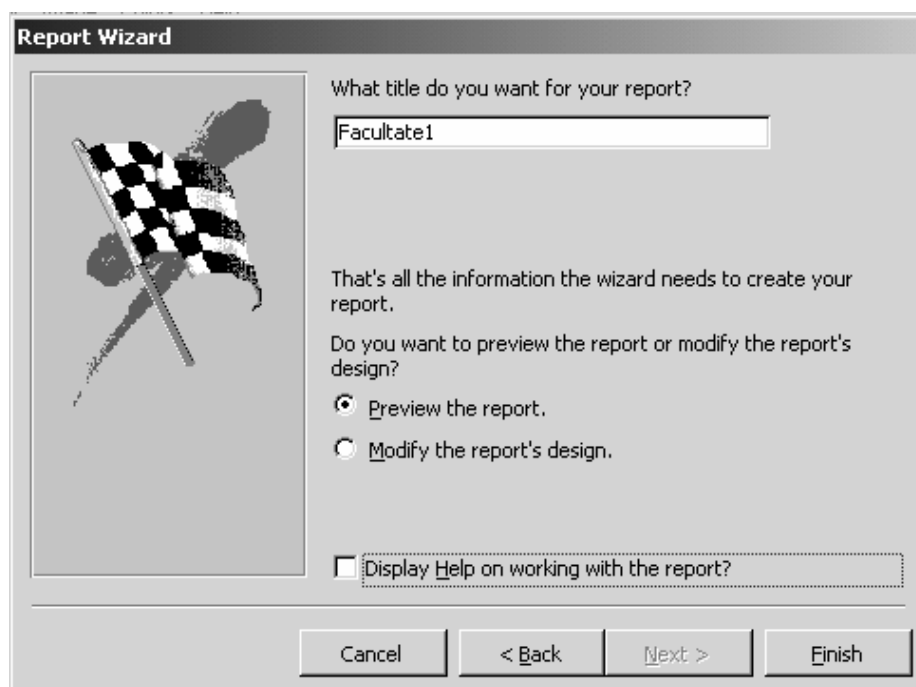
Pasul 6:

Se alege dintr-o listă predefinite de stiluri un mod de afișare a raportului.



Pasul 7:

Se denumește raportul cu un nume ales de utilizator și se apasă Next.




Pasul 8:

Se apasă butonul Finish.

Vă vom prezenta mai jos un raport creat.

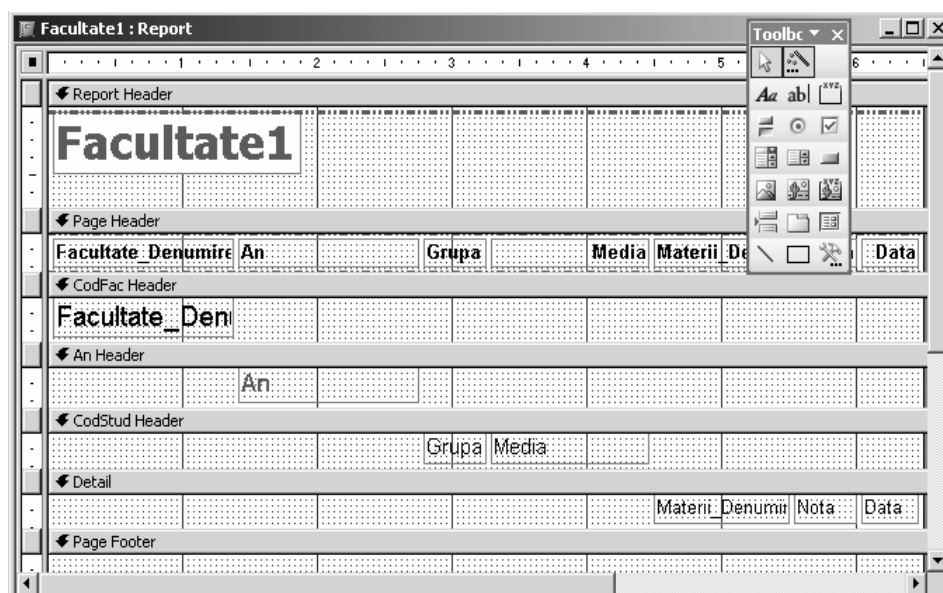
Proiecte_Personale	Data	Facultate_Personale	Nota
1	11.10	7.10	8.1000
		8.1000	2.1000
1	11.10	8.10	8.1000
		8.1000	8.1000
1	11.10	9.10	8.1000
		8.1000	8.1000

6.2.3. Adăugarea și modificarea textului în antet și subsol

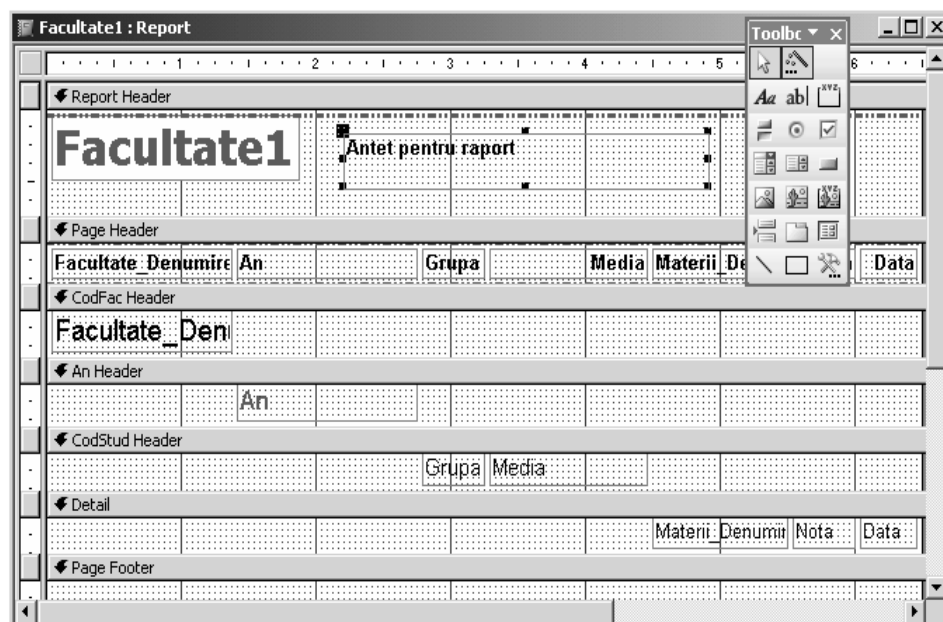
Pentru a adăuga un text sau o imagine în antetul sau subsolului unui raport este necesar să deschidem raportul în modul Design al raportului apăsând butonul  Design sau făcând opțiunea Design View, care apare la apăsarea clic dreapta pe respectivul formular.

Pe ecran va apărea constructorul de rapoarte în care putem modifica ceea ce dorim.

Pe ecran va apărea și o bară de instrumente corespunzătoare constructorului de rapoarte, prin intermediul căreia putem realiza diferite modificări în raport..




De exemplu, dorim să scriem în antet “Antet pentru raport “. Pentru asta va trebui să facem loc în partea antetului căsuței de text, tragând efectiv cu mouse-ul antetul și apoi în spațiul creat introducem textul dorit. Pentru a introduce un text trebuie apăsă butonul **Aa** de pe bara de instrumente. După apăsarea acestui buton selectați locul unde dorim poziționarea textului și apoi vom introduce textul.



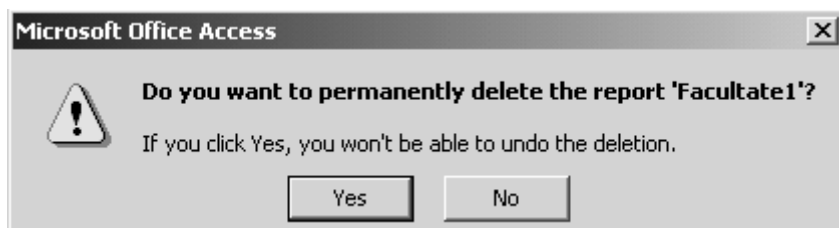
După introducerea textului se salvează raportul și pentru a vedea modificările făcute asupra ei deschidem raportul apăsând butonul Open.

Asemănător se poate modifica și subsolul raportului, sau putem introduce imagini în rapoarte cu ajutorul comenzii Insert-Picture.

6.2.4. Ștergerea unui raport


Putem șterge un raport prin mai multe metode. Primul pas pentru orice metodă este selectarea raportului dorit pentru ștergere și apoi fie apăsăm butonul Delete din tastatură, fie apăsând butonul  existent în parte de sus a ferestrei, fie apăsând clic dreapta și făcând opțiunea Delete.


Pe ecran va apărea o casetă de dialog în care se cere confirmarea ștergerii raportului.



Apăsând butonul Yes se confirmă ștergerea raportului, iar apăsând No se renunță la operația de ștergere.

6.3. Salvarea și închiderea unui raport

Pentru salvarea raportului sau a oricărei modificări făcute asupra lui, se apasă butonul Save  existent pe bara de instrumente a aplicației, sau făcând opțiunea Save din meniu.

Pentru închiderea raportului sau se apasă butonul  din partea dreaptă a ecranului sau făcând opțiunea Close din meniu.



Lucrare de verificare a cunoștințelor

1. Creați un raport cu toate informațiile stocate în baza de date despre facultăți.
2. Creați un raport care să conțină următoarele informații: nume studentului, denumirea materiei la care a susținut examen, nota obținută la acea testare și data testării.
3. Modificați structura raportului de la exercițiul 2, mai exact introduceți în antet titlul: Informații despre promovabilitatea studenților.
4. Explicați dacă se pot tipări rapoartele și explicați cum.



Teste de autoevaluare

1. Care sunt diferențele între un raport și un formular?
2. Se poate crea un raport cu informații din mai multe tabele?
3. Se poate modifica structura unui raport după generarea lui? Explicați.

↑



Răspunsuri și comentarii la întrebările din testele de autoevaluare

Întrebarea 1.

Diferențele dintre un formular și un raport sunt majore: în timp ce formularul este proiectat pentru lucrul pe ecran (deși și el poate fi tipărit), raportul este proiectat în primul rând pentru tipărire (cu toate că și el poate fi afișat pe ecran), iar o diferență importantă ar fi aceea că formularul este proiectat pentru acces la date (deci el poate modifica date), pe când un raport nu modifică date, ci doar le afișează pe ecran.

Întrebarea 2.

Da.

Întrebarea 3.

Da.

Indicații la problemele propuse

Problemele propuse sunt așezate după un exemplu și se fac după modelul de exemplu prezentat înaintea lor.

Bibliografie:

- Cârstoiu, Dorin, *Baze de date relaționale*, Editura Printech, 1999
- Rădulescu, Florin, *Baze de date în Internet*, Editura Printech, 2000
- Ionescu, Felicia, *Baze de date relaționale și aplicații*, Editura Tehnică, 2004
- Baltac, Vasile, *ECDL-Excel, Access, PowerPoint în 20 lecții și 75 de simulări*, Editura Andreco, 2003
- Browne, Allen, Balter Alison, *Bazele Access 95*, Editura Teora, 1999
- Pribeanu, Costin, *Baze de date și aplicații*, Editura MatrixRom, 2000
- Pascu, C., Pascu A., *Totul despre SQL*, Editura Tehnică, 1994

Vă felicităm pentru parcurgerea cu succes a manualului de baze de date și vă dorim baftă în realizarea proiectelor dumneavoastră viitoare în acest amplu domeniu.

