

## Indrumar Baze de Date - Capitolul 2

# LIMBAJUL SQL - PROIECTAREA BAZELOR DE DATE RELATIONALE

Majoritatea sistemelor relationale suporta diferite variante (dialecte) ale limbajului SQL (*Structured Query Language*). Limbajul SQL a fost dezvoltat într-un prototip de sistem relational - System R - la compania IBM la mijlocul anilor 1970. În anul 1979 corporatia Oracle a introdus prima implementare a limbajului SQL în varianta comerciala. În anul 1986 Institutul National American de Standarde (ANSI) a definit standardul limbajului SQL pentru bazele de date relationale. Organizatia Internationala de Standarde (ISO) a adoptat de asemenea SQL ca limbaj standard pentru sistemele relationale, sub denumirea de SQL-92 (sau, mai simplu, SQL2).

### 2.1 LIMBAJUL SQL

Limbajul SQL înglobeaza mai multe componente, dintre care cele mai importante sunt: componenta de descriere a datelor (LDD - Limbaj de Descriere a Datelor) (*Data Description Language - DDL*) si componenta de manipulare a datelor (LMD - Limbaj de Manipulare a Datelor) (*Data Manipulation Language - DML*).

#### 2.1.1 TIPURI DE DATE SQL2

În limbajul SQL (standardul SQL2) sunt predefinite mai multe tipuri de date: numeric, sir de caractere, data (calendaristica), timp, etc. Denumirile tipurilor de date, ca si limitele acestora (valoare minima, valoare maxima) prezinta diferite variatii în functie de implementare (versiunea sistemului SGBD), dar în general sunt destul de asemanatoare. În toate specificatiile de sintaxa, parantezele drepte [...] sunt folosite pentru parametrii optionali ai unei instructiuni.

*Tipul numeric* include numere întregi de diferite dimensiuni (`integer` sau `int` reprezentat pe 4 octeti, `smallint`, reprezentat pe 2 octeti), numere reale reprezentate în virgula flotanta, cu diferite precizii (`float`, reprezentat pe 4 octeti, `real` si `double [precision]` reprezentat pe 8 octeti) si numere zecimale reprezentate cu precizia dorita (*tipul numeric sau decimal*).

Formatul de reprezentare a numerelor zecimale cu precizia dorita este: `numeric [(p,s)]` (sau `decimal [(p,s)]`), unde `p` (precizia) este numarul total de cifre afisate, iar `s` (scara) este numarul de cifre dupa punctul zecimal. Pentru a pastra precizia dorita, numerele de tip `decimal` sau `numeric` sunt memorate ca sir de caractere, fiecare caracter reprezentând o cifra, punctul zecimal sau semnul.

*Tipul sir de caractere* permite definirea sirurilor de caractere de lungime fixa (`char(n)` sau `character(n)`), precum si a sirurilor de caractere de lungime variabila (`varchar(n)`). Ambele tipuri pot reprezenta siruri de maximum `n` caractere, cu diferenta ca, pentru siruri de lungime mai mica decât `n`, la tipul `char(n)` se completeaza sirul cu spatii albe pâna la `n` caractere, în timp ce la tipul `varchar(n)` se memoreaza numai atâtea caractere câte are sirul dat.

*Tipurile pentru data calendaristica si timp* sunt: `date`, `time`, `timestamp`, `interval`.

#### 2.1.2 FUNCTII DEFINITE ÎN LIMBAJUL SQL2

Funcțiile definite în SQL2 sunt de doua categorii: *functii scalare* si *functii agregat*.

*Funcțiile scalare* se folosesc în expresii, care pot sa apara în diferite clauze ale instructiunilor SQL. Acestea primesc unul sau mai multe argumente si returneaza valoarea calculata, sau NULL, în caz de eroare. Argumentele functiilor pot fi constante (literale) sau valori ale atributelor specificate prin numele coloanelor corespunzatoare. Exista mai multe tipuri de functii scalare SQL: functii numerice (`sin`, `cos`, `ln`, `log`, etc.), functii pentru manipularea sirurilor de caractere, functii pentru data calendaristica si timp, functii de conversie.

*Funcțiile agregat* calculează un rezultat din mai multe linii ale unui tabel. Acestea sunt:

COUNT: returnează numărul de linii ale rezultatului (care îndeplinesc condiția WHERE)  
SUM: returnează suma tuturor valorilor dintr-o coloană  
MAX: returnează valoarea cea mai mare dintr-o coloană  
MIN: returnează valoarea cea mai mică dintr-o coloană  
AVG: returnează media valorilor dintr-o coloană

Aceste funcții se pot folosi cu clauza GROUP BY, dacă se calculează valoarea dorită (medie, sumă, etc.) prin gruparea liniilor în funcție de valoarea uneia sau mai multor coloane, sau fără clauza GROUP BY dacă se calculează valoarea dorită considerând toate tuplurile relației. De exemplu, comanda următoare va afișa salariul mediu al tuturor angajaților:

```
SELECT AVG(Salariu) FROM ANGAJATI;
```

### 2.1.3 INSTRUCȚIUNI SQL DE DEFINIRE A DATELOR

**Instrucțiunea de creare a unui tabel** (CREATE TABLE) definește atributele (coloanele) tabelului, domeniile atributelor și diferite constrângeri pe care datele înregistrate (valori ale atributelor) trebuie să le respecte pentru asigurarea integrității (corectitudinii) bazei de date. Sintaxa generală a acestei instrucțiuni este:

```
CREATE TABLE nume_tabel (  
    coloana_1 domeniu_1 [constrangeri_coloana],  
    coloana_2 domeniu_2 [constrangeri_coloana],  
    .....  
    coloana_n domeniu_n [constrangeri_coloana]  
    [constrangeri_tabel] );
```

Constrângerile impuse fiecărui atribut (coloană), ca și constrângerile de tabel, sunt optionale.

Se pot introduce una sau mai multe constrângeri de atribut (coloană) ca: PRIMARY KEY, NOT NULL, DEFAULT. Constrângerea de coloană PRIMARY KEY definește atributul pe care îl însoțește ca fiind cheie primară, adică un identificator unic al tuplului respectiv. Într-o relație nu pot exista două sau mai multe tupluri cu aceeași valoare a cheii primare. Dacă cheia primară este compusă (formată din mai multe atribute), atunci constrângerea de cheie primară se specifică după definirea atributelor, ca o constrângere de tabel sub forma:

```
[CONSTRAINT nume_constr] PRIMARY KEY (lista_atribute)
```

Această formă de definire se poate folosi și pentru o cheie primară simplă. Cheia străină se introduce cu construcția:

```
[CONSTRAINT nume_constr] FOREIGN KEY (cheie_straina)  
    REFERENCES relatie_referita (cheie_candidata)
```

Constrângerea NOT NULL specifică faptul că atributul respectiv nu poate lua valori nedefinite (NULL). Constrângerea DEFAULT introduce o valoare implicită a atributului respectiv, care va fi folosită la inițializarea valorilor unui tuplu nou introdus, atunci când nu se specifică o valoare pentru acest atribut. În lipsa parametrului DEFAULT, valorile implicite ale atributelor depind doar de tipul atributului (numerele reale primesc valoarea implicită 0, sirurile de caractere sunt siruri vide, etc.).

**Instrucțiunea de modificare a unui tabel** (ALTER TABLE) permite adăugarea sau ștergerea unor atribute, modificarea domeniilor unor atribute, precum și adăugarea, modificarea sau ștergerea unor constrângeri ale tabelului. De exemplu, instrucțiunea de adăugare a atributului DataAngajării în tabelul ANGAJATI se scrie în felul următor:

```
ALTER TABLE ANGAJATI ADD DataAngajarii date;
```

Pentru ștergerea unui atribut (coloană) dintr-un tabel, în instrucțiunea ALTER TABLE se folosește cuvântul-cheie DROP. De exemplu, ștergerea atributului Funcție din tabelul ANGAJAT se poate face cu comanda:

```
ALTER TABLE ANGAJATI DROP Funcție;
```

**Instrucțiunea de ștergere a unui tabel** este: DROP TABLE nume\_tabel.

## 2.1.4 INSTRUCȚIUNE SQL DE MANIPULARE A DATELOR

**Instrucțiunea SELECT** reprezintă blocul de interogare de bază și ea selectează informațiile dorite din tabelele bazei de date. Instrucțiunea SELECT este foarte puternică și are următoarea sintaxă generală:

```
SELECT [DISTINCT] lista_coloane FROM lista_tabele [WHERE conditie]
[clauze_secundare];
```

Se remarcă 3 secțiuni (clauze) importante ale construcției de interogare: clauza SELECT, clauza FROM și clauza WHERE.

Clauza SELECT introduce lista atributelor (coloanelor) unor tabele sau al expresiilor care vor fi selectate și afișate. Coloanele din listă trebuie să aparțină uneia din tabelele specificate în clauza FROM.

Ca rezultat al instrucțiunii de mai sus se pot obține două sau mai multe linii identice, dacă există angajați cu același nume și prenume. În general, dacă lista de atribute nu conține o cheie a relației, rezultatul operației SELECT poate conține linii duplicate. Pentru eliminarea liniilor duplicate se introduce parametrul DISTINCT și atunci rezultatul este o relație în sensul definiției din modelul relational.

Dacă lista de atribute este un asterisc (\*), atunci se selectează toate atributele produsului cartezian al tabelelor indicate prin clauza FROM, care îndeplinesc condiția din clauza WHERE. În clauza SELECT se pot redenumi atributele (coloane ale tabelelor) sau se pot specifica nume pentru expresii, folosind următoarea sintaxă:

```
SELECT nume1 [AS] noul_nume1,..., expresie [AS] nume_expresie
FROM lista_tabele [alte_clauze];
```

Se observă că noul nume atribuit unei coloane sau expresii urmează vechiului nume sau expresiei, precedat (optional, depinzând de implementare) de cuvântul-cheie AS.

Clauza FROM este obligatorie dacă într-una din clauzele SELECT, WHERE, HAVING apar nume de atribute (coloane ale unor tabele). În acest caz, lista de tabele care însoțeste clauza FROM trebuie să conțină numele tuturor tabelelor (separate prin virgulă) ale căror coloane se folosesc. Dacă lista conține mai mult de un tabel, atunci numele coloanelor din clauza SELECT trebuie să fie diferite și, dacă nu sunt diferite, atunci se califică numele coloanei cu numele tabelului căruia îi aparține (precedând numele atributului cu numele tabelului urmat de operatorul “punct” (.). De exemplu:

```
SELECT ANGAJATI.Nume, Prenume, SECTII.Nume FROM ANGAJATI, SECTII;
```

Clauza WHERE restricționează tuplurile returnate ca rezultat la acele tupluri care îndeplinesc condiția introdusă de această clauză. În forma cea mai obișnuită, clauza WHERE este urmată de o condiție, dată ca o expresie booleană.

Clauza ORDER BY introduce numele atributului după care se face ordonarea liniilor rezultate. Ordonarea este implicit în ordine crescătoare; dacă numele atributului este urmat de cuvântul DESC, ordonarea liniilor se face în ordine descrescătoare a valorilor celui atribut.

Clauza GROUP BY se folosește pentru a grupa rezultatele funcțiilor agregat (totalizatoare) după valoarea uneia sau mai multor coloane. Dacă se dorește calculul unei valori totalizatoare separat pe grupe de linii, atunci se introduce clauza GROUP BY, urmată de numele uneia sau mai multor coloane. În acest caz, funcția totalizatoare se aplică separat acelor linii care au aceeași valoare a atributelor listate de clauza GROUP BY. De exemplu, salariul mediu calculat separat pe grupe de angajați, fiecare grup fiind compus din linii care au aceeași valoare a atributului Funcție, se obține cu următoarea comandă SQL:

```
SELECT AVG(Salariu) FROM ANGAJATI GROUP BY(Funcție);
```

Clauza HAVING este asemănătoare clauzei WHERE, adică introduce o condiție pe care trebuie să o îndeplinească tuplurile rezultat, dar, în plus, permite utilizarea funcțiilor agregat în expresia condițională. De exemplu:

```
SELECT Nume, Prenume FROM ANGAJATI HAVING Salariu >= AVG(Salariu);
```

**Instrucțiunea INSERT** se folosește pentru introducerea liniilor și are următoarea sintaxă:

```
INSERT INTO nume_tabel (coloana_1,coloana_2,...)
VALUES (valoare_1,valoare_2,...);
```

Între valori și numele de coloane trebuie să existe o corespondență unu la unu. Lista de coloane poate să lipsească, dacă se introduc valori în toate coloanele tabelului, dar în această situație ordinea valorilor introduse trebuie să respecte ordinea atributelor.

**Instrucțiunea UPDATE** permite actualizarea valorilor coloanelor (atributelor) din una sau mai multe linii ale unui tabel. Aceasta are sintaxa:

```
UPDATE nume_tabel  
SET col_1 = expr_1, col_2 = expr_2,... [WHERE conditie];
```

Clauza WHERE impune ca actualizarea valorilor coloanelor să se efectueze numai asupra acelor linii (tupluri) care îndeplinesc condiția dată. Dacă este omisă clauza WHERE, atunci vor fi modificate valorile coloanelor din toate liniile tabelului.

**Instrucțiunea DELETE** permite ștergerea uneia sau mai multor linii dintr-un tabel și are următoarea sintaxă:

```
DELETE FROM nume_tabel[WHERE conditie];
```

Din tabel se șterg acele linii care îndeplinesc condiția dată în clauza WHERE. Dacă este omisă clauza WHERE, atunci vor fi șterse toate liniile din tabel.

**Integritatea referențială** este proprietatea bazei de date care garantează că oricare valoare a unei chei străine se regăsește printre valorile cheii candidate corespunzătoare din relația referită, sau cheia străină are valoarea NULL. Operațiile de modificare a stării unei relații (introducerea, ștergerea și actualizarea tuplurilor relației) trebuie să fie efectuate astfel încât să asigure menținerea integrității referențiale a bazei de date.

Stabilirea modului de ștergere sau de actualizare a tuplurilor se face în comenzile SQL de creare sau modificare a tabelelor, prin adăugarea uneia din opțiunile ON DELETE, respectiv ON UPDATE, constrângerii de cheie străină. Valorile posibile ale acestor opțiuni sunt RESTRICT (pentru ștergerea restricționată) sau CASCADE (pentru ștergerea în cascada); valoarea RESTRICT este implicită. De exemplu, instrucțiunea SQL de creare a tabelului ANGAJATI cu opțiunea de ștergere în cascada pentru cheia străină IdSectie este:

```
CREATE TABLE ANGAJATI (  
    IdAngajat    integer PRIMARY KEY,  
    Nume         varchar (20) NOT NULL,  
    .....  
    FOREIGN KEY (IdSectie) REFERENCES SECTII(IdSectie)  
    ON DELETE CASCADE );
```

## 2.2 PROIECTAREA BAZELOR DE DATE RELATIONALE

Proiectarea unei baze de date constă din proiectarea schemei conceptuale (logice) și fizice a acesteia, astfel încât să răspundă cerințelor utilizatorilor pentru un anumit set de aplicații. În general, se consideră că proiectarea unei baze de date se poate diviza în următoarele faze:

- Colectarea și analiza cerințelor.
- Proiectarea conceptuală a bazei de date.
- Alegerea unui SGBD.
- Proiectarea logică a bazei de date.
- Proiectarea fizică a bazei de date.

Cele cinci faze de proiectare enumerate mai sus nu se desfășoară strict într-o singură secvență. În multe cazuri este necesară modificarea proiectului dintr-o fază inițială într-una din fazele ulterioare, pentru a se obține rezultatele dorite. Aceste bucle de reacție între faze (sau în interiorul unei faze) sunt, în general, frecvente în cursul proiectării unei baze de date.

Înainte de a se proiecta efectiv o bază de date, este necesar să se cunoască ce rezultate se așteaptă utilizatorii potențiali să obțină de la baza de date respectivă și ce informații primare sunt disponibile pentru aceasta. De asemenea, este necesar să se cunoască ce aplicații se vor efectua (aplicații de gestiune a stocurilor, aplicații contabile, salarizare, etc.).

### 2.2.1 PROIECTAREA CONCEPTUALA A BAZELOR DE DATE

În faza de proiectare conceptuala a bazelor de date se proiecteaza schema conceptuala si schemele externe ale bazei de date.

Deși nu este obligatoriu, aceasta faza se poate mentine independenta de SGBD si produce un model de date de nivel înalt, care va fi implementat dupa transpunerea lui într-un model de date specific. Chiar daca proiectantii pot porni direct cu scheme conceptuale specifice unui anumit SGBD (care se mai numesc si scheme logice), este totusi recomandabil sa se realizeze mai întâi schema conceptuala de nivel înalt independenta de SGBD, deoarece aceasta este o descriere stabila si învaluabila a bazei de date. Alegerea unui SGBD si deciziile ulterioare de proiectare se pot schimba fara ca aceasta sa se schimbe.

Proiectul conceptual de nivel înalt se realizeaza pe baza cerintelor definite în prima etapa de proiectare si se reprezinta, în general printr-o diagrama Entitate-Asociere (extinsa).

**Modelul Entitate-Asociere (Entity-Relationship Model)** este un model conceptual de nivel înalt al unei baze de date, care defineste multimile de entitati si asocierile dintre ele, dar nu impune nici un mod specific de structurare si prelucrare a datelor. Elementele esentiale ale modelului Entitate-Asociere sunt *entitatile (entities)* si *asocierile* dintre acestea (*relationships*).

O *entitate (entity)* este "*orice poate fi identificat în mod distinctiv*"; o entitate se refera la un aspect al realitatii obiective care poate fi deosebit de restul universului si poate reprezenta un obiect fizic, o activitate, un concept, etc. Orice entitate este descrisa prin attributele sale. Un atribut (*attribute*) este o proprietate care descrie un anumit aspect al unei entitati.

Toate entitatile similare, care pot fi descrise prin aceleasi attribute, apartin unui acelasi *tip de entitate (entity type)*, iar colectia tuturor entitatilor de acelasi tip dintr-o baza de date constituie o *multime de entitati (entities set)*. În general, în modelul E-A se foloseste aceeași denumire atât pentru un *tip de entitate* cât si pentru *multimea entitatilor* de acel tip.

De exemplu, tipul de entitate "angajat" (al unei institutii) reprezinta orice persoana angajata a institutiei, care are o anumita functie si primeste un anumit salariu. Acest tip de entitate poate fi descris prin mai multe attribute, dintre care o parte sunt attribute de identificare a persoanei (Nume, Prenume, DataNasterii, Adresa), iar altele sunt attribute legate de activitatea acesteia în institutia respectiva (Functie, Salariu).

În proiectarea bazelor de date se considera doua categorii de entitati: entitati normale (puternice, obisnuite - *regular entities*) si entitati slabe (dependente - *weak entities*).

Entitatile normale au o existenta proprie în cadrul modelului, în timp ce entitatile slabe nu pot exista decât daca exista o entitate normala (puternica) cu care sunt asociate. De exemplu, o entitate "dependent" poate sa reprezinte o persoana care depinde de un angajat al unei institutii (adica se afla în întreținerea acestuia). O entitate "angajat" este o entitate puternica, deoarece ea exista în mod normal în modelul activitatii institutiei, în timp ce o entitate "dependent" este o entitate slaba: nu se va înregistra o astfel de persoana decât daca parintele (sustinatorul) acesteia este angajat în acea institutie.

O *asociere (relationship)* este o corespondenta între entitati din doua sau mai multe multimi de entitati. Gradul unei asocieri este dat de numarul de multimi de entitati asociate. Asocierile pot fi binare (de gradul 2, între 2 multimi de entitati) sau multiple (între k multimi de entitati,  $k > 2$ ).

*Asocierile binare* sunt, la rândul lor, de trei categorii, dupa numarul elementelor din fiecare dintre cele doua multimi puse în corespondenta de asocierea respectiva. Fiind date doua multimi de entitati,  $E_1$  si  $E_2$ , se definesc urmatoarele categorii de asocieri binare:

- *Asocierea "unul-la-unul" (one-to-one)* este asocierea prin care unui element (entitate) din multimea  $E_1$  îi corespunde un singur element din multimea  $E_2$ , si reciproc; se noteaza cu 1:1.
- *Asocierea "unul-la-multe" (one-to-many)* este asocierea prin care unui element din multimea  $E_1$  îi corespund unul sau mai multe elemente din multimea  $E_2$ , dar unui element din  $E_2$  îi corespunde un singur element în multimea  $E_1$ ; se noteaza cu 1:N.
- *Asocierea "multe-la-multe" (many-to-many)* este asocierea prin care unui element din multimea  $E_1$  îi corespund unul sau mai multe elemente din multimea  $E_2$ , si reciproc; se noteaza cu M:N.

O asociere între doua sau mai multe multimi de entitati este, în acelasi timp, o asociere între tipurile de entitati corespunzatoare.

**Diagrama Entitate-Asociere** (*Entity-Relationship Diagram*) reprezinta modelul Entitate-Asociere prin multimile de entitati si asocierile dintre acestea.

Exista numeroase variante de notatii pentru redarea diagramei E.A. Una dintre cele mai folosite notatii reprezinta un tip de entitate (precum si multimea de entitati de acel tip) printr-un dreptunghi, iar atributelor tipului de entitate prin elipse conectate printr-o linie continua la acesta (Fig. 2.1). Pentru entitatile puternice se utilizeaza un dreptunghi încadrat cu o linie simpla, iar pentru entitatile slabe se utilizeaza un dreptunghi încadrat cu linie dubla.

O asociere (*tip de asociere*) dintre doua sau mai multe tipuri de entitati se reprezinta printr-un romb conectat prin link-uri (linii continue, formate din unul sau mai multe segmente) la tipurile de entitati asociate. O asociere poate sa aiba sau nu un nume; daca are un nume, acesta poate fi înscris în rombul respectiv sau în vecinatatea acestuia. Categoria asocierii se noteaza prin înscrierea multiplicitatii pe fiecare link care conduce la un tip de entitate. Este posibil ca o asociere sa prezinte ea însasi attribute, si aceste attribute se reprezinta prin elipse conectate la asocierea respectiva.

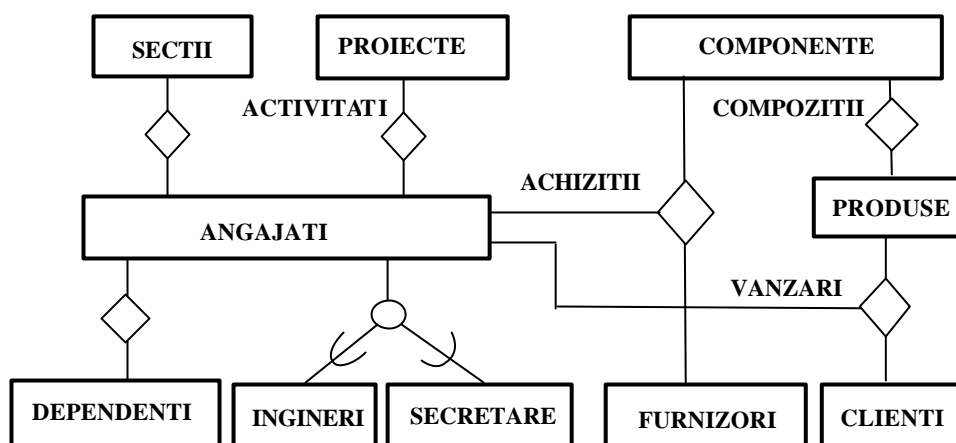
**Modelul Entitate-Asociere Extins** (*Enhanced Entity-Relationship Model*) permite definirea de subtipuri ale unui tip de entitati, care mostenesc attribute de la tipul de entitate pe care il extind (si care, în acest context, se numeste supertip) si au în plus attribute specifice semnificatiei lor. Prin definirea tipurilor si a subtipurilor de entitati se pot crea ierarhii de tipuri de entitati pe mai multe niveluri. Modelul *Entitate-Asociere Extins* se reprezinta printr-o diagrama EA extinsa, în care legatura între un supertip de entitati si subtipurile acestuia se reprezinta printr-o linie pe care se plaseaza un semicerc îndreptat catre supertip (Fig. 2.1).

**Exemplu de model Entitate-Asociere.** Se considera o baza de date a unei întreprinderi. Tipurile de entitati puternice (normale) care se pot defini pentru modelarea activitatii unei întreprinderi pot fi: SECTII, ANGAJATI, FURNIZORI, CLIENTI, PRODUSE, COMPONENTE (Fig. 2.1):

```
SECTII (Nume, Buget)
ANGAJATI (Nume, Prenume, DataNasterii, Adresa, Functie, Salariu)
FURNIZORI (Nume, Prenume, Adresa)
CLIENTI (Nume, Prenume, Adresa)
PRODUSE (Denumire, Descriere)
COMPONENTE (Denumire, Descriere)
```

La aceste multimi de entitati se adauga multimea de entitati slabe:

```
DEPENDENTI (Nume, Prenume, DataNasterii, GradRudenie)
```



**Fig. 2.1** Diagrama E-A a bazei de date a unei întreprinderi.

Pentru tipul ANGAJATI se defineste o specializare disjuncta partiala, cu subtipurile INGINERI si SECRETARE. Aceste subtipuri se afla în asociere 1:1 cu tipul de baza ANGAJATI, mostenesc attributele acestuia si fiecare mai contine attribute specifice:

INGINERI (Specialitatea)  
SECRETARE (VitezaRedactare)

Asocierile dintre multimile de entitati se stabilesc în functie de modul în care se desfasoara activitatea modelata. De exemplu, daca în întreprinderea respectiva fiecare angajat lucreaza într-o singura sectie, atunci între multimile de entitati `SECTII-ANGAJATI` exista o asociere 1:N.

O multime de entitati slabe se afla, de regula, în asociere N:1 cu multimea de entitati puternice de care depinde. In exemplul dat, între multimea `DEPENDENTI` si multimea de entitati `ANGAJATI` exista o asociere N:1. O multime de entitati de un subtip dat este, de regula, în asociere 1:1 cu multimea de entitati de supertipul acesteia. Pentru exemplul de mai sus, un angajat poate fi un (singur) inginer; iar un inginer este chiar un angajat, deci asocierea între multimile de entitati `ANGAJATI` si `INGINERI` exista o asociere cu raportul de cardinalitate 1:1.

## 2.2.2 PROIECTAREA LOGICA ABAZELOR DE DATE

În faza de proiectare logica a unei baze de date se realizeaza schema conceptuala globala si schemele conceptuale (vederile) externe pentru sistemul SGBD ales, pornind de la schema conceptuala si schemele externe de nivel înalt independente de SGBD, proiectate în faza precedenta. Aceasta faza de proiectare logica poate fi realizata în doua sub-faze: transpunerea schemei conceptuale în modelul de date al sistemului SGBD ales, dar independent de sistemul de gestiune propriu-zis si rafinarea schemei conceptuale si a schemelor externe obtinute anterior, astfel încât sa se utilizeze unele (sau cât mai multe) din facilitatile oferite de sistemul SGBD ales (modul de generare a cheilor primare, definirea constrângerilor, etc.).

Aceste doua sub-faze se pot realiza împreuna, folosind unul din instrumentele de proiectare oferite de sistemul SGBD ales. Rezultatul acestei faze de proiectare îl constituie, asadar, schema conceptuala si schemele externe ale bazei de date, dependente de sistemul SGBD ales si de modelul de date al acestuia. Pentru transpunerea modelului Entitate-Asociere (reprezentat prin diagrama E-A) în model relational se parcurg în principal doua etape: proiectarea relatiilor si proiectarea asociierilor.

**Proiectarea relatiilor.** În Fig. 2.2 este data schema conceptuala a bazei de date relationale corespunzatoare diagramei E-A din Fig. 2.1, dezvoltata în MS Access. În MS Access relatiile si asocierile între ele sunt reprezentate vizual în diagrama de asocieri (*Relationships*), care este corespondentul relational al diagramei E-A.

Multimile de entitati puternice (normale) din diagrama E-A devin relatii, cu attributele date de attributele entitatilor. În astfel de relatii cheia primara se defineste fie ca o cheie naturala (combinatie de attribute care definesc în mod unic un tuplu al relatiei), fie ca o cheie primara artificiala. În exemplul prezentat, în în fiecare din relatiile care corespund multimilor de entitati puternice s-a adaugat câte o cheie primara artificiala (`IdAngajat`, `IdSectie`, `IdProiect`, etc.).

Multimile de entitati slabe din diagrama E-A devin, de regula, relatii aflate în asociere N:1 cu relatia corespunzatoare multimii de entitati de care acestea depind. Pentru realizarea acestei asocieri, în relatia dependenta se adauga o cheie straina care refera cheia primara a relatiei referite (puternice).

Cheia primara a unei relatiei dependente poate fi o combinatie formata din atributul cheie straina si alte attribute care asigura posibilitatea de identificare unica a unui tuplu, sau poate fi o cheie artificiala. Cheia primara a relatiei `DEPENDENTI` este compusa din atributul cheie straina (`IdAngajat`, care refera cheia primara a relatiei `ANGAJATI`) si attributele `Nume` si `Prenume` (ale persoanei dependente).

Multimile de entitati care sunt subtipuri ale unui tip de entitate dat devin relatii aflate în asociere 1:1 cu relatia corespunzatoare multimii de entitati de tipul respectiv (supertip). Pentru realizarea acestei asocieri, în relatia corespunzatoare subtipului de entitati se defineste o cheie straina care refera cheia primara din relatia corespunzatoare supertipului de entitati; aceasta cheie straina este în acelasi timp si cheia primara în relatia corespunzatoare subtipului de entitati.

În exemplul prezentat, asocierile `ANGAJATI-INGINERI` si `ANGAJATI-SECRETARE` sunt asocieri 1:1. În relatia `INGINERI` atributul `IdAngajat` este cheie straina care refera cheia primara cu acelasi nume din relatia `ANGAJATI` si este în acelasi timp si cheia primara; la fel, în relatia `SECRETARE` atributul `IdAngajat` este cheie straina care refera cheia primara cu acelasi nume din relatia `ANGAJATI` si este în acelasi timp si cheia primara.

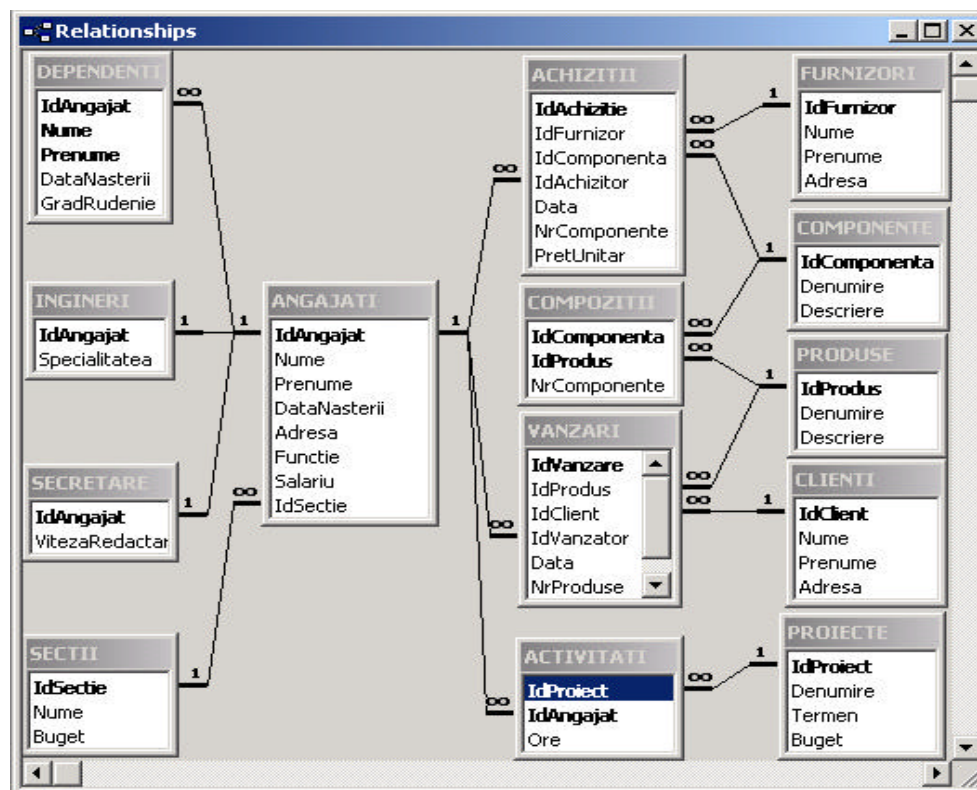


Fig. 2.2 Diagrama bazei de date INTREPRINDERE în MS Access.

**Proiectarea asocierilor.** Asocierea binară N:1 dintre două mulțimi de entități puternice din diagrama E-A se realizează în modelul relational prin intermediul unei chei straine în prima relație (cea cu multiplicitatea N a asocierii) care referă cheia primară (sau o cheie candidată) din relația referită (cea cu multiplicitatea 1 a asocierii). De exemplu, asocierea N:1 între relațiile ANGAJATI-SECTII se realizează prin cheia straină *IdSectie* adăugată relației ANGAJATI, care referă cheia primară cu același nume a relației SECTII.

Asocierea binară M:N dintre două mulțimi de entități din diagrama E-A se realizează în modelul relational prin intermediul unei noi relații, numită relație de asociere. Aceasta nouă relație se află în asociere M:1, respectiv N:1 cu fiecare din cele două relații date prin intermediul a două chei straine care referă cheile primare (sau chei candidate) din relațiile date.

De exemplu, pentru a reprezenta asocierea M:N dintre relațiile COMPONENTE-PRODUSE se adaugă o nouă relație numită COMPOZITII, care conține cheile straine *IdComponenta* și *IdProdus*, care referă cheile primare cu același nume din relațiile COMPONENTE, respectiv PRODUSE. Cheia primară a unei relații de asociere poate fi o cheie artificială sau poate fi compusă din cheile straine care referă cele două relații asociate, eventual împreună cu alte atribute ale relației, care caracterizează asocierea respectivă. Așa cum se poate vedea în Fig. 2.2, cheia primară a relației COMPOZITII este formată din cele două chei straine pe care le conține.

Asocierea binară 1:1 între două mulțimi de entități puternice se poate transpune în modelul relational în două moduri: fie prin intermediul unei chei straine (ca un caz particular al unei asocieri N:1), fie printr-o relație de asociere (ca un caz particular al unei asocieri M:N).

Asocierea binară 1:1 dintre o mulțime de entități de un subtip și mulțimea de entități supertip din diagrama Entitate-Asociere este tot o asociere binară cu raportul de cardinalitate 1:1 între relațiile corespunzătoare în modelul relational. Acest tip de asociere se realizează prin definirea în relația corespunzătoare subtipului de entități a unei chei straine care referă cheia primară din relația corespunzătoare supertipului de entități; această cheie straină este în același timp și cheie primară în relația corespunzătoare subtipului de entități. Acest mod de definire a fost folosit pentru asocierea dintre tabelele INGINERI, SECRETARE și tabelul ANGAJATI.



Asocierea multipla M:N:P:.... dintre mai mult de doua multimi de entitati din diagrama E-A se realizeaza în mod asemanator cu asocierea binara, prin intermediul unei noi relatii care se afla în asociere M:1, N:1, P:1, etc, cu fiecare din relatiile date. Aceasta asociere este realizata prin intermediul mai multor chei straine, fiecare cheie straina referind cheia primara (sau o cheie candidata) dintr-una din relatiile date. De exemplu, relatia ACHIZITII realizeaza asocierea între relatiile COMPONENTE, FURNIZORI, ANGAJATI. Ea contine trei chei straine (IdComponenta, IdFurnizor, IdAchizitor) care refera relatiile COMPONENTE, FURNIZORI, respectiv ANGAJATI, iar cheia primara este cheia artificiala IdAchizitie.

### 2.2.3 PROIECTAREA FIZICA A BAZELOR DE DATE

Proiectarea fizica a bazei de date este procesul de alegere a structurilor de memorare si de acces la fisierele bazei de date, pentru a obtine performante cât mai bune, pentru cât mai multe din aplicatiile proiectate. Ca parametri generali de alegere a optiunilor proiectului fizic al unei baze de date relationale se pot enumera: timpul de raspuns, utilizarea spatiului de memorare, capacitatea tranzactionala. Deciziile de proiectare fizica se pot lua numai dupa o analiza a aplicatiilor care se vor executa si în principal a interogarilor si tranzactiilor pe care acestea le vor lansa. În urma analizei se pot sintetiza informatii care sa dea imaginea de ansamblu a utilizarii atributelor relatiilor bazei de date: care attribute sunt actualizate cel mai frecvent, care attribute sunt folosite cel mai frecvent în selectii ale interogarilor, etc. Aceste informatii se folosesc pentru stabilirea indexurilor secundare ale relatiilor.

## 2.3 PARTICULARITATILE LIMBAJULUI SQL SI DE PROIECTARE A BAZELOR DE DATE ÎN DIFERITE SISTEME SGBD

Majoritatea sistemelor SGBD relationale actuale suporta standardul SQL2, dar fiecare implementeaza, de fapt, un dialect specific al limbajului SQL. În diferitele implementari ale limbajului SQL pot sa lipseasca unele comenzi prevazute în standardul SQL2, dar pot exista extensii specifice, neprevazute în standard, care micsoreaza gradul de portabilitate a aplicatiilor. În continuare vor fi prezentate particularitatile limbajului SQL si de proiectare a bazelor de date în diferite sisteme SGBD.

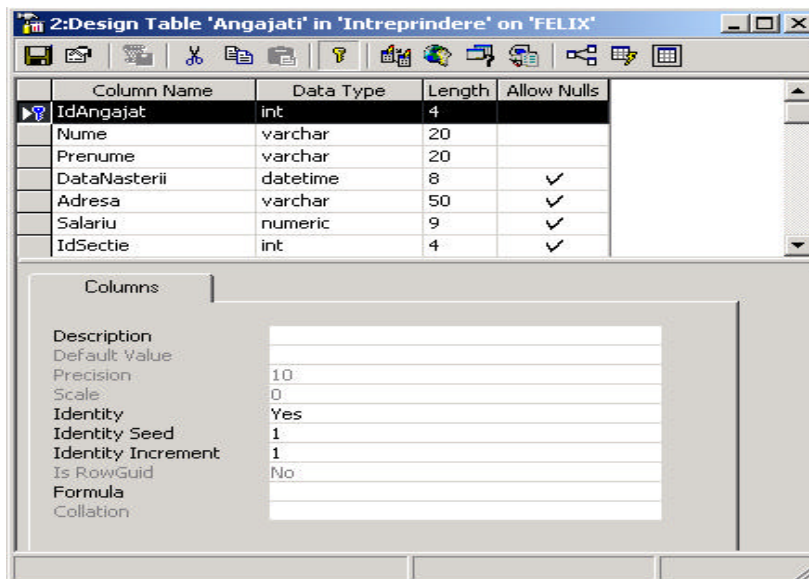
### 2.3.1 SISTEMUL SQL SERVER

Sistemul SQL Server poate executa programe în limbajul SQL sau în limbajul Transact-SQL, care este extensia procedurala a limbajului SQL. Limbajul Transact-SQL contine instructiuni SQL (asemanatoare celor specificate în standardul SQL2) precum si instructiuni de control al executiei (care vor fi prezentate în Capitolul 4). Instructiunile Transact-SQL se transmit sistemului grupate în loturi de executie (*batches*), prin intermediul programelor de aplicatii sau al programelor utilitare *osql* sau *Query Analyzer*.

Proiectarea tabelor se poate face atât vizual, folosind generatoarele de cod (*Wizards*) ale programului utilitar *SQL Server Enterprise Manager*, cât si prin comenzi (grupate în scripturi) care contin loturi de executie Transact-SQL, executate de la consola (folosind utilitarul *osql*) sau din programul *Query Analyzer*.

Proiectarea vizuala a tabelor se poate face în programul *SQL Server Enterprise Manager*. La comanda *New Table*, care se actioneaza din meniul contextual care se deschide la apasarea butonului dreapta al mouse-ului atunci când este selectat directorul *Tables* al bazei de date proprii, se deschide o fereasta de proiectare foarte asemanatoare cu cea din MS Access, cu unele diferente (care se pot observa în Fig. 2.3 de mai jos). La definirea cheii primare se poate specifica proprietatea de autoincrementare cu optiunea *IDENTITY*, care este echivalenta cu optiunea *AutoNumber* din MS Access, dar, în plus, aceasta optiune permite specificarea valorii de început a secventei de numere crescatoare (*Identity Seed*), si valoarea de incrementare (*Identity Increment*) care este implicit 1.

Asocierea între relatii se stabileste prin comanda *New Database Diagram* din meniul de context care se deschide la apasarea butonului dreapta al mouse-ului atunci când este selectat subdirectorul *Diagrams* al bazei de date. La aceasta comanda se deschide o fereasta de proiectare a asocierilor asemanatoare cu cea din MS Access (*Relationships*), prin care se stabilesc cheile straine si modul de referire între relatii.



**Fig. 2.3** Fereastra de proiectare a unui tabel folosind SQL Server Enterprise Manager.

Scripturile de comenzi în SQL Server contin loturi de executie Transact-SQL si pot fi executate prin intermediul unor programe utilitare, cum sunt *osq* sau *Query Analyzer*. Pentru detalii privind folosirea acestor programe este necesar, bineînțeles, sa fie studiata documentatia oferita de furnizor (*Books Online*). Ca exemplu se prezinta scriptul de creare a tabelelor SECTII, ANGAJATI în baza de date proprie din sistemul SQL Server (fisier *creare\_tabele\_sqlserver.sql*).

Script de creare a tabelelor ANGAJATI si SECTII în SQL Server

---

```

DROP TABLE ANGAJATI
GO
CREATE TABLE ANGAJATI (
    IdAngajat      int PRIMARY KEY IDENTITY,
    Nume           varchar(20) NOT NULL,
    Prenume       varchar(20) NOT NULL,
    DataNasterii   datetime,
    Adresa        varchar(50),
    Salariu        decimal DEFAULT 2800,
    IdSectie       int)
GO
DROP TABLE SECTII
GO
CREATE TABLE SECTII (
    IdSectie       int PRIMARY KEY IDENTITY,
    Nume           varchar(50) NOT NULL,
    Buget          decimal)
GO
ALTER TABLE ANGAJATI
    ADD CONSTRAINT FK_ANGAJATI FOREIGN KEY (IdSectie)
    REFERENCES SECTII(IdSectie)
GO

```

---

La inserarea liniilor în tabele (cu instructiuni `INSERT`), nu se admite precizarea valorii pentru un atribut cheie primara de tip `IDENTITY`. Daca se executa fisierul *introducere\_sqlserver.sql* pentru introducerea unor linii în tabelele SECTII si ANGAJATI, se va observa ca liniile în care se specifica valoarea cheii primare nu sunt admise (produc eroare de executie). De asemenea, sistemul

efectueaza verificarea cheilor straine si nu admite tupluri care contin o cheie straina a carui valoare nu se regaseste în nici o valoare a cheii primare referite (Fig. 2. 4).

Cheile primare au fost definite de tipul `IDENTITY`, care este un atribut cu autoincrementare, iar cheia straina din tabelul `ANGAJATI` s-a definit printr-o instructiune `ALTER TABLE`. Aceasta modalitate este frecvent folosita, deoarece permite crearea tabelelor în orice ordine si adaugarea dupa aceea a cheilor straine. Daca s-ar fi dorit definirea cheii straine chiar în instructiunea `CREATE TABLE` `ANGAJATI`, atunci relatia referita (în acest caz relatia `SECTII`) ar fi trebuit sa fie definit înaintea relatiei care refera (în acest caz relatia `ANGAJATI`). La proiectarea unei baze de date mari, cu multe relatii si referiri între ele, este destul de dificil de stabilit ordinea completa de referiri între tabele si de aceea se prefera definirea separata a cheilor straine, prin instructiuni `ALTER TABLE`.

În pagina de afisare *Messages* se gasesc mesajele de eroare returnate de sistemul de gestiune. La executia instructiunilor din fereastra de interogari din Fig. 2.4 se obtin urmatoarele mesaje:

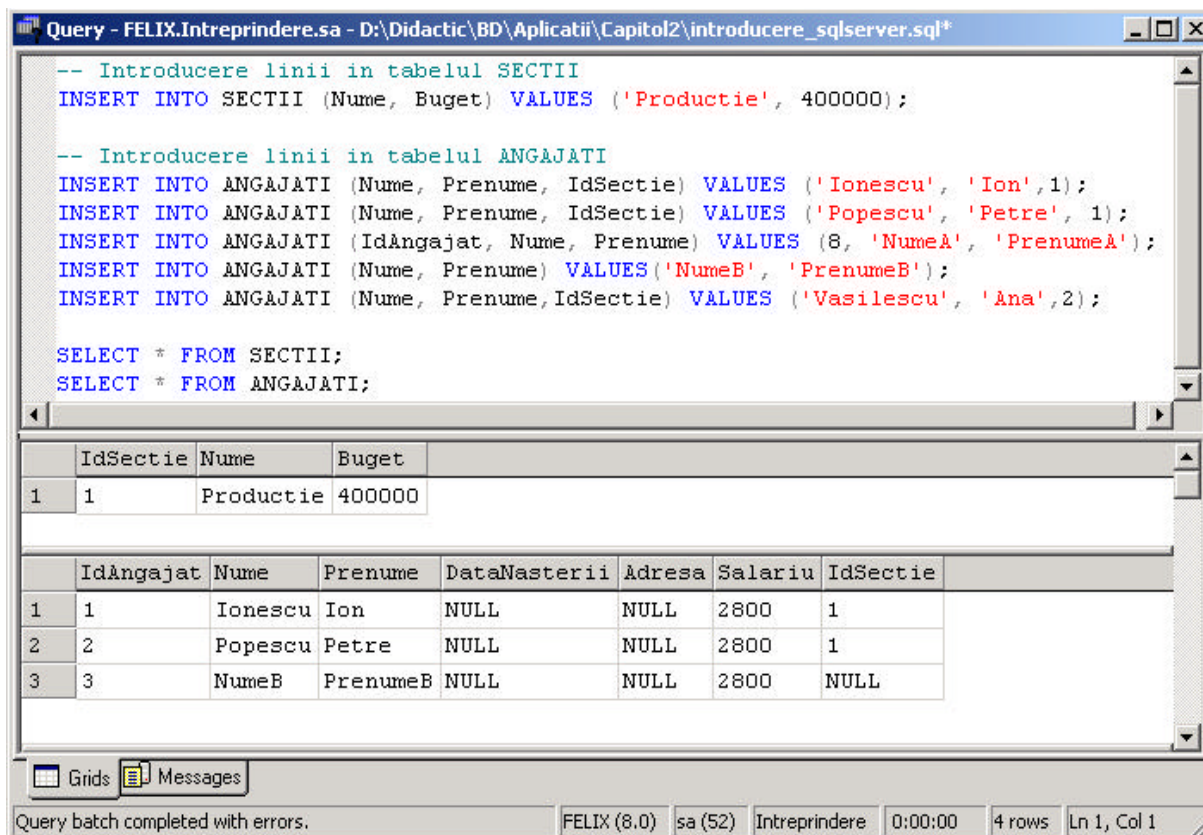
```
Cannot insert explicit value for identity column in table 'ANGAJATI'
when IDENTITY_INSERT is set to OFF.
```

```
INSERT statement conflicted with COLUMN FOREIGN KEY constraint
'FK_ANGAJATI'. The conflict occurred in database 'Intreprindere',
table 'SECTII', column 'IdSectie'.
```

Primul mesaj se refera la faptul ca nu se admit valori explicite pentru o cheie cu proprietatea `IDENTITY` decât daca se seteaza la `ON` proprietatea `IDENTITY_INSERT`. Instructiunea Transact-SQL care seteaza aceasta proprietate este:

```
SET IDENTITY_INSERT [database.[owner.]]{table}{ON|OFF}
```

Cel de-al doilea mesaj se refera la faptul ca nu se admit tupluri care nu respecta integritatea referentiala (egalitatea valorii cheii straine cu o valoare a cheii primare din relatia referita).



Query - FELIX.Intreprindere.sa - D:\Didactic\BD\Aplicatii\Capitol2\introducere\_sqlserver.sql\*

```
-- Introducere linii in tabelul SECTII
INSERT INTO SECTII (Nume, Buget) VALUES ('Productie', 400000);

-- Introducere linii in tabelul ANGAJATI
INSERT INTO ANGAJATI (Nume, Prenume, IdSectie) VALUES ('Ionescu', 'Ion', 1);
INSERT INTO ANGAJATI (Nume, Prenume, IdSectie) VALUES ('Popescu', 'Petre', 1);
INSERT INTO ANGAJATI (IdAngajat, Nume, Prenume) VALUES (8, 'NumeA', 'PrenumeA');
INSERT INTO ANGAJATI (Nume, Prenume) VALUES ('NumeB', 'PrenumeB');
INSERT INTO ANGAJATI (Nume, Prenume, IdSectie) VALUES ('Vasilescu', 'Ana', 2);

SELECT * FROM SECTII;
SELECT * FROM ANGAJATI;
```

	IdSectie	Nume	Buget
1	1	Productie	400000

	IdAngajat	Nume	Prenume	DataNasterii	Adresa	Salariu	IdSectie
1	1	Ionescu	Ion	NULL	NULL	2800	1
2	2	Popescu	Petre	NULL	NULL	2800	1
3	3	NumeB	PrenumeB	NULL	NULL	2800	NULL

Query batch completed with errors. FELIX (8.0) sa (52) Intreprindere 0:00:00 4 rows Ln 1, Col 1

Fig. 2.4. Afisarea liniilor inserate în tabelul `ANGAJATI` în SQL Server Query Analyzer.

### 2.3.2 SISTEMUL MS ACCESS

Sistemul MS Access ofera o interfata grafica de proiectare a bazelor de date, cu mai multe instrumente software de generare a tabelelor, asocierilor, formularelor, etc.

Pentru crearea sau modificarea tabelelor, în fereastra *Database*, se selecteaza comanda *Tables* care afiseaza panoul cu toate tabelele existente. La comanda de creare a unui tabel nou (comanda *New*), se deschide o fereastra de dialog (*New Table*) în care sunt prezentate mai multe optiuni de afisare si creare:

- *Datasheet View* prezinta o foaie de calcul alba în care se introduc valorile datelor. Daca nu se definesc tipurile de date în modul de afisare *Design*, programul MS Access le asigneaza singur.
- *Design View* este o grila în care se pot selecta definitiile datelor din diferite liste; în acest mod de afisare nu se introduce în mod explicit nici o data.
- *Table Wizard* este un program expert care, dupa alegerea unei baze de date predefinite, conduce procesul de selectare a câmpurilor si de stabilire a cheilor si a sistemului de asocieri.
- *Import Table* este o metoda folosita pentru a importa un tabel de date dintr-un alt fisier, creat în programul MS Access sau într-o alta aplicatie de baza de date care este recunoscuta de catre Access.
- *Link Table* opereaza la fel ca metoda anterioara, dar datele externe ramân în fisierul extern.

Atunci când se creaza un tabel nou în modul de afisare *Design View*, pe ecran apare o fereastra care are în partea superioara "grila de câmpuri" (*Field Grid*) - locul în care se introduc numele si se specifica tipul câmpurilor (coloanelor) care vor alcatui tabelul. Panoul din partea de jos, denumit "proprietatile câmpurilor" (*Field Properties*), permite modificarea proprietatilor fiecarui câmp (coloana) din tabel. Un nou câmp într-un tabel se creaza astfel:

- Se introduce un nume în coloana *Field Name*.
- Se selecteaza un tip de date în coloana *Data Type* din caseta combinata corespunzatoare.
- Optional se poate introduce în coloana *Description* un comentariu care descrie modul de utilizare a câmpului.

La închiderea ferestrei modului de afisare *Design View*, sistemul MS Access salveaza modificarile efectuate în tabelul original, în cazul editarii unui tabel existent, sau solicita introducerea unui nume pentru tabelul nou creat.

Tipul de date selectat pentru fiecare câmp în parte determina modul de stocare folosit de MS Access. De aceea, selectarea tipului corect de date pentru fiecare câmp este un element important în vederea obtinerii unor informatii corecte din baza de date. În aplicatie se pot folosi tipurile de date *Text* (un sir de max. 50 caractere), *Number* (un numar întreg sau în virgula mobila), *Date/Time*, *AutoNumber* (un numar întreg care este incrementat automat pe masura ce sunt introduse noi înregistrari într-un tabel, folosit ca si cheie primara).

Pentru fiecare tabel trebuie sa fie specificata cheia primara (*Primary Key*) care este o submultime a câmpurilor (coloanelor) tabelului cu proprietatea ca are valoare unica pentru fiecare din rândurile (tuplurile) tabelului. Cheia primara se poate stabili pe unul sau mai multe câmpuri, prin selectarea acestora si actionarea comenzii *Primary Key* din bara de instrumente (care are ca pictograma o cheie de lacat).

Celelalte proprietati ale unui câmp din tabel se stabilesc în panoul *Field Properties* si depind de tipul de date al acestuia si de calitatea de a apartine cheii primare sau nu. Toate tipurile de date prezinta mai multe proprietati (optiuni), dintre care unele pot fi configurate. În general, optiunile prestabilite de sistemul MS Access sunt satisfacatoare pentru cele mai multe câmpuri de date si numele lor sunt suficient de explicative. O atentie mai deosebita trebuie sa fie acordata proprietatilor: "câmp cerut" (*Required*), "admite lungime zero" (*Allow Zero Length*) si "câmp indexat" (*Indexed*).

Un câmp pentru care se selectează opțiunea *Yes* pentru proprietatea *Required* este un câmp în care nu se admit valori *NULL*; dacă se selectează opțiunea *No*, atunci valoarea acestui câmp poate să fie specificată sau nu, nespecificarea valorii însemnând o valoare de *NULL* pentru acel câmp.

Proprietatea *Allow Zero Length* este prezentă numai pentru tipul de date *Text*. Opțiunea *Yes* pentru această proprietate validează acceptarea unui text de lungime zero, iar opțiunea *No* invalidează un text de lungime zero.

O altă proprietate a câmpurilor care poate fi configurată este proprietatea *Indexed*. Dacă se selectează opțiunea *No*, atunci câmpul nu este indexat. Dacă se selectează una din opțiunile *Yes(No Duplicates)*, sau *Yes (Duplicates OK)* sistemul MS Access creează un index (o structură de date diferită de tabelul înșus), care este folosit pentru căutarea rapidă a înregistrărilor după valoarea acelui câmp. Atunci când nu se admit duplicate, trebuie ca valorile din câmpul indexat să fie unice în înregistrările tabelului. Acest lucru se asigură automat dacă acel câmp este cheie primară; dacă acel câmp nu este cheie primară, atunci valorile introduse sunt verificate și se rejectează acele înregistrări care au valori duplicate în câmpul astfel indexat. Pentru un câmp care constituie singură cheia primară, se atribuie în mod automat un index fără duplicate.

Cheile străine permit stabilirea asocierilor între tabele și în MS Access se definesc în două etape. În prima etapă, la crearea tabelelor, câmpurile (sau câmpul) care vor constitui cheia străină trebuie să fie definite de același tip de date (cu același domeniu) ca și câmpurile corespunzătoare din cheia primară din tabela pe care o referă. După definirea tabelelor (tabelele referite și tabelele care referă), se folosește comanda de meniu *Tools/Relationships* (sau comanda *Relationships* din bara de instrumente, care are o pictogramă reprezentând un arbore) pentru a defini asocierile și deci cheile străine între tabele.

La acționarea comenzii *Relationships*, programul Access afișează o fereastră numită *Relationships* și mai multe comenzi de meniu asociate acestei ferestre. În fereastra *Relationships* sunt reprezentate tabelele asociate, fiecare tabelă având toate câmpurile definite, iar o asociere este reprezentată printr-un link (conexiune) între două tabele, în dreptul atributelor (câmpurilor) corespondente. Tabelele afișate pot fi rearanjate în cadrul ferestrei trăgându-le cu mouse-ul (Fig. 2.5).

Pentru a crea o nouă asociere între două tabele, se parcurg următorii pași:

1. Se adaugă în fereastra *Relationships* tabelele între care se dorește crearea de asocieri. Pentru aceasta se acționează comanda *Show Table* din meniul *Relationships* sau din meniul de context, obținut prin click pe butonul dreapta al mousului în fereastra *Relationships*. La această comandă, se deschide încă o fereastră, cu titlul *Show Table*, care listează toate tabelele definite. Se selectează una sau mai multe tabele și se dă comanda de buton *Add*, care introduce tabelele selectate în fereastra *Relationship*. După aceasta, fereastra *Show Table* poate fi închisă (cu comanda de buton *Close*).
2. Cheile primare din fiecare tabelă sunt afișate cu caractere îngrosate. Cu mouse-ul, se trage numele cheii primare din tabelă referențiată peste numele câmpului corespunzător cheii străine sau cheii primare din tabelă care referențiază. Va fi afișată o nouă fereastră *Relationships*, care permite stabilirea unor opțiuni de asociere între tabele. Prin acest mecanism, se definesc atât asocieri 1:1 cât și asocieri 1:N. Link-ul de conectare între două tabele asociate are eticheta 1 pe capatul dinspre tabelă referențiată (care conține cheia primară) și eticheta  $\infty$  pe capatul dinspre tabelă care referențiază (care conține cheia străină).
3. În fereastra *Relationships* (Fig. 2.5) apar numele câmpurilor care au fost asociate. În majoritatea situațiilor, se recomandă selectarea casetei de validare *Enforce Referential Integrity* (forțarea integrității referențiale), ceea ce impune verificarea condiției de integritate referențială, adică pentru cheia străină din tabelă care referențiază nu se admit decât valori care există în cheia primară dintr-un tuplu (linie) din tabelă referențiată.
4. În fereastra de editare a unei asocieri *Relationships* se mai poate valida opțiunea de "actualizare în cascada" a câmpurilor corelate prin referire (*Cascade Update Related Fields*) și opțiunea de "stergere în cascada" a câmpurilor corelate prin referențiere (*Cascade Delete Related Fields*).

Comanda de buton *Join Type* (Tipul de cuplare) permite stabilirea tipului de cuplare (*join*) între tabele. La acționarea acestei, comenzi se deschide o fereastră de dialog modal (*Join Properties*) prin care se poate selecta unul din trei tipuri de operații de cuplare. Prima opțiune este cea implicită (cuplare internă - *internal join*) este cea mai frecvent utilizată; celelalte două tipuri (cuplări externe la dreapta sau la stânga) vor fi studiate din documentația MS Access (*Help*).

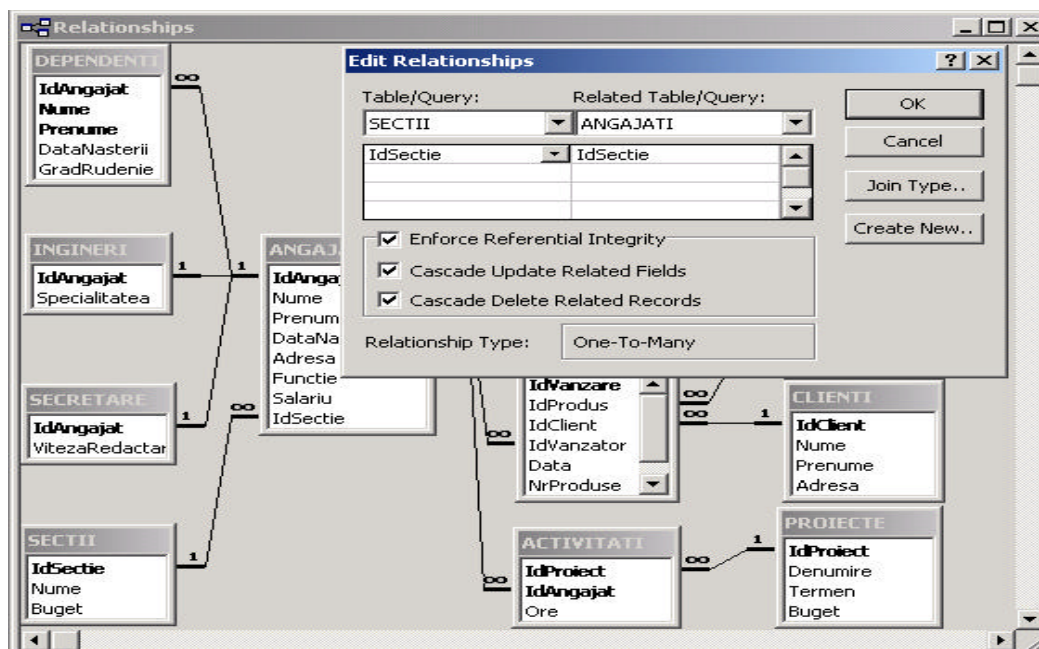


Fig. 2.5 Crearea unei asocieri între două tabele în MS Access.

### 2.3.3 SISTEMUL ORACLE

În sistemul Oracle proiectarea bazelor de date se poate realiza prin comenzi SQL transmise serverului din programele de aplicații sau prin intermediul unor programe utilitare (*SQL\* Plus*, *SQL\* Plus Worksheet* și altele).

În Oracle, pentru generarea valorilor unei chei primare artificiale se creează o secvență (cu comanda *CREATE SEQUENCE*) și la fiecare apel al metodei *NEXTVAL* al secvenței se obține următoarea valoare din succesiunea de numere întregi generate.

Pentru crearea tabelor *ANGAJATI* și *SECTII* se lansează executia în *SQL\* Plus Worksheet* a fișierului *create\_tabele\_oracle.sql* de mai jos:

Script de creare a tabelor *ANGAJATI* și *SECTII* în Oracle

```
DROP TABLE ANGAJATI;
DROP TABLE SECTII;
DROP SEQUENCE PK_ANGAJATI;
DROP SEQUENCE PK_SECTII;
CREATE TABLE ANGAJATI (
    IdAngajat      NUMBER PRIMARY KEY,
    Nume           varchar(20)    NOT NULL,
    Prenume       varchar(20)    NOT NULL,
    DataNasterii   date,
    Adresa         varchar(50),
    Salariu        decimal DEFAULT 2800,
    IdSectie       NUMBER);
CREATE TABLE SECTII (
    IdSectie       NUMBER PRIMARY KEY,
    Nume           varchar(50)    NOT NULL,
    Buget          decimal);
```



```

CREATE SEQUENCE PK_SECTII;
CREATE SEQUENCE PK_ANGAJATI;
ALTER TABLE ANGAJATI ADD FOREIGN KEY (IdSectie)
REFERENCES SECTII(IdSectie);

```

La introducerea datelor într-un tabel se apelează metoda NEXTVAL a secvenței cheii primare a tabelului respectiv. De exemplu, introducerea unor linii în tabelele SECTII și ANGAJATI se realizează cu scriptul de mai jos:

Script pentru introducerea datelor în tabele în Oracle

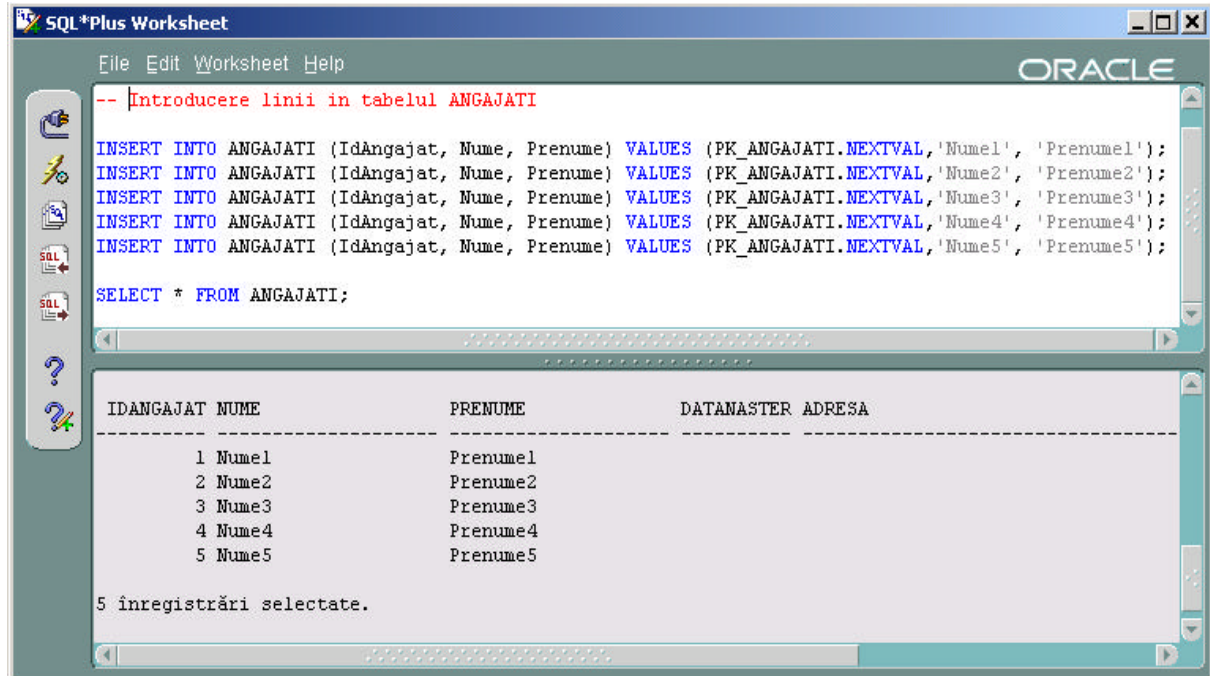
```

-- Introducere linii in tabelul SECTII
INSERT INTO SECTII (IdSectie, Nume, Buget)
VALUES (PK_SECTII.NEXTVAL, 'Productie', 400000);

-- Introducere linii in tabelul ANGAJATI
INSERT INTO ANGAJATI (IdAngajat, Nume, Prenume)
VALUES (PK_ANGAJATI.NEXTVAL, 'Nume1', 'Prenume1');
INSERT INTO ANGAJATI (IdAngajat, Nume, Prenume)
VALUES (PK_ANGAJATI.NEXTVAL, 'Nume2', 'Prenume2');
INSERT INTO ANGAJATI (IdAngajat, Nume, Prenume)
VALUES (PK_ANGAJATI.NEXTVAL, 'Nume3', 'Prenume3');
INSERT INTO ANGAJATI (IdAngajat, Nume, Prenume)
VALUES (PK_ANGAJATI.NEXTVAL, 'Nume4', 'Prenume4');
INSERT INTO ANGAJATI (IdAngajat, Nume, Prenume)
VALUES (PK_ANGAJATI.NEXTVAL, 'Nume5', 'Prenume5');

```

După introducerea acestor linii, se poate observa conținutul tabelului ANGAJATI cu comanda SELECT \* FROM ANGAJATI; executată în *Oracle SQL\* Plus Worksheet* (Fig. 2. 6).



**Fig. 2.6** Afisarea conținutului tabelului ANGAJATI în Oracle SQL\* Plus Worksheet.

În Oracle se impune introducerea valorilor pentru attributele cheii primare și aceste valori se extrag, de regulă, din secvențe. Nu este recomandabil să se amestece valori generate de secvențe cu alte valori, deoarece nu există garanția că nu vor avea loc suprapuneri, ceea ce conduce la refuzarea inserării tuplurilor care au valori duplicate a cheii primare.

### 2.3.4 SISTEMUL MYSQL

Limbajul SQL implementat în sistemul MySQL respecta majoritatea caracteristicilor standardului SQL2, și aceasta corespondența se îmbunătățește de la o versiune la alta.

Instrucțiunile SQL de definire și manipulare a datelor se pot introduce de la monitorul *mysql* direct (de la tastatură), sau folosind fișiere de script lansate în execuție cu comanda `source nume_fisier`. La definirea unei tabel (cu instrucțiunea `CREATE TABLE`) se poate stabili cheia primară printr-un atribut dat ca număr întreg (`int`) cu proprietatea de autoincrementare (`AUTO_INCREMENT`), ceea ce asigură unicitatea cheii în cadrul relației.

Considerând că a fost deja creat un utilizator (cu numele `user1`) și o bază de date a acestuia (cu același nume, `user1`) se lansează monitorul *mysql* cu comanda:

```
C:\mysql -u user1 -p user1
```

La promptul monitorului *mysql* se pot introduce comenzile de creare a tabelelor manual sau executând fișierul de script `creare_tabele_mysql.sql` (care se găsește în directorul capitoului 2 al îndrumarului):

```
mysql> source creare_tabele_mysql.sql;
```

În fișierul script se înscriu cu un editor de text oarecare instrucțiunile de creare a tabelelor. De exemplu, pentru fișierul `creare_tabele_mysql.sql` poate avea următorul conținut:

Script de creare a tabelelor ANGAJATI și SECTII în MYSQL

---

```
DROP TABLE ANGAJATI;
DROP TABLE SECTII;

CREATE TABLE ANGAJATI (
    IdAngajat    int PRIMARY KEY AUTO_INCREMENT,
    Nume         varchar(20) NOT NULL,
    Prenume     varchar(20) NOT NULL,
    DataNasterii date,
    Adresa       varchar(50),
    Salariu      decimal(10) DEFAULT 2800,
    IdSectie     int);
CREATE TABLE SECTII (
    IdSectie     int PRIMARY KEY AUTO_INCREMENT,
    Nume         varchar(50) NOT NULL,
    Buget        decimal(10));
ALTER TABLE ANGAJATI
    ADD CONSTRAINT FOREIGN KEY (IdSectie) REFERENCES SECTII(IdSectie);
```

---

Aceste comenzi sterg mai întâi tabelele `ANGAJATI` și `SECTII` (dacă există), după care le creează conform definiției, iar în tabelul `ANGAJATI` se adaugă în atribut nou (`IdSectie`, care este cheie străină) folosind comanda `ALTARE TABLE`. Se observă modul de introducere a proprietății `AUTO_INCREMENT` pentru cheile primare în cele două tabele și a cheii străine în tabelul `ANGAJATI`.

În MySQL, tabelele unei baze de date se pot afișa cu comanda `SHOW TABLES`, iar structura fiecărui tabel existent se poate afla cu comanda `DESCRIBE nume_tabel`. De exemplu, după execuția comenzii: `source creare_tabele_mysql.sql`, se pot obține următoarele informații despre baza de date curentă:

```
mysql> SHOW TABLES;
```

```
Tables_in_intreprindere
angajati
sectii
```



```
mysql> DESCRIBE ANGAJATI;
```

Field	Type	Null	Key	Default	Extra
IdAngajat	int		PRI	NULL	auto_increment
Nume	varchar(20)				
Prenume	varchar(20)				
DataNasterii	date	YES		NULL	
Adresa	varchar(50)	YES		NULL	
Salariu	decimal(10,0)	YES		NULL	
IdSectie	int	YES		NULL	

Constrângerea de cheie straina (FOREIGN KEY) este acceptata din punct de vedere sintactic în orice instructiune CREATE TABLE sau ALTER TABLE, dar nu are efect de a impune integritatea referentiala decât daca tabelul se creaza de tipul InnoDB. Pentru a înțelege aceasta caracteristica a sistemului MySQL se recomanda studierea manualului de documentatie al produsului.

Pentru crearea tuturor tabelelor bazei de date se poate completa fisierul script creare\_tabele\_mysql.sql.

La introducerea liniilor în tabelele în care cheia primara are proprietatea AUTO\_INCREMENT se poate sa nu se specifice valoarea atributului cheii primare si aceasta este setata automat de catre sistemul de gestiune, cu valoarea urmatoare (incrementata cu 1) fata de cea mai mare valoare a cheii primare existente în tabel. Ca exemplu, se înscriu mai multe linii în tabelele SECTII si ANGAJATI (executând fisierul introducere\_mysql.sql care se gaseste în Capitolul 2 al îndrumarului):

#### Script pentru introducerea datelor în tabele în MYSQL

```
-- Introducere linii in tabelul SECTII
INSERT INTO SECTII (Nume, Buget) VALUES ('Productie', 400000);

-- Introducere linii in tabelul ANGAJATI
INSERT INTO ANGAJATI (Nume,Prenume,IdSectie) VALUES ('Ionescu', 'Ion',1);
INSERT INTO ANGAJATI (Nume,Prenume,IdSectie) VALUES ('Popescu','Petre', 1);
INSERT INTO ANGAJATI (IdAngajat,Nume,Prenume)VALUES(8,'NumeA', 'PrenumeA');
INSERT INTO ANGAJATI (Nume, Prenume) VALUES('NumeB', 'PrenumeB');
INSERT INTO ANGAJATI(Nume,Prenume,IdSectie)
VALUES('Vasilescu', 'Ana',2);
```

Dupa executia acestor comenzi, tabelul SECTII contine un singur tuplu (1, Productie, 400000) iar tabelul ANGAJATI va arata astfel:

IdAngajat	Nume	Prenume	DataNasterii	Adresa	Salariu	IdSectie
1	Ionescu	Ion	NULL	NULL	2800	1
2	Popescu	Petre	NULL	NULL	2800	1
8	NumeA	PrenumeA	NULL	NULL	2800	NULL
9	NumeB	PrenumeB	NULL	NULL	2800	NULL
10	Vasilescu	Ana	NULL	NULL	2800	2

Se poate observa ca sistemul MySQL nu efectueaza nici o verificare a cheilor straine: a fost introdus în tabelul ANGAJATI un tuplu care contine o valoare a cheii straine (valoarea 2) care nu exista în tabelul referit (SECTII). Este evident ca sistemul de gestiune MySQL nu asigura verificarea si impunerea integritatii referentiale, si acest aspect trebuie sa fie avut în vedere la proiectarea aplicatiilor. Pentru mentinerea integritatii referentiale tabelele trebuie sa fie definite de tip InnoDB.



## Exercitii - Capitolul 2

**2.1** Deschideti baza de date MS Access `INTREPRINDERE` cu diagrama din Fig. 2.2 (continuta în fisierul `Intreprindere.mdb` din directorul de aplicatii) si urmariti modul de definire a tabelelor si a asocierilor.

**2.2** Creati aceleasi tabele ca ale bazei de date `INTREPRINDERE` în baza de date (schema) proprie din sistemul SQL Server, Oracle sau MySQL astfel încât sa obtineti aceleasi asocieri ca cele din Fig. 2.2. Se vor folosi fisiere script cu instructiuni SQL (asemanatoare celor prezentate în aceasta lucrare). Pentru sistemul SQL Server realizati si diagrama bazei de date folosind comanda *New Diagram* din *SQL Server Enterprise Manager*.

**2.3** În tabelele create introduceti mai multe linii cu valori ale atributelor cât mai variate folosind fisiere de script cu comenzi `INSERT`. Verificati datele introduse cu comanda `SELECT * FROM nume_tabel;`

**2.4** Scrieti si executati instructiunile SQL pentru selectia si afisarea urmatoarelor date:

- Numele, prenumele, data nasterii si adresa tuturor angajatilor întreprinderii, ordonati dupa nume.
- Numele, prenumele si data nasterii tuturor angajatilor nascuti dupa 1 martie 1970.
- Numele si bugetul tuturor sectiilor întreprinderii.
- Numele si prenumele tuturor clientilor din localitatile Iasi, Timisoara.
- Numele si prenumele tuturor furnizorilor din localitatile Bucuresti, Ploiesti si Craiova.

**2.5** Calculati si afisati urmatoarele date, folosind functii agregat ale limbajului SQL:

- Numarul de salariati ai întreprinderii.
- Salariul mediu, minim si maxim al angajatilor întreprinderii.
- Salariul mediu, minim si maxim pentru diferite categorii (functii) ale salariatilor (cercetator, proiectant etc.)
- Numele si prenumele salariatilor care au salariul mai mare decât salariul mediu din întreprindere.
- Numele si prenumele salariatilor care au salariul mai mare decât salariul mediu pentru functia careia apartin.
- Denumirile functiilor angajatilor din companie si numarul de angajati pentru fiecare functie.
- Bugetul total al întreprinderii (al tuturor sectiilor întreprinderii).

**2.6** Modificati datele stocate în tabele si verificati rezultatul acestor operatii:

- Acordati o marire de salariu cu 10% tuturor angajatilor cu functia `Cercetator` (folosind instructiuni `UPDATE`) si afisati numele, prenumele si noul salariu al angajatilor (folosind instructiuni `SELECT`).
- Acordati o marire de buget cu 10% tuturor sectiilor si afisati denumirea si noul buget al fiecarei sectii.

**2.7\*** Modificati optiunea de definire a cheii straine a relatiei `ANGAJATI` la valoarea `CASCADED` si urmariti comportarea operatiilor de actualizare a tabelor `SECTII` si `ANGAJATI` în fiecare din sistemele de gestiune SQL Server, Oracle sau MySQL.

**2.8\*** Transpuneti în model relational urmatoarea diagrama Entitate-Asociere (pentru o baza de date `PUBLICATII`). Definiti relatiile si asocierile dintre acestea în SQL Server, Oracle sau MySQL.



În relația de asociere **AUTORI-CARTI** prevedeti un atribut care să memoreze ordinea autorilor unei cărți (primul autor, al doilea autor, etc.).

**2.9\*** Transpuneti în model relational următoarea diagramă Entitate-Asociere (pentru o bază de date **MEDICALA**). Definiti relațiile și asocierile dintre acestea în SQL Server, Oracle sau MySQL.

