Capitolul 2
Baze de date relaționale

2.1. Normalizarea unei baze de date relaționale

Pentru a proiecta o bază de date relațională trebuie să avem definită o schemă a acesteia. Schema este formată din relațiile bazei de date. Se determină domeniile pe care vor fi definite relațiile, modul lor de grupare pentru delimitarea fiecărei relații, legăturile logice din interiorul fiecărei relații și dintre relații. Având o schemă a unei relații se definește numele acesteia și mulțimea atributelor acelei relații. Dintr-o schemă a unei relații se definește modul de grupare a datelor și regulile ce vor guverna relația, definind restricțiile datelor și a legăturilor logice dintre date. Legăturile logice dintre relații se definesc prin intermediul domeniilor compatibile care determină atributele comune ale relațiilor. Modul de grupare a domeniilor depinde de complexitatea legăturilor care se definesc între relații. Relațiile ce cuprind prea multe informații specifice pot conduce la probleme redundanță. De exemplu, să presupunem că folosim relația următoare pentru gestionarea vânzărilor unor produse:

Factură(cod_factură, număr, serie, data_facturării, nume_produs, cantitate, preț)

Astfel, pentru fiecare produs care se vinde se generează aceleași date din nou pentru număr, serie și data_facturării. Deci, relația definită nu este potrivită, este o relație ce provoacă redundanță și anomalii la operații de actualizare. De exemplu dacă vindem un produs în cantități separate pe aceeași factură, la o operație de actualizare nu putem distinge cele două înregistrări. Se poate reduce redundanța și anomaliile generate de aceasta prin definirea unei relații noi și reducerea datelor din prima relație:

Factură(cod_factură, număr, serie, data_facturării, cod_produs, cantitate)
Produs(cod_produs, nume_produs, preț)

Constatăm că redundanța a fost înlăturată din relația Factură prin definirea relației Produs, care conține informații legate de produs. Această operație a condus la un nou inconvenient, deoarece dacă dorim aflarea numelor tuturor produselor care au fost vândute pe o anumită factură trebuie să cuplăm cele două relații prin intermediul atributului cod_produs. Operația de cuplare a două relații (join) este operație destul de costisitoare deoarece presupune realizarea produsului cartezian a celor două relații. În teorie, schema bazei de date se realizează pornind de la identificarea tuturor grupurilor de entități și a proprietăților lor. În practică, determinarea atributelor și proprietăților acestora poate fi de multe ori dificilă. În plus, în definirea unei scheme pentru o bază de date relațională trebuie să ținem cost și de faptul că un număr mic de relații poate provoca redundanță, dar și că un număr prea mare de relații crește costul cererilor către către baza de date, deoarece aceste cereri se realizează pe baza operațiilor de cuplare între relații.

În concluzie observăm că apare necesitatea definirii unui procedeu optimizare a schemei bazei date relaționale, pentru a înlătura redundanțele și anomaliile care apar odată cu acestea, dar și pentru definirea legăturilor care există între grupuri de entități. Acest procedeu se numește *Normalizarea relațiilor din baza de date relațională* și presupune pornirea de la o schemă a bazei de date determinată empiric, după care se aplică o serie de transformări succesive asupra schemelor care o alcătuiesc.

Normalizarea se bazează pe formalizarea legăturilor logice (numite dependențe), ce există între atributele unei relații și pe analiza acestora. Dependențele care se definesc între atributele unei relații au fost împărțite în trei categorii:

- dependențe funcționale;
- dependențe multi-valoare;
- dependențe de uniune;

2.2. Forme normale

Existența dependențelor funcționale este naturală. Puteam lua spre exemplu dependența fiecărui atribut față de cheia primară a relației. Dependențele reprezintă informații suplimentare asociate relației, informații ce nu pot fi demonstrate, dar care pot fi verificate

prin confruntarea cu realitatea. Dependențele definite pentru schema unei relații pot fi considerate constrângeri de integritate a datelor, iar implementarea lor în baza de date se face prin normalizarea relației respective.

Procesul prin care se ajunge la o schemă optimă a bazei de date, din punctul de vedere al modelării legăturilor existente în cadrul relațiilor și pentru reducerea redundanței reprezintă normalizarea relațiilor din baza de date. Se începe cu identificarea atributelor relației care se normalizează și a dependențelor în care există aceste atribute, în funcție de natura dependențelor care au fost identificate se determină forma normală în care se află relația respectivă. Prin aplicarea unui set de reguli de normalizarea, specifice acelei forme normale, se transformă relația care a fost identificată într-un set de relații echivalente, care aparțin unei forme normale mai superioare. Normalizarea este un proces iterativ, fiecare dependență identificată trebuie să treacă prin aceste faze, pentru a se ajunge la ultima formă normală.

Normalizarea bazei de date presupune satisfacerea condițiilor de formă normală:

- prima formă normală (1NF – First Normal Form);
- a doua formă normală (2NF – Second Normal Form);
- a treia formă normală (3NF – Third Normal Form);

Prima formă normală exclude posibilitatea existenței grupurilor repetitive, cerând ca fiecare atribut într-o bază de date să cuprindă numai o valoare atomică. De asemenea, prima formă normală presupune și ca fiecare înregistrare să fie definită astfel încât să fie identificată în mod unic prin intermediul unei chei primare.
Exemplu de încălcare a primei forme normale:

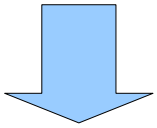
Persoană	Filme închiriate
Daniel	Titanic, Filantropica, Pinocchio
Ovidiu	Tăcerea mieilor, Pinocchio
Ionel	Filantropica, Cenușăreasa

Tabelul 2.1. *Exemplu de tabel ce încalcă prima formă normală*

Problema cu tabela exemplificată în **Tabelul 2.1.** este că la interogarea devine foarte încurcată. De exemplu dacă dorim să aflăm care sunt persoanele ce au închiriat filmul ”Titanic” și ”Pinocchio” trebuie să parcurgem fiecare șir, să-l împărțim în subșiruri și să

vedem dacă apare în vreun subșir titlurile căutate. După care trebuie selectate doar acele înregistrări ce conțin ambele titluri. O soluție de satisfacere a primei forme normale reprezintă introducerea unei noi înregistrări pentru fiecare film.
Astfel, din prima formă a tabelei obținem:

Persoană	Filme închiriate
Daniel	Titanic, Filantropica, Pinocchio
Ovidiu	Tăcerea mieilor, Pinocchio
Ionel	Filantropica, Cenușăreasa



ID	Persoană	Filme închiriate
1	Daniel	Titanic
2	Daniel	Filantropica
3	Daniel	Pinocchio
4	Ovidiu	Tăcerea mieilor
5	Ovidiu	Pinocchio
6	Ionel	Filantropica
7	Ionel	Cenușăreasa

Tabelul 2.2. *Transformarea unei tabele pentru a satisface prima formă normală*

A doua formă normală cere ca toate atributele să fie dependente funcțional de totalitatea cheii primare. Dacă unul sau mai multe atribute sunt dependente funcțional numai de o parte a cheii primare, atunci ele trebuie să fie în tabele separate. Dacă o tabelă are o cheie primară formată numai din un atribut, atunci ea este automat în a doua formă normală.
Exemplu care încalcă a doua formă normală:

ID_comandă	ID_produs	nume_produs	cantitate
12	1	cereale	20
13	2	ciocolată	7

Tabelul 2.3. *Tabela comandă ce încalcă a doua formă normală*

În cazul tabelei ”Comandă” exemplificată în **Tabelul 2.3** cheia primară este o cheie

compusă, în alcătuirea ei intră atât *ID_comandă*, cât și *ID_produs*. Observăm că *nume_produs* nu depinde de *ID_comandă*, ea depinde doar de *ID_comandă*. Pentru a fi în a doua formă normală, tabela ”Comandă” trebuie modificată ca în **Tabelul 2.4** de mai jos.

ID_comandă	ID_produs	nume_produs	cantitate
12	1	cereale	20
13	2	ciocolată	7

ID_produs	nume_produs
1	cereale
2	ciocolată

ID_comandă	ID_produs	cantitate
12	1	20
13	2	7

Tabelul 2.4. *Transformarea unei tabele pentru a satisface a doua formă normală*

A treia formă normală presupune că toate atributele non-cheie trebuie să fie mutual independente. Relația *depinde de cheie* este bazată în întregime de cheie și de nimic altceva.

Piesă	nume_producător	adresă_producător
1245	Boss	București
5487	Huawei	Craiova
9541	Boss	București

Tabela 2.5. *Exemplu de tabelă ce încalcă a treia formă normală*

Tabela 2.5 poate fi reorganizată ca în **Tabela 2.6** pentru a satisface a treia formă normală.

Piesă	nume_producător
1245	Boss
5487	Huawei
9541	Boss

nume_producător	adresă_producător
Boss	București
Huawei	Craiova

Tabela 2.6. *Transformarea* **Tabelei 2.5** *astfel încât să satisfacă a treia formă normală*

2.3. Normalitatea bazei de date a aplicației

Baza de date a aplicației, ilustrată în **Figura 2.1** este organizată astfel: avem facturi, produse, firme, stoc și plăți. Aceste patru componente trebuie organizate astfel încât să permită o evidență clară a produselor din stoc, a plăților, firmelor.

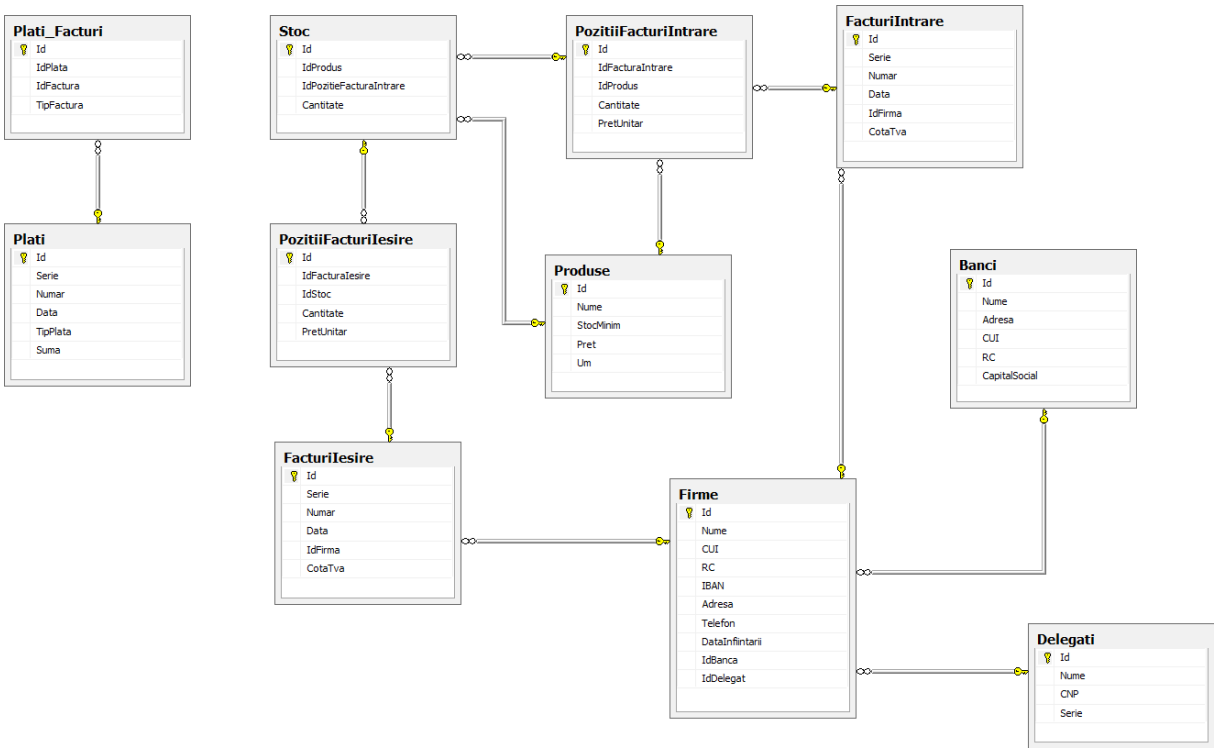


Figura 2.1. *Vedere de ansamblu a bazei de date a aplicației*

Această structurare a relațiilor din baza de date a condus la evidențierea a trei părți strâns legate între ele.

Prima parte are ca piesă centrală *Firmele* și dependențele sale (cum se vede în **Figura 2.2**). Tabela firmelor depinde de tabelele *Bănci* și *Delegați*, dar de ea depind și facturile.

Tabela *Delegați* din **Figura 2.2** prezintă următoarea relație:

Delegați(**Id**, Nume, CNP, Serie)

Această relație conține atribute care sunt legate strict de cheia sa. Nu avem grup compus, iar relația este strâns legată de cheie. Această tabelă este în a treia formă normală.

Tabela *Bănci* din **Figura 2.2** are relația:

Bănci(**Id**, Nume, Adresă, CUI, RC, Capital social)

Această relație nu conține grupuri compuse, toate atributele sunt strâns legate de cheie. Nu avem anomalii, nici redundanță. Această tabelă satisface condițiile celei de-a treia formă normale.

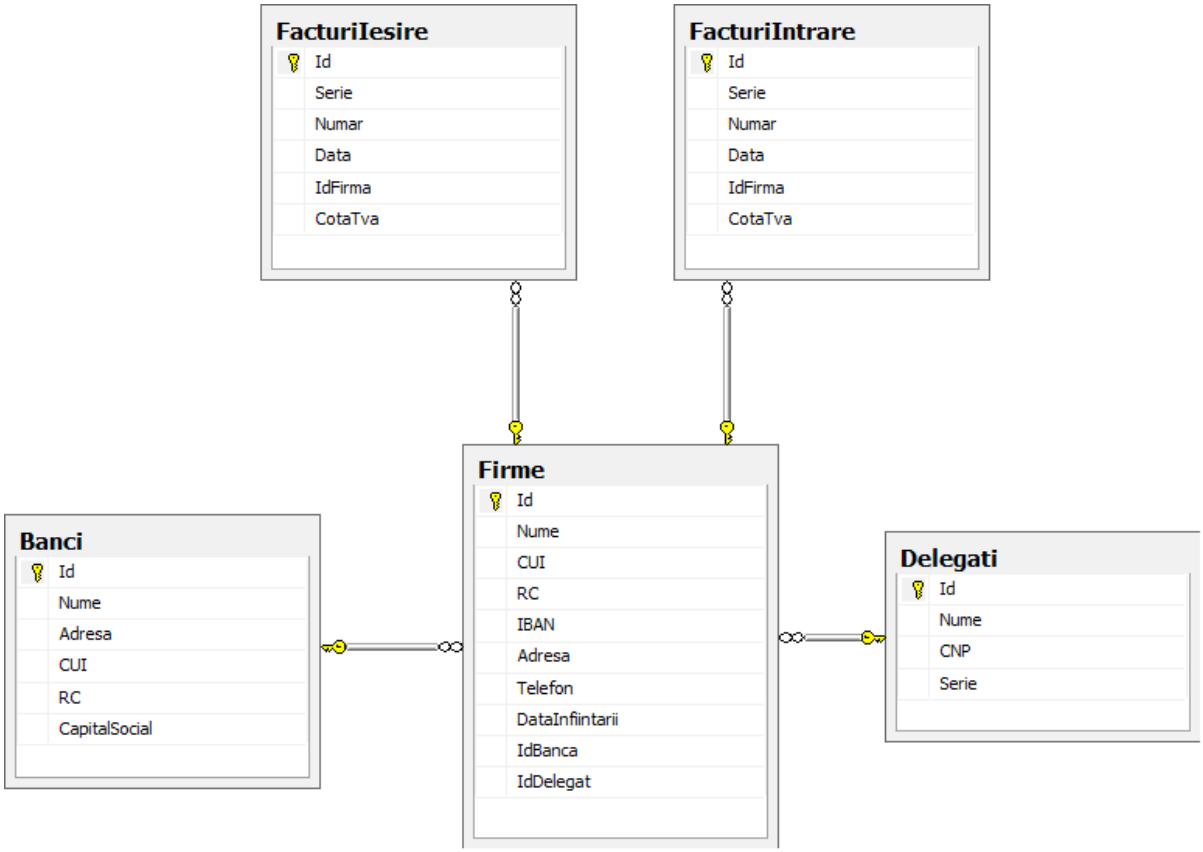


Figura 2.2. *Tabela firmelor și dependențele sale*

Tabela *Firme* din **Figura 2.2** este caracterizată de următoarea relație:

Firme(**Id**, Nume, CUI, RC, IBAN, Adresă, Telefon, DataÎnființării, *IdBancă*, *IdDelegat*)

Această relație nu conține grupuri compuse, toate atributele sunt strâns legate de cheie. Avem două atribute: *IdBancă* și *IdDelegat*, care sunt necesare pentru operația de cuplare cu tabelele *Bănci* și *Delegați*.

În schema reprezentată în **Figura 2.2** mai avem două tabele pentru reținerea datelor necesare facturilor de intrare. Cele două tabele se bazează pe relația:

Facturi(**Id**, Serie, Număr, Data, *IdFirmă*, CotaTva)

Această relație satisface la rândul ei condițiile formei normale trei, nu conține grupuri

compuse, iar toate atributele sunt strâns legate de cheie. Atributul *IdFirmă* este un atribut necesar pentru operația de cuplare cu tabela *Firme* din **Figura 2.2**, pentru a afla datele despre firma căreia i s-a eliberat o factură.

A doua parte se concentrează asupra gestionării stocului, a produselor vândute și a produselor cumpărate. Această gestionare este facilitată de două relații importante care descriu produsele ce au ieșit din stoc, cât și cele ce au intrat în stoc. Aceste relații, care se pot vedea în **Figura 2.3**, sunt următoarele:

PoziţiiFacturiIntrare(**Id**, *IdFacturăIntrare*, *IdProdus*, Cantitate, PreţUnitar)

PoziţiiFacturiIeşire(**Id**, *IdFacturăIeşire*, *IdStoc*, Cantitate, PreţUnitar)

Aceste relații sunt ambele în formă normală trei, având doar două atribute ce descriu cantitatea și prețul unitar al produsul; mai conțin și câte două atribute care sunt necesare pentru operațiile de cuplare cu tabelele *FacturiIntrare*, *FacturiIeșire*, *Produse* și *Stoc*.

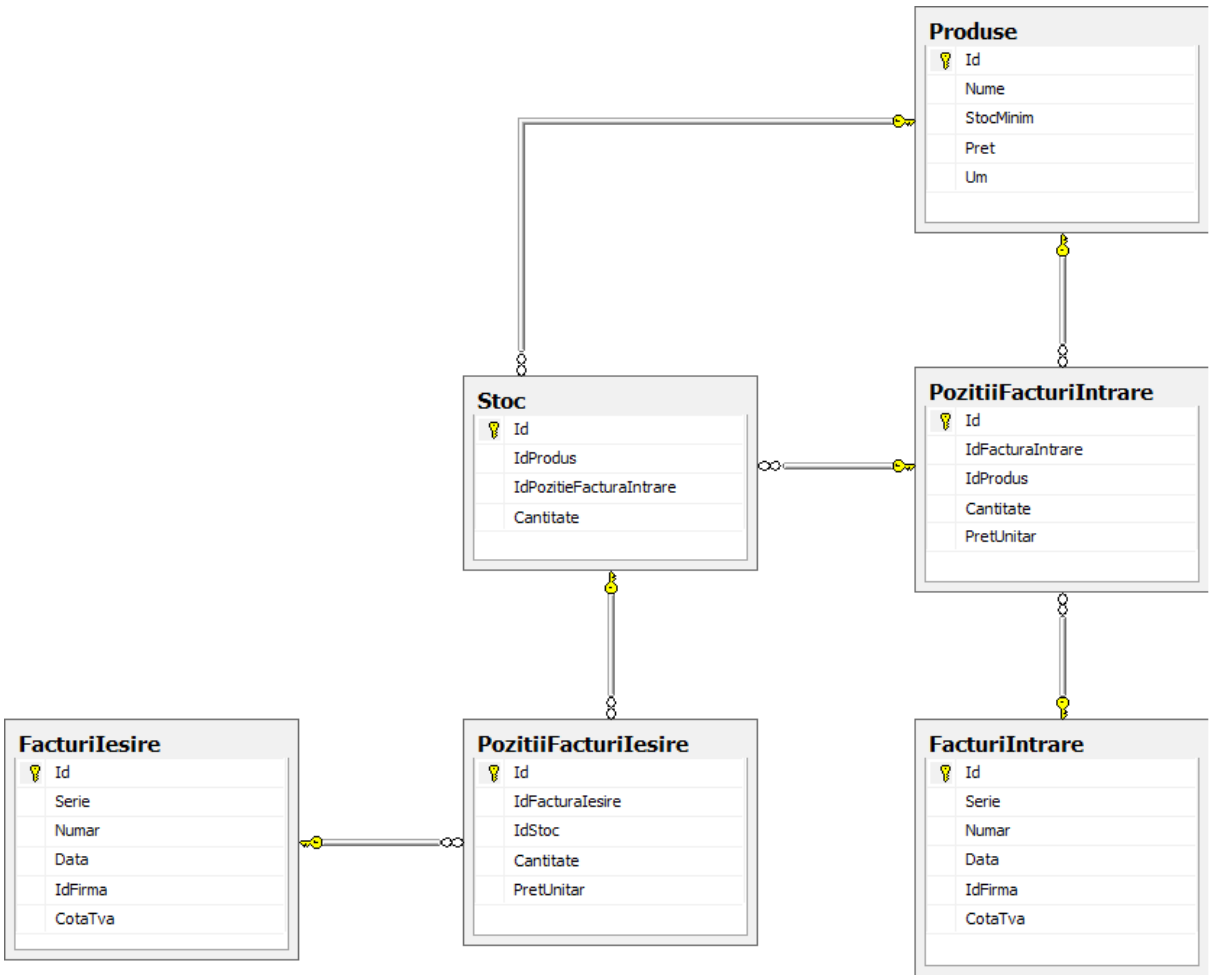


Figura 2.3. *Tabelele care se ocupă cu gestionarea stocurilor*

În schema din **Figura 2.3** introducem două tabele noi: *Produse* și *Stoc*. Tabela produselor reține date despre produsele care se comercializează, iar tabela stocurilor ține evidența acestor produse, din interogarea sa putem deduce dacă un produs este disponibil sau nu, iar dacă este disponibil atunci aflăm și cantitatea disponibilă.

Relația tabelei *Produse* este:

Produse(**Id**, Nume, StocMinim, Preț, UM)

Această relație este simplă, are patru atribute ce depind exclusiv de cheie, nu prezintă grupuri compuse, ea satisface condițiile formei normale trei.

Tabela *Stoc* este reprezentată de relația:

Stoc(**Id**, *IdProdus, IdPoziţieFacturăIntrare*, Cantitate)

Relația nu conține decât un singur atribut, care reprezintă cantitatea produsului din stoc. Mai avem două atribute, necesare pentru operațiile de cuplare, care indică produsul și poziția din factura de intrare cu care a fost adăugat produsul respectiv în stoc.

A treia componentă a bazei de date, ilustrată în **Figura 2.4**, se are în vedere detaliile legate de achitarea facturilor. Ea aduce nou o tabelă în care se reţin datele plăților efectuate și o tabelă de legătură, care permite cuplarea plăților cu facturile.

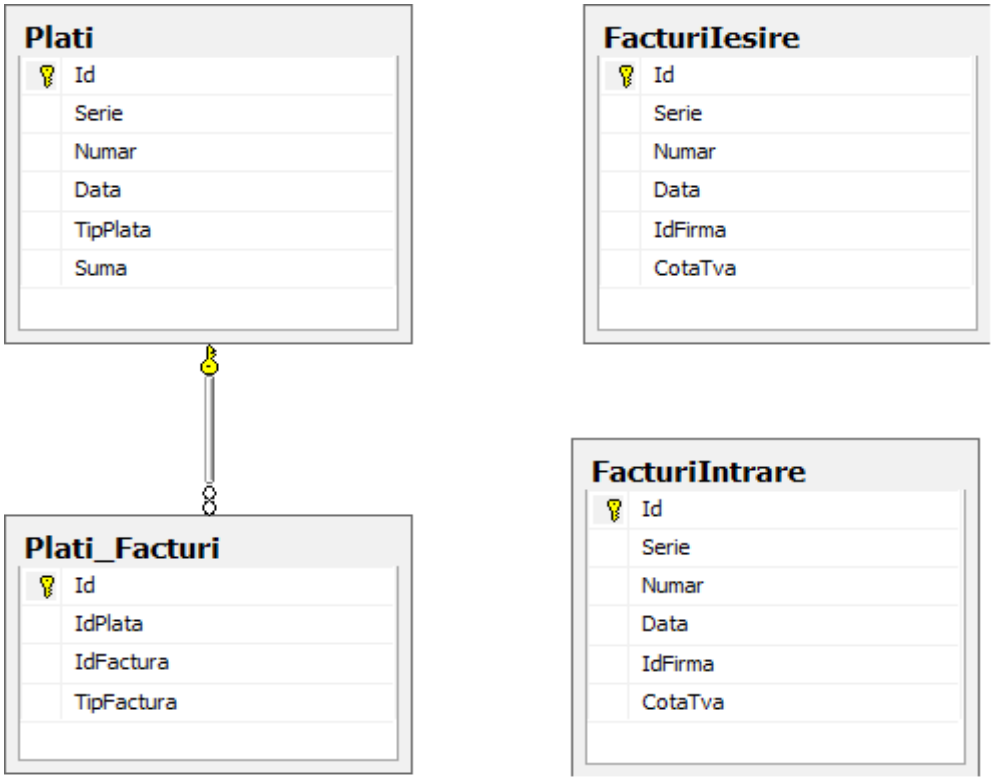


Figura 2.4. *Tabelele care se ocupă cu plățile*

Tabela *Plăți* are următoarea relație:

Plăți(**Id**, Serie, Număr, Data, TipPlată, Sumă)

Relația Plăți conține atribute care sunt strict legate de cheie, nu conține grupuri compuse, în concluzie satisface necesitățile pentru a fi în forma normală trei.

Tabela *Plăți_Facturi* este o tabelă intermediară între plăți și facturi, are relația:

Plăți_Facturi(**Id**, *IdPlată*, *IdFactură*, TipFactură)

Această relație conține un sigur atribut care sugerează tipul facturii care este plătită și două atribute necesare pentru operațiile de cuplare cu tabelele *Plăți, FacturiIntrare* și *FacturiIeșire* pentru a putea afla care sunt facturile plătite, facturile neplătite, cât mai trebuie plătit pentru o factură și cât s-a plătit pentru o factură. Toate atributele sunt strâns legate de plăți pe facturi, iar având doar un atribut în cheie, relația *Plăți_Facturi* satisface condițiile pentru forma normală trei.

Aplicația mai conține și o componentă ce adaugă o facilitate de autentificare pentru folosirea aplicației. Această componentă presupune o autentificare înainte de folosirea aplicației, adăugare și modificarea drepturilor pentru fiecare utilizator creat și existența unui utilizator care este administrator pe întreaga aplicație și care nu poate fi șters.

Schema componentei (ilustrată în **Figura 2.5**) are trei tabele necesare pentru utilizatori, roluri și drepturile existente în aplicație. Mai există o tabelă de legătură între roluri și drepturi și o tabelă care nu este dependentă de alte tabele în care se reține parola administratorului și firma curentă a aplicației (necesară la eliberarea de facturi).

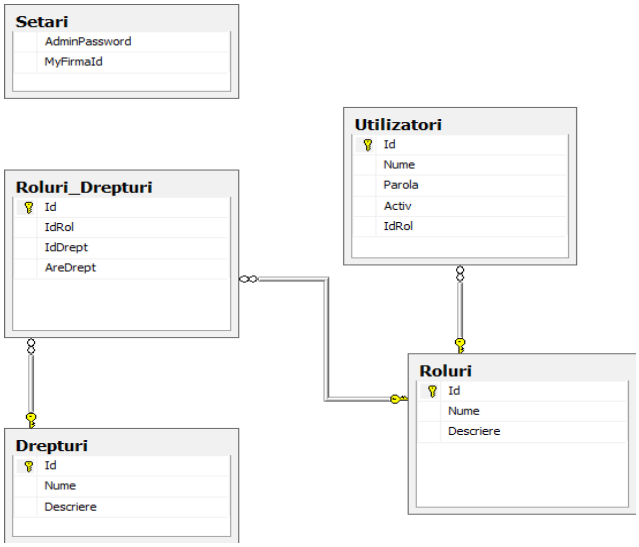


Figura 2.5 *Tabelele care se ocupă cu partea de administrare*

Capitolul 2 – Baze de date relaționale

Tabela *Utilizatori* este reprezentată de relația:

Utilizatori(**Id**, Nume, Parolă, Activ, *IdRol*)

Această relație se află în formă normală trei, deoarece nu conține grupuri compuse, iar toate atributele sunt strâns legate de cheie. Atributul *IdRol* este necesar pentru operațiile de cuplare cu tabela de roluri.

Tabela *Roluri* este reprezentată de relația:

Roluri(**Id**, Nume, Descriere)

Această tabelă este evident în a treia formă normală deoarece conține două atribute care au legătură exclusiv cu cheia relației.

Tabela *Drepturi* este reprezentată de relația:

Drepturi(**Id**, Nume, Descriere)

Tabela este în a treia formă normală deoarece conține două atribute care au legătură exclusiv cu cheia relației.

Tabela *Roluri_Drepturi* este reprezentată de relația:

Roluri_Drepturi(**Id**, *IdRol*, *IdDrept,* AreDrept)

Tabela este în a treia formă normală deoarece un sigur atribut care se află înlegătură strânsă cu cheia relației și două atribute necesare pentru cuplarea cu tabelele de roluri și drepturi.

Tabela *Setări* conține două atribute care reprezintă parola curent a administratorului și firma care este setată ca fiind firma curentă a aplicației.