UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

PROIECT LA BAZE DE DATE

PROFESOR COORDINATOR:

VASILE SILVIU-LAURENȚIU

STUDENT:

MAXIM TIBERIU

BUCUREȘTI

2020

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

MANAGEMENTUL SPITALELOR

PROFESOR COORDINATOR:

VASILE SILVIU-LAURENȚIU

STUDENT:

MAXIM TIBERIU

BUCUREȘTI

2020

CUPRINS

1. Prezentarea modelului

Tema aleasă pentru proiectul la disciplina Baze de Date este MANAGEMENTUL SPITALELOR. Voi considera faptul că proiectul face referire la o bază de date la nivel național, care integrează spitale și pacienți din diferite județe.

Pentru un spital, o bază de date structurată corect, înseamnă enorm. Astfel, se reduce probabilitatea de a introduce erori în baza de date, ceea ce, în unele cazuri, poate duce la probleme grave pentru pacienți.

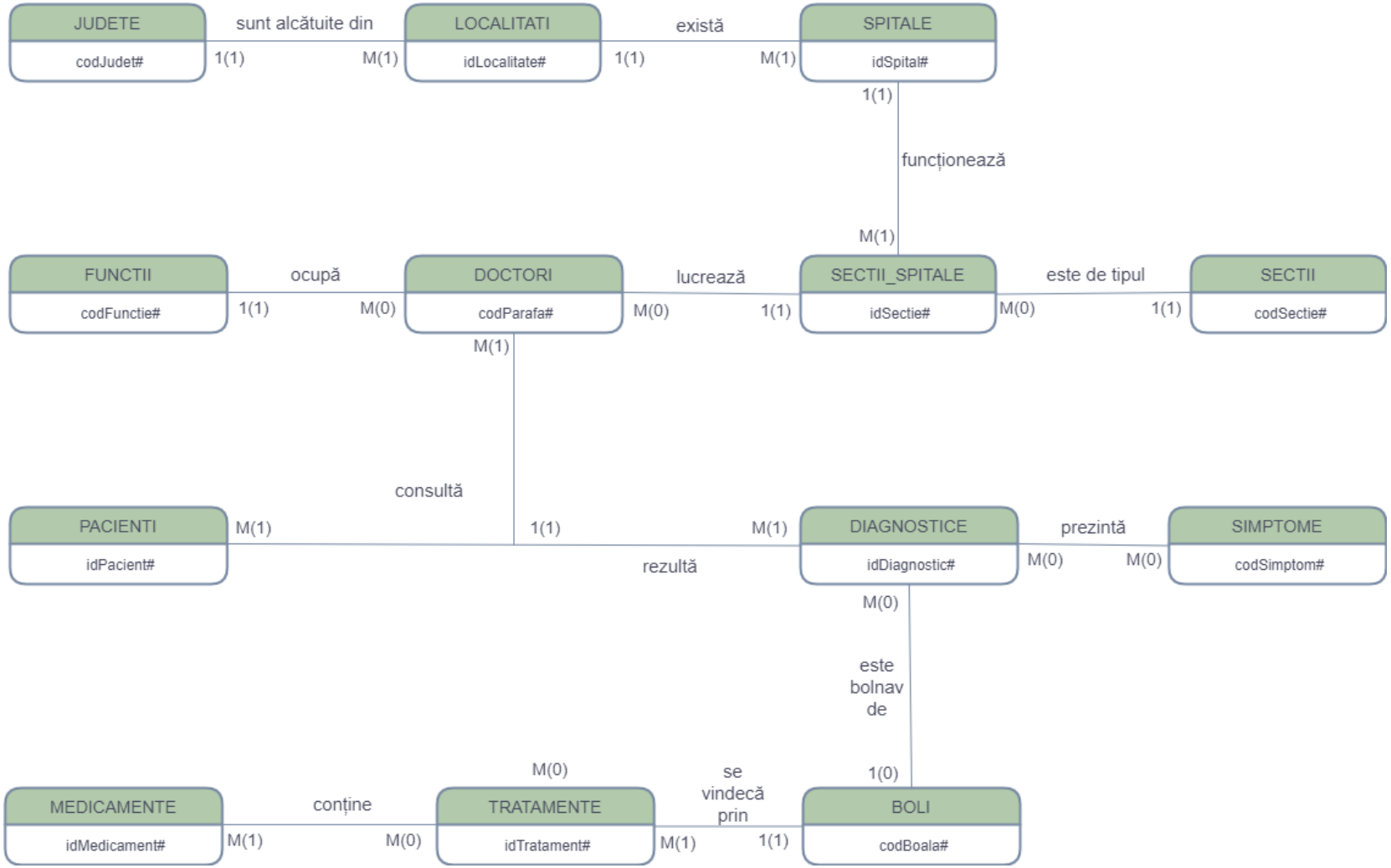
Din acest motiv, am realizat o bază de date ce are ca scop gestionarea datelor pacienților internați în spitale. Aceasta stochează câteva date administrative (date despre spitale, secțiile existente în acestea, medicii care lucrează în ele), date despre pacienți, consultațiile acestora, diagnosticele și tratamentele pentru respectivul diagnostic.

2. Regulile modelului

- Există cel puțin un județ în baza de date.
- Fiecare județ are cel puțin o localitate în care există un spital.
- Într-o localitate prezentă în baza de date, există cel puțin un spital.
- Considerăm că fiecare spital are cel puțin o secție funcțională.
- Există cel puțin un tip de secție.
- Pot exista secții fără medici. (ex.: secție închisă, secție în renovare, etc.)
- Există cel puțin un tip de funcție pe care o pot avea medicii.
- Considerăm faptul că pacientul nu este tratat în ambulatoriu și că acesta trebuie să rămână internat din cauza a minim un diagnostic.
- Pentru fiecare pacient, va exista cel puțin un diagnostic care necesită internarea, diagnostic rezultat în urma unei consultații. Pot exista pacienți care vor fi consultați de mai mulți doctori, dar la un anumit doctor, în urma consultației, să fie sănătos.
- Există cel puțin un simptom pe care îl pot avea pacienții.
- Există cel puțin o boală pe care o pot avea pacienții.
- Considerăm că fiecare boală are nevoie de tratament.
- Există cel puțin un medicament disponibil pentru administrarea în tratamente.

3. Diagrama Entitate-Relație

3.1. Reprezentarea diagramei



3.2. Descrierea entităților, atributelor, cheilor, relațiilor și a cardinalităților

3.2.1. Descrierea entităților, atributelor și a cheilor

3.2.1.1. Tabelul JUDETE

Tabelul JUDETE stochează datele elementare despre județele din baza de date. Structura tabelului JUDETE este:

Cheie	Denumire atribut	Descriere
PK	codJudet	identificatorul unic al județului
	numeJudet	numele județului

3.2.1.2. Tabelul LOCALITATI

Tabelul LOCALITATI stochează datele despre localitățile care au cel puțin un spital. Structura tabelului LOCALITATI este:

Cheie	Denumire atribut	Descriere
PK	idLocalitate	identificatorul unic al localității
	numeLocalitate	numele localității
	codPostal	codul poștal al localității
FK	codJudet	codul județului de care aparține localitatea

3.2.1.3. Tabelul SPITALE

Tabelul SPITALE memorează adresa spitalului și datele de contact ale acestuia. Fiecare spital din țară va avea un cod unic de identificare. Structura tabelului SPITALE este:

Cheie	Denumire atribut	Descriere
PK	idSpital	identificatorul unic al spitalului
	numeSpital	numele spitalului
FK	idLocalitate	identificatorul unic al localității în care este amplasat spitalul
	strada	adresă
	numar	adresă
	telefon	date de contact
	email	date de contact

3.2.1.4. Tabelul SECTII

Tabelul SECTII stochează tipurile de secții care pot exista într-un spital. Structura tabelului SECTII este:

Cheie	Denumire atribut	Descriere
PK	codSectie	codul prin care se identifică un tip de secție
	numeSectie	numele secției

3.2.1.5. Tabelul SECTII_SPITALE

Tabelul SECTII_SPITALE stochează date despre secțiile existente în fiecare spital. Acestea sunt identificate printr-un cod unic de identificare. Structura tabelului SECTII_SPITALE este:

Cheie	Denumire atribut	Descriere
PK	idSectie	identificatorul unic al secției
FK	idSpital	ID-ul spitalului în care este amplasată secția
FK	codSectie	tipul secției
	corp	locație
	etaj	locație
	telefon	date de contact
	email	date de contact

3.2.1.6. Tabelul FUNCTII

Tabelul FUNCTII stochează date despre funcțiile pe care le pot ocupa medicii. Structura tabelului FUNCTII este:

Cheie	Denumire atribut	Descriere
PK	codFunctie	identificatorul unic al funcției
	numeFunctie	numele funcției

3.2.1.7. Tabelul DOCTORI

Tabelul DOCTORI stochează datele despre toți doctorii care lucrează în spitalele existente în baza de date. Structura tabelului DOCTORI este:

Cheie	Denumire atribut	Descriere
PK	codParafa	identificatorul unic al fiecărui medic
	nume	date personale
	prenume	date personale
	CNP	date personale
	telefon	date de contact
	email	date de contact
FK	codFunctie	funcția ocupată
FK	idSectie	secția în care lucrează

3.2.1.8. Tabelul PACIENTI

Tabelul PACIENTI stochează datele pacienților internați în spital sau ale pacienților vechi, externați. Structura tabelului PACIENTI este:

Cheie	Denumire atribut	Descriere
PK	idPacient	identificatorul unic al pacientului
	nume	numele pacientului
	prenume	prenumele pacientului
	CNP	CNP-ul pacientului
	strada	adresa
	numar	adresa
	localitate	adresa
	telefon	date de contact
	email	date de contact
	asigurat	verificare asigurare pacient
	dataInternare	data internării
	dataExternare	data externării

3.2.1.9. Tabelul BOLI

Tabelul BOLI stochează doar tipul bolii pe care un pacient o poate avea. Structura tabelului BOLI este:

Cheie	Denumire atribut	Descriere
PK	codBoala	identificatorul unic al bolii
	denumireBoala	denumirea bolii

3.2.1.10. Tabelul DIAGNOSTICE

Tabelul DIAGNOSTICE stochează informațiile cu privire la pacienții diagnosticați, în urma unei consultații. Structura tabelului DIAGNOSTICE este:

Cheie	Denumire atribut	Descriere
PK	idDiagnostic	identificatorul unic al diagnosticului
FK	idConsultatie	consultația din care a rezultat acest diagnostic
FK	codBoala	boala de care suferă pacientul

3.2.1.11. Tabelul SIMPTOME

Tabelul SIMPTOME stochează informațiile cu privire la simptomele care sunt prezente în cazul în care pacientul este diagnosticat cu o boală. Structura tabelului SIMPTOME este:

Cheie	Denumire atribut	Descriere
PK	codSimptom	identificatorul unic al simptomului
	denumireSimptom	denumirea simptomului

3.2.1.12. Tabelul TRATAMENTE

Tabelul TRATAMENTE stochează informațiile cu privire la tratamentele care se pot administra în procesul de vindecare al unui pacient care suferă de o boală existentă în baza de date. Structura tabelului TRATAMENTE este:

Cheie	Denumire atribut	Descriere
PK	idTratament	identificatorul unic al tratamentului
FK	codBoala	boala pentru care se administrează tratamentul
	perioadaAdministrare	perioada de administrare a tratamentului
	indicatii	indicații, dacă este cazul

3.2.1.13. Tabelul MEDICAMENTE

Tabelul MEDICAMENTE stochează informații despre medicamentele folosite în diverse tratamente. Structura tabelului MEDICAMENTE este:

Cheie	Denumire atribut	Descriere
PK	idMedicamente	identificatorul unic al diagnosticului
	numeMedicament	numele medicamentului
	tipMedicament	tipul de medicament (antibiotic, analgezic, etc.)
	dozaUnitate	doza per unitate (250/500/etc. mg)

3.2.2. Descrierea relațiilor și a cardinalităților

3.2.2.1. JUDETE – LOCALITATI

Relația: *JUDETELE* **sunt alcătuite din** *LOCALITATI.*

Cardinalități:

- **Cardinalitate maximală**
  - Câte localități aparțin de un județ? => MULTE
  - De câte județe aparține o localitate? => 1
- **Cardinalitate minimală**
  - Câte localități trebuie să aibă un județ? => 1
  - De câte județe trebuie să aparțină o localitate? => 1

3.2.2.2. LOCALITATI – SPITALE

Relația: În *LOCALITATI* **există** *SPITALE*.

Cardinalități:

- **Cardinalitate maximală**
    - Câte spitale pot să existe într-o localitate? => MULTE
    - De câte localități poate să aparțină un spital? => 1
- **Cardinalitate minimală**
    - Câte spitale trebuie să existe într-o localitate? => 1
    - De câte localități trebuie să aparțină un spital? => 1

3.2.2.3. SPITALE – SECTII_SPITALE

Relația: În *SPITALE* **funcționează** *SECTII_SPITALE*.

Cardinalități:

- **Cardinalitate maximală**
    - Câte secții_spitale poate avea un spital? => MULTE
    - În câte spitale poate exista o secție_spital? => 1
- **Cardinalitate minimală**
    - Câte secții_spitale trebuie să aibă un spital? => 1
    - În câte spitale poate exista o secție_spital? => 1

3.2.2.4. SECTII_SPITALE – SECTII

Relația: *SECTII_SPITALE* **sunt de tipul** *SECTII*.

Cardinalități:

- **Cardinalitate maximală**
    - De câte tipuri de secții poate fi o secție_spital? => 1
    - Câte secții_spitale pot avea același tip de secție? => MULTE
- **Cardinalitate minimală**
    - De câte tipuri de secții trebuie să fie o secție_spital? => 1
    - Câte secții_spitale trebuie să aibă același tip de secție? => 0

3.2.2.5. SECTII_SPITALE – DOCTORI

Relația: În *SECTII_SPITALE* **lucrează** *DOCTORI*.

Cardinalități:

- **Cardinalitate maximală**
  - Câți doctori poate avea o secție_spital? => MULȚI
  - În câte secții_spital poate lucra un doctor? => 1
- **Cardinalitate minimală**
  - Câți doctori trebuie să aibă o secție_spital? => 0
  - În câte secții_spital trebuie să lucreze un doctor? => 1

3.2.2.6. DOCTORI – FUNCTII

Relația: *DOCTORI* **ocupă** *FUNCTII*.

Cardinalități:

- **Cardinalitate maximală**
  - Câte funcții poate avea un doctor? => 1
  - Câți doctori pot avea aceeași funcție? => MULȚI
- **Cardinalitate minimală**
  - Câte funcții trebuie să aibă un doctor? => 1
  - Câți doctori trebuie să aibă aceeași funcție? => 0

3.2.2.7. DOCTORI – PACIENTI

Relația: *DOCTORI* **consultă** *FUNCTII*.

Cardinalități:

- **Cardinalitate maximală**
  - Câți doctori pot consulta un pacient? => MULȚI
  - Câți pacienți pot fi consultați de același doctor? => MULȚI
- **Cardinalitate minimală**
  - Câți doctori trebuie să consulte un pacient? => 1
  - Câți pacienți trebuie să fie consultați de același doctor? => 0

NOTĂ! Fiind o relație M:N va rezulta un tabel intermediar al consultațiilor dintre medici și pacienți. Relația „rezultă” (n.r. diagnostic) se va face între tabelul asociativ CONSULTATII și tabelul DIAGNOSTICE.

3.2.2.8. CONSULTATII – DIAGNOSTICE

Relația: Din *CONSULTATII* **rezultă** *DIAGNOSTICE*.

Cardinalități:

- **Cardinalitate maximală**
    - În urma unei consultații, câte diagnostice pot rezulta? => MULTE
        - ex.: medic cardiolog, consultație => suferă de hipertensiune și anevrism
    - De câte consultații aparține un diagnostic? => 1

- **Cardinalitate minimală**
    - În urma unei consultații, câte diagnostice trebuie să rezulte? => 1
    - De câte consultații trebuie să aparțină un diagnostic? => 1

3.2.2.9. DIAGNOSTICE – SIMPTOME

Relația: *DIAGNOSTICE* **prezintă** *SIMPTOME*.

Cardinalități:

- **Cardinalitate maximală**
    - Câte simptome pot corespunde unui diagnostic? => MULTE
    - Câte diagnostice pot avea aceleași simptome? => MULTE
- **Cardinalitate minimală**
    - Câte simptome trebuie să corespundă unui diagnostic? => 0
        - ex.: pacientul este SĂNĂTOS (codBoala = null)
    - Câte diagnostice trebuie să aibă aceleași simptome? => 0

3.2.2.10. DIAGNOSTICE – BOLI

Relația: Prin *DIAGNOSTICE* rezultă că pacientul **este bolnav de** *BOLI*.

Cardinalități:

- **Cardinalitate maximală**
    - Câte boli pot să reiasă dintr-un diagnostic? => 1
    - Câte diagnostice pot avea aceleași boală? => MULTE
- **Cardinalitate minimală**
    - Câte boli trebuie să reiasă dintr-un diagnostic? => 0
        - ex.: pacientul este SĂNĂTOS (codBoala = null)
    - Câte diagnostice trebuie să aibă aceleași boală? => 0

3.2.2.11. BOLI – TRATAMENTE

Relația: *BOLI* **se vindecă prin** *TRATAMENTE*.

Cardinalități:

- **Cardinalitate maximală**
    - Câte tratamente poate avea o boală? => MULTE
    - La câte boli face referință un tratament? => 1
- **Cardinalitate minimală**
    - Câte tratamente trebuie să aibă o boală? => 0
        - ex.: pacientul este SĂNĂTOS (codBoala = null)
    - La câte boli trebuie să facă referință un tratament? => 1

3.2.2.12. TRATAMENTE – MEDICAMENTE

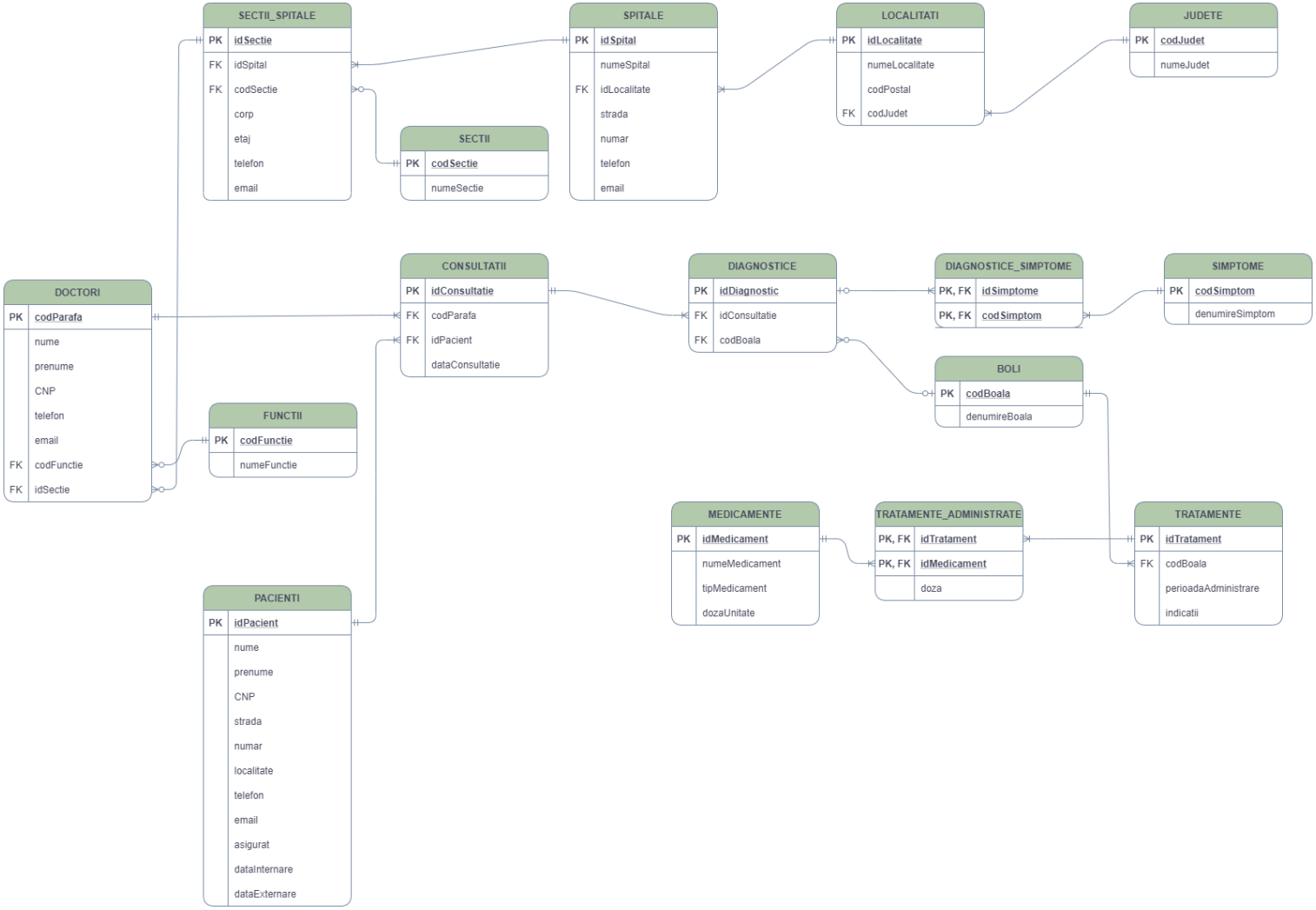
Relația: *TRATAMENTE* **conțin** *MEDICAMENTE*.

Cardinalităţi:

- **Cardinalitate maximală**
  - Câte medicamente poate conține un tratament? => MULTE
  - De câte tratamente poate depinde un medicament? => MULTE
- **Cardinalitate minimală**
  - Câte medicamente trebuie să conțină un tratament? => 1
  - De câte tratamente trebuie să depindă un medicament? => 0

4. Diagrama Conceptuală

4.1. Reprezentare diagramă



4.2. Descrierea constrângerilor de integritate

4.2.1. Tabelul JUDETE

- **Constrângeri de tip PRIMARY KEY**
  - JUDETE_codJudet_PK, pentru atributul codJudet;
- **Constrângeri de tip NOT NULL**
  - JUDETE_numeJudet_NN, pentru atributul numeJudet;
    - *județul trebuie să aibă obligatoriu un nume.*
- **Constrângeri de tip UNIQUE**
  - JUDETE_numeJudet_U, pentru atributul numeJudet.
    - *numele județelor trebuie să fie diferite.*

4.2.2. Tabelul LOCALITATI

- **Constrângeri de tip PRIMARY KEY**
  - LOCALITATI_idLocalitate_PK, pentru atributul idLocalitate;
- **Constrângeri de tip FOREIGN KEY**
  - LOCALITATI_codJudet_FK, pentru atributul codJudet;
    - *face legătura cu tabelul JUDETE (codJudet);*
    - *determină județul din care face parte localitatea.*
- **Constrângeri de tip NOT NULL**
  - LOCALITATI_numeLocalitate_NN, pentru atributul numeLocalitate;
    - *localitatea trebuie să aibă obligatoriu un nume.*
  - LOCALITATI_codPostal_NN, pentru atributul codPostal;
    - *codul poștal face parte din adresa spitalului, deci este obligatoriu.*
  - LOCALITATI_codJudet_NN, pentru atributul codJudet;
    - *localitatea trebuie să aparțină unui județ.*
- **Constrângeri de tip UNIQUE**
  - LOCALITATI_codPostal_U, pentru atributul codPostal.
    - *codurile poștale sunt unice pentru fiecare localitate.*

4.2.3. Tabelul SPITALE

- **Constrângeri de tip PRIMARY KEY**
  - SPITALE_idSpital_PK, pentru atributul idSpital;
- **Constrângeri de tip FOREIGN KEY**
  - SPITALE_idLocalitate_FK, pentru atributul idLocalitate;
    - *face legătura cu tabelul LOCALITATI (idLocalitate);*
    - *determină localitatea de care aparține spitalul.*
- **Constrângeri de tip NOT NULL**
  - SPITALE_numeSpital_NN, pentru atributul numeSpital;
    - *este important ca un spital să aibă un nume, pentru o identificare ușoară.*
  - SPITALE_idLocalitate_NN, pentru atributul idLocalitate;
    - *spitalul trebuie să aparțină de o localitate.*
  - SPITALE_strada_NN, pentru atributul strada;
    - *strada face parte din adresa spitalului, deci este obligatorie.*
  - SPITALE_numar_NN, pentru atributul numar;
    - *numărul face parte din adresa spitalului, deci este obligatoriu.*
  - SPITALE_telefon_NN, pentru atributul telefon;
    - *datele de contact sunt obligatorii.*
  - SPITALE_email_NN, pentru atributul email.
    - *datele de contact sunt obligatorii.*
- **Constrângeri de tip UNIQUE**
  - SPITALE_telefon_U, pentru atributul telefon;
    - *nu pot exista mai multe spitale cu același număr de telefon.*
  - SPITALE_email_U, pentru atributul email;
    - *nu pot exista mai multe spitale cu aceeași adresă de email.*
- **Constrângeri de tip CHECK**
  - SPITALE_telefon_C, pentru atributul telefon;
    - *prin funcția LENGTH(), verificăm dacă numărul de telefon are 10 cifre.*
  - SPITALE_email_C, pentru atributul email.
    - *prin funcția REGEXP_LIKE(), verificăm dacă emailul este valid.*

4.2.4. Tabelul SECTII

- **Constrângeri de tip PRIMARY KEY**
  - SECTII_codSectie_PK, pentru atributul codSectie;
- **Constrângeri de tip NOT NULL**
  - SECTII_numeSectie_NN, pentru atributul numeSectie;
- **Constrângeri de tip UNIQUE**
  - SECTII_numeSectie_U, pentru atributul numeSectie.
    - *nu pot exista două tipuri de secții cu exact același nume.*

4.2.5. Tabelul SECTII_SPITALE

- **Constrângeri de tip PRIMARY KEY**
  - SECTII_SPITALE_idSectie_PK, pentru atributul idSectie;
- **Constrângeri de tip FOREIGN KEY**
  - SECTII_SPITALE_idSpital_FK, pentru atributul idSpital;
    - *face legătura cu tabelul SPITALE (idSpital);*
    - *determină în ce spital funcționează secția respectivă.*
  - SECTII_SPITALE_codSectie_FK, pentru atributul codSectie;
    - *face legătura cu tabelul SECTII (codSectie);*
    - *determină tipul secției.*
- **Constrângeri de tip NOT NULL**
  - SECTII_SPITALE_idSpital_NN, pentru atributul idSpital;
    - *secția trebuie să aparțină de un spital.*
  - SECTII_SPITALE_codSectie_NN, pentru atributul codSectie;
    - *secția trebuie să fie de un anume tip.*
  - SECTII_SPITALE_etaj_NN, pentru atributul etaj;
    - *secția se află pe la un anumit etaj (parter = 0).*
- **Constrângeri de tip UNIQUE**
  - SECTII_SPITALE_telefon_U, pentru atributul telefon;
    - *nu pot exista mai multe secții cu același număr de telefon.*
  - SECTII_SPITALE_email_U, pentru atributul email;
    - *nu pot exista mai multe secții cu aceeași adresă de email.*

- **Constrângeri de tip CHECK**
    - SECTII_SPITALE_telefon_C, pentru atributul telefon;
        - *prin funcția LENGTH(), verificăm dacă numărul de telefon are 10 cifre.*
    - SECTII_SPITALE_email_C, pentru atributul email.
        - *prin funcția REGEXP_LIKE(), verificăm dacă emailul este valid.*

4.2.6. Tabelul FUNCTII

- **Constrângeri de tip PRIMARY KEY**
    - FUNCTII_codFunctie_PK, pentru atributul codFunctie;
- **Constrângeri de tip NOT NULL**
    - FUNCTII_numeFunctie_NN, pentru atributul numeFunctie;
        - *funcția trebuie să aibă o denumire, obligatoriu.*
- **Constrângeri de tip UNIQUE**
    - FUNCTII_numeFunctii_U, pentru atributul numeFunctie.
        - *nu pot exista mai multe tipuri de funcții, cu exact același număr.*

4.2.7. Tabelul DOCTORI

- **Constrângeri de tip PRIMARY KEY**
    - DOCTORI_codParafa_PK, pentru atributul codParafa;
- **Constrângeri de tip FOREIGN KEY**
    - DOCTORI_codFunctie_FK, pentru atributul codFunctie;
        - *face legătura cu tabelul FUNCTII (codFunctie);*
        - *determină ce tip de funcție ocupă doctorul.*
    - DOCTORI_idSectie_FK, pentru atributul idSectie;
        - *face legătura cu tabelul SECTII_SPITALE (idSectie);*
        - *determină secția în care lucrează doctorul.*
- **Constrângeri de tip NOT NULL**
    - DOCTORI_nume_NN, pentru atributul nume;
        - *doctorul trebuie să aibă numele obligatoriu public.*
    - DOCTORI_prenume_NN, pentru atributul prenume;
        - *doctorul trebuie să aibă numele obligatoriu public.*
    - DOCTORI_CNP_NN, pentru atributul CNP;

        - *doctorul trebuie să aibă CNP-ul în baza de date pentru autentificare.*
    - DOCTORI_telefon_NN, pentru atributul telefon;
        - *datele de contact sunt obligatorii.*
    - DOCTORI_email_NN, pentru atributul email;
        - *datele de contact sunt obligatorii.*
- **Constrângeri de tip UNIQUE**
    - DOCTORI_CNP_U, pentru atributul CNP;
        - *CNP-ul este unic, pentru fiecare medic.*
    - DOCTORI_telefon_U, pentru atributul telefon;
        - *nu pot exista mai mulți doctori cu același număr de telefon.*
    - DOCTORI_email_U, pentru atributul email;
        - *nu pot exista mai mulți doctori cu aceeași adresă de email.*
- **Constrângeri de tip CHECK**
    - DOCTORI_CNP_C, pentru atributul CNP;
        - *prin funcția LENGTH(), verificăm dacă CNP-ul are 13 cifre.*
    - DOCTORI_telefon_C, pentru atributul telefon;
        - *prin funcția LENGTH(), verificăm dacă numărul de telefon are 10 cifre.*
    - DOCTORI_email_C, pentru atributul email.
        - *prin funcția REGEXP_LIKE(), verificăm dacă emailul este valid.*

4.2.8. Tabelul PACIENTI

- **Constrângeri de tip PRIMARY KEY**
    - PACIENTI_idPacient_PK, pentru atributul idPacient;
- **Constrângeri de tip NOT NULL**
    - PACIENTI_nume_NN, pentru atributul nume;
        - *numele pacientului este obligatoriu.*
    - PACIENTI_prenume_NN, pentru atributul prenume;
        - *numele pacientului este obligatoriu.*
    - PACIENTI_CNP_NN, pentru atributul CNP;
        - *CNP-ul pacientului este necesar în identificare, dar și util în baza de date pentru ca pacientul să acceseze detaliile despre internările acestuia în spitale.*

- PACIENTI_strada_NN, pentru atributul strada;
  - *adresa pacientului este obligatorie.*
- PACIENTI_numar_NN, pentru atributul numar;
  - *adresa pacientului este obligatorie.*
- PACIENTI_localitate_NN, pentru atributul localitate;
  - *adresa pacientului este obligatorie.*
- PACIENTI_telefon_NN, pentru atributul telefon;
  - *numărul de telefon este obligatoriu.*
- PACIENTI_asigurat_NN, pentru atributul asigurat;
  - *spitalul în care este internat pacientul trebuie să știe dacă acesta este sau nu este asigurat.*
- PACIENTI_dataInternare_NN, pentru atributul dataInternare;
  - *data internării este obligatorie.*

- **Constrângeri de tip CHECK**
  - PACIENTI_CNP_C, pentru atributul CNP;
    - *prin funcția LENGTH(), verificăm dacă CNP-ul are 13 cifre.*
  - PACIENTI_telefon_C, pentru atributul telefon;
    - *prin funcția LENGTH(), verificăm dacă numărul de telefon are 10 cifre.*
  - PACIENTI_email_C, pentru atributul email.
    - *prin funcția REGEXP_LIKE(), verificăm dacă emailul este valid.*
  - PACIENT_asigurat_C, pentru atributul asigurat;
    - *se verifică dacă valoarea atributului "asigurat" este egală cu 0 sau 1.*
- **TRIGGERS (explicate în partea a 5-a detaliat)**
  - CHECK_dataInternare, pentru atributul dataInternare;
    - *acest trigger verifică dacă data internării este mai mică sau egală cu data sistemului sau mai mare decât 01-JAN-1900.*
  - CHECK_Internare, pentru tot tabelul;
    - *triggerul se asigură că nu se introduc mai multe internări simultane ale aceleiași persoane.*
  - CHECK_dataExternare, pentru atributul dataExternare.
    - *acest trigger verifică dacă data externării este mai mare sau egală cu data internării sau dacă este mai mică sau egală cu data sistemului.*

4.2.9. Tabelul CONSULTATII

- **Constrângeri de tip PRIMARY KEY**
  - CONSULTATII_idConsultatie_PK, pentru atributul idConsultatie;
- **Constrângeri de tip FOREIGN KEY**
  - CONSULTATII_codParafa_FK, pentru atributul codParafa;
    - *face legătura cu tabelul DOCTORI (codParafa);*
    - *determină medicul care a efectuat consultația.*
  - CONSULTATII_idPacient_FK, pentru atributul idPacient;
    - *face legătura cu tabelul PACIENTI (idPacient);*
    - *determină pacientul care a fost examinat în consultație.*
- **Constrângeri de tip NOT NULL**
  - CONSULTATII_idPacient_NN, pentru atributul idPacient;
    - *consultația trebuie să implice obligatoriu un pacient.*
  - CONSULTATII_dataConsultatie_NN, pentru atributul dataConsultatie;
    - *data consultației trebuie să fie obligatorie.*
- **TRIGGERS (explicate în partea a 5-a detaliat)**
  - CHECK_dataConsultatie, pentru atributul dataConsultatie.
    - *acest trigger verifică, mai întâi verifică dacă pacientul figurează ca fiind internat, după care verifică dacă data consultației este mai mare sau egală cu data internării și dacă este mai mică sau egală cu data sistemului.*

4.2.10. Tabelul BOLI

- **Constrângeri de tip PRIMARY KEY**
  - BOLI_codBoala_PK, pentru atributul codBoala;
- **Constrângeri de tip NOT NULL**
  - BOLI_denumireBoala_NN, pentru atributul denumireBoala;
    - *boala trebuie să aibă o denumire obligatoriu.*
- **Constrângeri de tip UNIQUE**
  - BOLI_denumireBoala_U, pentru atributul denumireBoala.
    - *nu pot exista două boli cu exact același nume.*

4.2.11. Tabelul DIAGNOSTICE

- **Constrângeri de tip PRIMARY KEY**
  - DIAGNOSTICE_idDiagnostic_PK, pentru atributul idDiagnostic;
- **Constrângeri de tip FOREIGN KEY**
  - DIAGNOSTICE_idConsultatie_FK, pentru atributul idConsultatie;
    - *face legătura cu tabelul CONSULTATII (idConsultatie);*
    - *determină consultația din care reiese diagnosticul.*
  - DIAGNOSTICE_codBoala_FK, pentru atributul codBoala;
    - *face legătura cu tabelul BOLI (codBoala);*
    - *determină boala de care suferă pacientul, dacă este cazul.*
- **Constrângeri de tip NOT NULL**
  - DIAGNOSTICE_idConsultatie_NN, pentru atributul idConsultatie.
    - *diagnosticul vine obligatoriu în urma unei consultații, deci este un câmp obligatoriu.*

4.2.12. Tabelul SIMPTOME

- **Constrângeri de tip PRIMARY KEY**
  - SIMPTOME_codSimptom_PK, pentru atributul codSimptom;
- **Constrângeri de tip NOT NULL**
  - SIMPTOME_denumireSimptom_NN, pentru atributul denumireSimptom;
    - *simptomul are obligatoriu o denumire.*
- **Constrângeri de tip UNIQUE**
  - SIMPTOME_denumireSimptom_U, pentru atributul denumireSimptom.
    - *nu pot exista simptome cu exact aceeași denumire.*

4.2.13. Tabelul DIAGNOSTICE_SIMPTOME

- **Constrângeri de tip PRIMARY KEY**
  - DIAG_SIMP_PK, pentru atributele (idDiagnostic, codSimptom);
- **Constrângeri de tip FOREIGN KEY**
  - DIAG_SIMP_idDiagnostic_FK, pentru atributul idDiagnostic;
    - *face legătura cu tabelul DIAGNOSTICE (idDiagnostic);*
    - *determină diagnosticul pentru care există o serie de simptome.*

- DIAG_SIMP_codSimptom_FK, pentru atributul codSimptom.
  - *face legătura cu tabelul SIMPTOME (codSimptom);*
  - *determină simptomul care corespunde unui diagnostic.*

4.2.14. Tabelul TRATAMENTE

- **Constrângeri de tip PRIMARY KEY**
  - TRATAMENTE_idTratament_PK, pentru atributul idTratament;
- **Constrângeri de tip FOREIGN KEY**
  - TRATAMENTE_codBoala_FK, pentru atributul codBoala;
    - *face legătura cu tabelul BOLI;*
    - *determină boala pentru care se aplică tratamentul.*
- **Constrângeri de tip NOT NULL**
  - TRATAMENTE_codBoala_NN, pentru atributul codBoala;
    - *tratamentul se aplică obligatoriu pentru o boală, deci câmpul este obligatoriu.*
  - TRATAMENTE_perioadaAdministrare_NN, pentru atributul perioadaAdministrare;
    - *tratamentul are o perioadă de administrare.*

4.2.15. Tabelul MEDICAMENTE

- **Constrângeri de tip PRIMARY KEY**
  - MEDICAMENTE_idMedicament_PK, pentru atributul idMedicament;
- **Constrângeri de tip NOT NULL**
  - MEDICAMENTE_numeMedicament_NN, pentru atributul numeMedicament;
    - *medicamentul trebuie să aibă obligatoriu o denumire.*
  - MEDICAMENTE_tipMedicament_NN, pentru atributul tipMedicament;
    - *tipul de medicament trebuie specificat, fiind un detaliu esențial.*
  - MEDICAMENT_dozaUnitate_NN, pentru atributul dozaUnitate.
    - *doză per unitate, trebuie specificată fiind un detaliu esențial.*

4.2.16. Tabelul TRATAMENTE_ADMINISTRATE

- **Constrângeri de tip PRIMARY KEY**
  - TRAT_ADM_PK, pentru atributele (idTratament, idMedicament);
- **Constrângeri de tip FOREIGN KEY**
  - TRAT_ADM_idTratament_FK, pentru atributul idTratament;
    - *face legătura cu tabelul TRATAMENTE (idTratament);*
    - *determină tratamentul care conține o serie de medicamente.*
  - TRAT_ADM_idMedicament_FK, pentru atributul idMedicament;
    - *face legătura cu tabelul MEDICAMENTE (idMedicament);*
    - *determină medicamentul care face parte dintr-un tratament.*
- **Constrângeri de tip NOT NULL**
  - TRAT_ADM_doza_NN, pentru atributul doza.
    - *doza care se administrează este un detaliu esențial în fiecare tratament, deci este un atribut obligatoriu.*

4.3. Schemele relaționale

4.3.1. Schemele relaționale

Schemele relaționale atașate diagramei conceptuale sunt:

- JUDETE (codJudet#, numeJudet);
- LOCALITATI (idLocalitate#, numeLocalitate, codPostal, codJudet (FK));
- SPITALE (idSpital#, numeSpital, idLocalitate (FK), strada, numar, telefon, email);
- SECTII (codSectie#, numeSectie);
- SECTII_SPITALE (idSectie#, idSpital (FK), codSectie (FK), corp, etaj, telefon, email);
- FUNCTII (codFunctie#, numeFunctie);
- DOCTORI (codParafa#, nume, prenume, CNP, telefon, email, codFunctie (FK), idSectie (FK));
- PACIENTI (idPacient, nume, prenume, strada, numar, localitate, telefon, email, asigurat, dataInternare, dataExternare);
- CONSULTATII (idConsultatie#, codParafa (FK), idPacient (FK), dataConsultatie);
- DIAGNOSTICE (idDiagnostic, idConsultatie (FK), codBoala (FK));
- DIAGNOSTICE_SIMPTOME (idDiagnostic# (FK), codSimptom# (FK));
- SIMPTOME (codSimptom#, denumireSimptom);

- BOLI (codBoala#, denumireBoala);
- TRATAMENTE (idTratament#, codBoala (FK), perioadaAdministrare, indicatii);
- TRATAMENTE_ADMINISTRATE (idTratament# (FK), idMedicament# (FK), doza);
- MEDICAMENTE (idMedicament#, numeMedicament, tipMedicament, dozaUnitate).

4.3.2. Descrierea constrângerilor ON DELETE

Există 15 constrângeri ON DELETE, câte una pentru fiecare constrângere de tip FOREIGN KEY. Acestea sunt:

- LOCALITATI_codJudet_FK, **ON DELETE CASCADE**
  - când un județ este șters din baza de date, toate localitățile care aveau codul județului respectiv vor fi șterse;
  - ON DELETE SET NULL nu era o opțiune bună, deoarece nu există localități care să nu aparțină de un județ.
- SPITALE_idLocalitate_FK, **ON DELETE CASCADE**
  - când o localitate este ștearsă din baza de date, toate spitalele care se aflau în acea localitate vor fi șterse.
- SECTII_SPITALE_idSpital_FK, **ON DELETE CASCADE**
  - când un spital este șters din baza de date, toate secțiile care funcționau în acel spital vor fi șterse.
- SECTII_SPITALE_codSectie_FK, **ON DELETE SET NULL**
  - dacă un tip de secție este șters din baza de date, secțiile care erau de acel tip, vor rămâne în spital, dar fără un tip specific;
  - practic, dacă se șterge, spre exemplu, tipul de secție „Pediatrie”, atunci toate secțiile „Pediatrie” din spitalele existente în baza de date vor deveni nefuncționale (se vor închide) până când li se vor atribui câte o nouă funcționalitate.
- DOCTORI_codFunctie_FK, **ON DELETE SET NULL**
  - dacă un tip de funcție este șters din baza de date, doctorul nu va mai avea o funcție specifică, însă acesta rămâne doctor în spitalul în care lucrează.
- DOCTORI_idSectie_FK, **ON DELETE SET NULL**
  - dacă secția în care lucra medicul este ștearsă din baza de date, doctorul nu va fi șters, ci va putea să primească o nouă secție în care să lucreze.

- CONSULTATII_codParafa_FK, **ON DELETE SET NULL**
    - dacă un doctor care a efectuat o consultație pentru un pacient este șters din baza de date, consultația poate rămâne in baza de date, pentru a putea fi folosită de alți medici, deoarece pe baza consultației se stabilește diagnosticul;
    - în acest fel, chiar dacă doctorul nu mai există, istoricul pacientului rămâne.
- CONSULTATII_idPacient_FK, **ON DELETE CASCADE**
    - în schimb, dacă pacientul este șters din baza de date, consultația în care acesta apărea, este nulă;
    - nemaiavând valoare, importanță, consultațiile pacientului respectiv pot fi șterse.
- DIAGNOSTICE_idConsultatie_FK, **ON DELETE CASCADE**
    - în cazul în care consultația asociată unui diagnostic a fost ștearsă din baza de date, acest diagnostic este nul, deci poate fi șters.
- DIAGNOSTICE_codBoala_FK, **ON DELETE SET NULL**
    - dacă un tip de boală este șters din baza de date, atunci diagnosticele care aveau această boală drept rezultat al consultației, nu vor fi șterse, deoarece, chiar dacă nu mai au codul bolii, simptomele care au fost prezente la pacient, rămân;
    - în acest fel, chiar dacă nu mai este specificată boala, pacientului i se poate face un istoric medical, doar cu simptomele acelui diagnostic.
- DIAG_SIMP_idDiagnostic_FK, **ON DELETE CASCADE**
    - în cazul în care un diagnostic este șters, simptomele asociate acelui diagnostic trebuie să fie șterse.
- DIAG_SIMP_codSimptom_FK, **ON DELETE CASCADE**
    - dacă un simptom este șters din baza de date, atunci toate intrările unde diagnosticele care prezentau și acest simptom vor fi șterse.
- TRATAMENTE_codBoala_FK, **ON DELETE CASCADE**
    - dacă un tip de boală este șters, atunci, în mod implicit, tratamentul pentru vindecarea acelei boli, este nul, deci trebuie să fie șters.
- TRAT_ADM_idTratament_FK, **ON DELETE CASCADE**
    - dacă un tratament este șters, medicamentele folosite pot fi șterse din tabel.
- TRAT_ADM_idMedicament_FK, **ON DELETE CASCADE**
    - dacă un medicament este șters, tratamentul va continua doar cu restul medicamentelor.

5. Scriptul SQL

5.1. Introducere

Scriptul SQL, salvat cu numele **scriptSQL_MAXIM_TIBERIU_gr251.sql**, a fost scris folosind Oracle SQL Developer v20, iar pentru baza de date locală, Oracle Database 18c.

Scriptul SQL trebuie să ruleze fără probleme. Singura excepție este prima rulare a scriptului, deoarece, în prima etapă are loc procesul de DROP pentru secvențe, tabele, constrângeri și triggere. Din cauza faptului că cele menționate anterior încă nu există, se vor afișa erori, dar crearea tabelelor și popularea acestora vor continua fără probleme.

5.2. Etapa de DROP

Inițial se face DROP la secvențe, după care DROP la triggere. Urmează tabelele, iar pentru cele care au constrângeri de tip FOREIGN KEY, se face mai întâi DROP acelor constrângeri.

```
DROP SEQUENCE JUDETE_codJudet_SEQ;

DROP SEQUENCE LOCALITATI_idLocalitate_SEQ;

DROP SEQUENCE LOCALITATI_codPostal_SEQ;

DROP SEQUENCE SPITALE_idSpital_SEQ;

DROP SEQUENCE SECTII_codSectie_SEQ;

DROP SEQUENCE SECTII_SPITALE_idSectie_SEQ;

DROP SEQUENCE FUNCTII_codFunctie_SEQ;

DROP SEQUENCE DOCTORI_codParafa_SEQ;

DROP SEQUENCE PACIENTI_idPacient_SEQ;

DROP SEQUENCE CONSULTATII_idConsultatie_SEQ;

DROP SEQUENCE BOLI_codBoala_SEQ;

DROP SEQUENCE DIAGNOSTICE_idDiagnostic_SEQ;

DROP SEQUENCE SIMPTOME_codSimptom_SEQ;

DROP SEQUENCE TRATAMENTE_idTratament_SEQ;

DROP SEQUENCE MEDICAMENTE_idMedicament_SEQ;
```

```
DROP TRIGGER CHECK_dataInternare;

DROP TRIGGER CHECK_dataExternare;

DROP TRIGGER CHECK_dataConsultatie;

ALTER TABLE TRATAMENTE_ADMINISTRATE

   DROP CONSTRAINT TRAT_ADM_idTratament_FK

   DROP CONSTRAINT TRAT_ADM_idMedicament_FK;

DROP TABLE TRATAMENTE_ADMINISTRATE ;

DROP TABLE MEDICAMENTE;

ALTER TABLE TRATAMENTE

   DROP CONSTRAINT TRATAMENTE_codBoala_FK;

DROP TABLE TRATAMENTE;

ALTER TABLE DIAGNOSTICE_SIMPTOME

   DROP CONSTRAINT DIAG_SIMP_codSimptom_FK

   DROP CONSTRAINT DIAG_SIMP_idDiagnostic_FK;

DROP TABLE DIAGNOSTICE_SIMPTOME;

DROP TABLE SIMPTOME;

ALTER TABLE DIAGNOSTICE

   DROP CONSTRAINT DIAGNOSTICE_codBoala_FK

   DROP CONSTRAINT DIAGNOSTICE_idConsultatie_FK;

DROP TABLE DIAGNOSTICE;

DROP TABLE BOLI;

ALTER TABLE CONSULTATII

   DROP CONSTRAINT CONSULTATII_idPacient_FK

   DROP CONSTRAINT CONSULTATII_codParafa_FK;

DROP TABLE CONSULTATII;
```

```
DROP TABLE PACIENTI;

ALTER TABLE DOCTORI
   DROP CONSTRAINT DOCTORI_idSectie_FK
   DROP CONSTRAINT DOCTORI_codFunctie_FK;
DROP TABLE DOCTORI;

DROP TABLE FUNCTII;

ALTER TABLE SECTII_SPITALE
DROP CONSTRAINT SECTII_SPITALE_idSpital_FK
DROP CONSTRAINT SECTII_SPITALE_codSectie_FK;
DROP TABLE SECTII_SPITALE;

DROP TABLE SECTII;

ALTER TABLE SPITALE
   DROP CONSTRAINT SPITALE_idLocalitate_FK;
DROP TABLE SPITALE;

ALTER TABLE LOCALITATI
   DROP CONSTRAINT LOCALITATI_codJudet_FK;
DROP TABLE LOCALITATI;

DROP TABLE JUDETE;
```

5.3. Crearea tabelelor, inclusiv a constrângerilor

5.3.1. Crearea tabelului JUDETE

Am creat secvența JUDETE_codJudet_SEQ pentru a asigna valori pentru cheia primară codJudet. Sunt doar 42 de valori posibile (pentru județele României, inclus fiind și București), deoarece proiectul gestionează pacienții la nivel național, deci nu se va putea adăuga un alt județ.

```
CREATE SEQUENCE JUDETE_codJudet_SEQ MAXVALUE 42 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;

CREATE TABLE JUDETE (
   codJudet NUMBER(2, 0)
      CONSTRAINT JUDETE_codJudet_PK PRIMARY KEY,
   numeJudet VARCHAR2(30 CHAR)
      CONSTRAINT JUDETE_numeJudet_NN NOT NULL
      CONSTRAINT JUDETE_numeJudet_U UNIQUE
);
```

5.3.2. Crearea tabelului LOCALITATI

Am creat secvența LOCALITATI_idLocalitate_SEQ pentru a asigna valori cheii primare idLocalitate. De asemenea, secvența LOCALITATI_codPostal_SEQ asignează automat câte un cod poștal pentru fiecare localitate introdusă în baza de date.

```
CREATE SEQUENCE LOCALITATI_idLocalitate_SEQ MAXVALUE 99999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;

CREATE SEQUENCE LOCALITATI_codPostal_SEQ MAXVALUE 999999 INCREMENT BY 1 START WITH 100000 NOCACHE ORDER NOCYCLE;

CREATE TABLE LOCALITATI (
   idLocalitate NUMBER(5, 0)
      CONSTRAINT LOCALITATI_idLocalitate_PK PRIMARY KEY,
   numeLocalitate VARCHAR2(30 CHAR)
      CONSTRAINT LOCALITATI_numeLocalitate_NN NOT NULL,
   codPostal NUMBER(6, 0)
      CONSTRAINT LOCALITATI_codPostal_NN NOT NULL
      CONSTRAINT LOCALITATI_codPostal_U UNIQUE,
   codJudet NUMBER(2, 0)
      CONSTRAINT LOCALITATI_codJudet_FK REFERENCES JUDETE(codJudet) ON DELETE CASCADE
      CONSTRAINT LOCALITATI_codJudet_NN NOT NULL
);
```

5.3.3. Crearea tabelului SPITALE

Am creat secvența SPITALE_idSpital_SEQ pentru a asigna valori cheii primare idSpital.

```
CREATE SEQUENCE SPITALE_idSpital_SEQ MAXVALUE 99999 INCREMENT BY 1 START WITH 10001 NOCACHE ORDER NOCYCLE;
```

```
CREATE TABLE SPITALE (
   idSpital NUMBER(5, 0)
      CONSTRAINT SPITALE_idSpital_PK PRIMARY KEY,
   numeSpital VARCHAR2(50 CHAR)
      CONSTRAINT SPITALE_numeSpital_NN NOT NULL,
   idLocalitate NUMBER(5, 0)
      CONSTRAINT SPITALE_idLocalitate_FK REFERENCES LOCALITATI(idLocalitate) ON DELETE CASCADE
      CONSTRAINT SPITALE_idLocalitate_NN NOT NULL,
   strada VARCHAR2(30 CHAR)
      CONSTRAINT SPITALE_strada_NN NOT NULL,
   numar NUMBER(3, 0)
      CONSTRAINT SPITALE_numar_NN NOT NULL,
   telefon VARCHAR2(10 CHAR)
      CONSTRAINT SPITALE_telefon_NN NOT NULL
      CONSTRAINT SPITALE_telefon_U UNIQUE
      CONSTRAINT SPITALE_telefon_C CHECK (LENGTH(telefon)=10),
   email VARCHAR2(50 CHAR)
      CONSTRAINT SPITALE_email_NN NOT NULL
      CONSTRAINT SPITALE_email_U UNIQUE
      CONSTRAINT SPITALE_email_C CHECK (REGEXP_LIKE(email, '[[:alnum:]]+@[[:alnum:]]+\.[[:alnum:]]'))
);
```

5.3.4. Crearea tabelului SECTII

Am creat secvența SECTII_codSectie_SEQ pentru a asigna valori cheii primare codSectie.

```
CREATE SEQUENCE SECTII_codSectie_SEQ MAXVALUE 100 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;

CREATE TABLE SECTII (
   codSectie   NUMBER(3, 0)
      CONSTRAINT SECTII_codSectie_PK PRIMARY KEY,
   numeSectie  VARCHAR2(40 CHAR)
      CONSTRAINT SECTII_numeSectie_NN NOT NULL
      CONSTRAINT SECTII_numeSectie_U UNIQUE
);
```

5.3.5. Crearea tabelului SECTII_SPITALE

Am creat secvența SECTII_SPITALE_idSectie_SEQ pentru a asigna valori cheii primare idSectie.

```
CREATE SEQUENCE SECTII_SPITALE_idSectie_SEQ MAXVALUE 999999 INCREMENT BY 1 START WITH 100001 NOCACHE ORDER NOCYCLE;

CREATE TABLE SECTII_SPITALE (
   idSectie NUMBER(6, 0)
      CONSTRAINT SECTII_SPITALE_idSectie_PK PRIMARY KEY,
   idSpital NUMBER(5, 0)
      CONSTRAINT SECTII_SPITALE_idSpital_FK REFERENCES SPITALE(idSpital) ON DELETE CASCADE
      CONSTRAINT SECTII_SPITALE_idSpital_NN NOT NULL,
   codSectie NUMBER(3, 0)
```

```
    CONSTRAINT SECTII_SPITALE_codSectie_FK REFERENCES SECTII(codSectie) ON DELETE SET NULL,
  corp VARCHAR2(15 CHAR),
  etaj NUMBER(1, 0)
    CONSTRAINT SECTII_SPITALE_etaj_NN NOT NULL,
  telefon VARCHAR2(10 CHAR)
    CONSTRAINT SECTII_SPITALE_telefon_U UNIQUE
    CONSTRAINT SECTII_SPITALE_telefon_C CHECK (LENGTH(telefon)=10),
  email VARCHAR2(50 CHAR)
    CONSTRAINT SECTII_SPITALE_email_U UNIQUE
    CONSTRAINT SECTII_SPITALE_email_C CHECK (REGEXP_LIKE(email, '[[:alnum:]]+@[[:alnum:]]+\.[[:alnum:]]'))
);
```

5.3.6. Crearea tabelului FUNCTII

Am creat secvența FUNCTII_codFunctie_SEQ pentru a asigna valori cheii primare codFunctie.

```
CREATE SEQUENCE FUNCTII_codFunctie_SEQ MAXVALUE 100 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;
CREATE TABLE FUNCTII (
  codFunctie NUMBER(3, 0)
    CONSTRAINT FUNCTII_codFunctie_PK PRIMARY KEY,
  numeFunctie VARCHAR2(30 CHAR)
    CONSTRAINT FUNCTII_numeFunctie_NN NOT NULL
    CONSTRAINT FUNCTII_numeFunctii_U UNIQUE
);
```

5.3.7. Crearea tabelului DOCTORI

Am creat secvența DOCTORI_codParafa_SEQ pentru a asigna automat valori cheii primare codParafa.

```
CREATE SEQUENCE DOCTORI_codParafa_SEQ MAXVALUE 999999 INCREMENT BY 1
START WITH 100000 NOCACHE ORDER NOCYCLE;

CREATE TABLE DOCTORI (
   codParafa NUMBER(6, 0)
      CONSTRAINT DOCTORI_codParafa_PK PRIMARY KEY,
   nume VARCHAR2(30 CHAR)
      CONSTRAINT DOCTORI_nume_NN NOT NULL,
   prenume VARCHAR2(30 CHAR)
      CONSTRAINT DOCTORI_prenume_NN NOT NULL,
   CNP VARCHAR2(13 CHAR)
      CONSTRAINT DOCTORI_CNP_NN NOT NULL
      CONSTRAINT DOCTORI_CNP_U UNIQUE
      CONSTRAINT DOCTORI_CNP_C CHECK (LENGTH(CNP)=13),
   telefon VARCHAR2(10 CHAR)
      CONSTRAINT DOCTORI_telefon_NN NOT NULL
      CONSTRAINT DOCTORI_telefon_U UNIQUE
      CONSTRAINT DOCTORI_telefon_C CHECK (LENGTH(telefon)=10),
   email VARCHAR2(30 CHAR)
      CONSTRAINT DOCTORI_email_NN NOT NULL
      CONSTRAINT DOCTORI_email_U UNIQUE
      CONSTRAINT DOCTORI_email_C CHECK (REGEXP_LIKE(email,
'[[:alnum:]]+@[[:alnum:]]+\.[[:alnum:]]')),
```

```
    codFunctie NUMBER(3, 0)
        CONSTRAINT DOCTORI_codFunctie_FK REFERENCES FUNCTII(codFunctie) ON DELETE SET NULL,
    idSectie NUMBER(6, 0)
        CONSTRAINT DOCTORI_idSectie_FK REFERENCES SECTII_SPITALE(idSectie) ON DELETE SET NULL
);
```

5.3.8. Crearea tabelului PACIENTI

Am creat secvența PACIENTI_idPacient_SEQ pentru a asigna valori cheii primare idPacient.

```
CREATE SEQUENCE PACIENTI_idPacient_SEQ MAXVALUE 999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;
CREATE TABLE PACIENTI (
    idPacient NUMBER(6, 0)
        CONSTRAINT PACIENTI_idPacient_PK PRIMARY KEY,
    nume VARCHAR2(30 CHAR)
        CONSTRAINT PACIENTI_nume_NN NOT NULL,
    prenume VARCHAR2(30 CHAR)
        CONSTRAINT PACIENTI_prenume_NN NOT NULL,
    CNP VARCHAR2(13 CHAR)
        CONSTRAINT PACIENTI_CNP_NN NOT NULL
        CONSTRAINT PACIENTI_CNP_C CHECK (LENGTH(CNP)=13),
    strada VARCHAR2(30 CHAR)
        CONSTRAINT PACIENTI_strada_NN NOT NULL,
    numar NUMBER(3, 0)
        CONSTRAINT PACIENTI_numar_NN NOT NULL,
```

```
    localitate VARCHAR2(30 CHAR)
        CONSTRAINT PACIENTI_localitate_NN NOT NULL,
    telefon VARCHAR2(10 CHAR)
        CONSTRAINT PACIENTI_telefon_NN NOT NULL
        CONSTRAINT PACIENTI_telefon_C CHECK (LENGTH(telefon)=10),
    email VARCHAR2(30 CHAR)
        CONSTRAINT PACIENTI_email_C CHECK (REGEXP_LIKE(email,
'[[:alnum:]]+@[[:alnum:]]+\.[[:alnum:]]')),
    asigurat NUMBER(1, 0)
        CONSTRAINT PACIENTI_asigurat_NN NOT NULL
        CONSTRAINT PACIENTI_asigurat_C CHECK (asigurat=1 OR asigurat=0),
    dataInternare DATE
        CONSTRAINT PACIENTI_dataInternare_NN NOT NULL,
    dataExternare DATE
);
```

Pentru tabelul PACIENTI, există trei triggere care verifică condițiile pentru data internării și pentru data externării.

Triggerul CHECK_dataInternare verifică la inserare sau la update dacă data internării este validă. O dată validă presupune ca aceasta să se afle în intervalul [01/01/1900, sysdate]. În cazul în care se introduce o dată incorectă, va returna o eroare cu un mesaj sugestiv.

Triggerul CHECK_Internare va returna o eroare cu un mesaj sugestiv în cazul în care se încearcă inserarea unui nou pacient, implicit o nouă interna, dacă deja este activă o internare a aceluiași pacient (CNP-ul este același).

Triggerul CHECK_dataExternare verifică la inserare sau la update dacă data externării este validă. O dată validă presupune ca aceasta să se afle în intervalul [dataInternare, sysdate]. În cazul în care se introduce o dată incorectă, va returna o eroare cu un mesaj sugestiv.

```
CREATE OR REPLACE TRIGGER CHECK_dataInternare
   BEFORE INSERT OR UPDATE ON PACIENTI
   FOR EACH ROW
BEGIN
      IF(:NEW.dataInternare < date '1900-01-01' or :NEW.dataInternare > sysdate)
      THEN
        RAISE_APPLICATION_ERROR(-20001, 'Data internarii nu poate fi mai mica decat 01-JAN-1900 sau mai mare decat data curenta!');
      END IF;
END;
/


CREATE OR REPLACE TRIGGER CHECK_Internare
   BEFORE INSERT ON PACIENTI
   FOR EACH ROW
DECLARE
   pacienti_CNT NUMBER;
BEGIN
      SELECT COUNT(*) INTO pacienti_CNT FROM PACIENTI WHERE (CNP=:NEW.CNP AND dataExternare IS NULL);
      IF(pacienti_CNT > 0)
      THEN
        RAISE_APPLICATION_ERROR(-20001, 'Pacientul este deja internat in spital!');
      END IF;
END;
/
```

```
CREATE OR REPLACE TRIGGER CHECK_dataExternare
   BEFORE INSERT OR UPDATE ON PACIENTI
   FOR EACH ROW
BEGIN
   IF(:NEW.dataExternare > sysdate)
   THEN
      RAISE_APPLICATION_ERROR(-20001, 'Data externarii nu poate fi mai mare decat data curenta!');
   ELSIF(:NEW.dataExternare < :OLD.dataInternare)
   THEN
      RAISE_APPLICATION_ERROR(-20001, 'Data externarii nu poate fi mai mica decat data internarii!');
   END IF;
END;
/
```

5.3.9. Crearea tabelului CONSULTATII

Am creat secvența CONSULTATII_idConsultatie_SEQ pentru a asigna valori cheii primare idConsultatie.

```
CREATE SEQUENCE CONSULTATII_idConsultatie_SEQ MAXVALUE 999999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;

CREATE TABLE CONSULTATII (
   idConsultatie NUMBER(6, 0)
      CONSTRAINT CONSULTATII_idConsultatie_PK PRIMARY KEY,
   codParafa NUMBER(6, 0)
      CONSTRAINT CONSULTATII_codParafa_FK REFERENCES DOCTORI(codParafa) ON DELETE SET NULL,
   idPacient NUMBER(6, 0)
      CONSTRAINT CONSULTATII_idPacient_FK REFERENCES PACIENTI(idPacient) ON DELETE CASCADE
      CONSTRAINT CONSULTATII_idPacient_NN NOT NULL,
   dataConsultatie DATE
      CONSTRAINT CONSULTATII_dataConsultatie_NN NOT NULL
);
```

De asemenea, tabelul CONSULTATII conține un trigger, CHECK_dataConsultatie, care face două verificări.

- verifică dacă pacientul căruia dorim să îi inserăm sau să îi modificăm consultația figurează ca fiind internat; acest lucru înseamnă că data externării este null.
- dacă pacientul este internat, atunci se verifică ca data consultației să se afle în intervalul [data internării, sysdate].
- în cazul în care una dintre cele două condiții nu se îndeplinește, triggerul va returna o eroare cu un mesaj sugestiv.

```
CREATE OR REPLACE TRIGGER CHECK_dataConsultatie
   BEFORE INSERT OR UPDATE ON CONSULTATII
   FOR EACH ROW
DECLARE
   pacienti_CNT NUMBER;
   dataInt DATE;
BEGIN
   SELECT COUNT(*) INTO pacienti_CNT FROM PACIENTI;
   IF pacienti_CNT > 0
   THEN
      SELECT dataInternare INTO dataInt FROM PACIENTI WHERE (idPacient=:NEW.idPacient
AND dataExternare IS NULL);
      IF( :NEW.dataConsultatie > sysdate)
      THEN
         RAISE_APPLICATION_ERROR(-20001, 'Data consultatiei nu poate fi mai mare decat data
curenta!');
      ELSIF(:NEW.dataConsultatie < dataInt)
      THEN
         RAISE_APPLICATION_ERROR(-20001, 'Data consultatiei nu poate fi mai mica decat data
internarii!');
      END IF;
   ELSE
      RAISE_APPLICATION_ERROR(-20001, 'Pacientul nu figureaza ca fiind internat in spital!');
   END IF;
END;
/
```

5.3.10. Crearea tabelului BOLI

Am creat secvența BOLI_codBoala_SEQ pentru a asigna valori cheii primare codBoala.

```
CREATE SEQUENCE BOLI_codBoala_SEQ MAXVALUE 999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;

CREATE TABLE BOLI (
   codBoala NUMBER(3, 0)
      CONSTRAINT BOLI_codBoala_PK PRIMARY KEY,
   denumireBoala VARCHAR2(50 CHAR)
      CONSTRAINT BOLI_denumireBoala_NN NOT NULL
      CONSTRAINT BOLI_denumireBoala_U UNIQUE
);
```

5.3.11. Crearea tabelului DIAGNOSTICE

Am creat secvența DIAGNOSTICE_idDiagnostic_SEQ pentru autoincrementare.

```
CREATE SEQUENCE DIAGNOSTICE_idDiagnostic_SEQ MAXVALUE 999999 INCREMENT BY 1 START WITH 100000 NOCACHE ORDER NOCYCLE;

CREATE TABLE DIAGNOSTICE (
   idDiagnostic NUMBER(6, 0)
      CONSTRAINT DIAGNOSTICE_idDiagnostic_PK PRIMARY KEY,
   idConsultatie NUMBER(6, 0)
      CONSTRAINT DIAGNOSTICE_idConsultatie_FK REFERENCES CONSULTATII(idConsultatie) ON DELETE CASCADE
      CONSTRAINT DIAGNOSTICE_idConsultatie_NN NOT NULL,
   codBoala NUMBER(3, 0)
      CONSTRAINT DIAGNOSTICE_codBoala_FK REFERENCES BOLI(codBoala) ON DELETE SET NULL
);
```

5.3.12. Crearea tabelului SIMPTOME

Am creat secvența SIMPTOME_codSimptom_SEQ pentru autoincrementare.

```
CREATE SEQUENCE SIMPTOME_codSimptom_SEQ MAXVALUE 999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;

CREATE TABLE SIMPTOME (
   codSimptom NUMBER(3, 0)
      CONSTRAINT SIMPTOME_codSimptom_PK PRIMARY KEY,
   denumireSimptom VARCHAR2(30 CHAR)
      CONSTRAINT SIMPTOME_denumireSimptom_NN NOT NULL
      CONSTRAINT SIMPTOME_denumireSimptom_U UNIQUE
);
```

5.3.13. Crearea tabelului DIAGNOSTICE_SIMPTOME

```
CREATE TABLE DIAGNOSTICE_SIMPTOME (
   idDiagnostic NUMBER(6, 0)
      CONSTRAINT DIAG_SIMP_idDiagnostic_FK REFERENCES DIAGNOSTICE(idDiagnostic) ON DELETE CASCADE,
   codSimptom NUMBER(3, 0)
      CONSTRAINT DIAG_SIMP_codSimptom_FK REFERENCES SIMPTOME(codSimptom) ON DELETE CASCADE,
   CONSTRAINT DIAG_SIMP_PK PRIMARY KEY(idDiagnostic, codSimptom)
);
```

5.3.14. Crearea tabelului TRATAMENTE

Am creat secvența TRATAMENTE_idTratament_SEQ pentru autoincrementare.

```
CREATE SEQUENCE TRATAMENTE_idTratament_SEQ MAXVALUE 999999 INCREMENT BY 1 START WITH 100000 NOCACHE ORDER NOCYCLE;

CREATE TABLE TRATAMENTE (
   idTratament NUMBER(6, 0)
      CONSTRAINT TRATAMENTE_idTratament_PK PRIMARY KEY,
   codBoala NUMBER(3, 0)
      CONSTRAINT TRATAMENTE_codBoala_FK REFERENCES BOLI(codBoala) ON DELETE CASCADE
      CONSTRAINT TRATAMENTE_codBoala_NN NOT NULL,
   perioadaAdministrare NUMBER(3, 0)
      CONSTRAINT TRATAMENTE_perioadaAdministrare_NN NOT NULL,
   indicatii VARCHAR2(250 CHAR)
);
```

5.3.15. Crearea tabelului MEDICAMENTE

Am creat secvența MEDICAMENTE_idMedicament_SEQ pentru autoincrementare.

```
CREATE SEQUENCE MEDICAMENTE_idMedicament_SEQ MAXVALUE 99999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER NOCYCLE;

CREATE TABLE MEDICAMENTE (
   idMedicament NUMBER(5, 0)
      CONSTRAINT MEDICAMENTE_idMedicament_PK PRIMARY KEY,
   numeMedicament VARCHAR2(30 CHAR)
      CONSTRAINT MEDICAMENTE_numeMedicament_NN NOT NULL,
   tipMedicament VARCHAR2(30 CHAR)
      CONSTRAINT MEDICAMENTE_tipMedicament_NN NOT NULL,
```

```
    dozaUnitate NUMBER(4, 0)
        CONSTRAINT MEDICAMENTE_dozaUnitate_NN NOT NULL
);
```

5.3.16. Crearea tabelului TRATAMENTE_ADMINISTRATE

```
CREATE TABLE TRATAMENTE_ADMINISTRATE  (
    idTratament NUMBER(6, 0)
        CONSTRAINT  TRAT_ADM_idTratament_FK  REFERENCES  TRATAMENTE(idTratament)
ON DELETE CASCADE,
    idMedicament NUMBER(5, 0)
        CONSTRAINT TRAT_ADM_idMedicament_FK REFERENCES
MEDICAMENTE(idMedicament) ON DELETE CASCADE,
    doza NUMBER(1, 0)
        CONSTRAINT TRAT_ADM_doza_NN NOT NULL,
    CONSTRAINT TRAT_ADM_PK PRIMARY KEY(idTratament, idMedicament)
);
```

5.4. Introducerea datelor în baza de date

Dat fiind faptul că baza de date conține un număr mare de intrări, mai jos vor apărea doar câteva exemple de populări ale bazei de date, în fiecare tabel.

5.4.1. Introducerea datelor în tabelul JUDETE

```
INSERT INTO JUDETE (codJudet, numeJudet) VALUES (JUDETE_codJudet_SEQ.NEXTVAL, 'Alba');
INSERT INTO JUDETE (codJudet, numeJudet) VALUES (JUDETE_codJudet_SEQ.NEXTVAL, 'Arad');
INSERT INTO JUDETE (codJudet, numeJudet) VALUES (JUDETE_codJudet_SEQ.NEXTVAL, 'Arges');
INSERT INTO JUDETE (codJudet, numeJudet) VALUES (JUDETE_codJudet_SEQ.NEXTVAL, 'Bacau');
INSERT INTO JUDETE (codJudet, numeJudet) VALUES (JUDETE_codJudet_SEQ.NEXTVAL, 'Bihor');
INSERT INTO JUDETE (codJudet, numeJudet) VALUES (JUDETE_codJudet_SEQ.NEXTVAL, 'Bistrita-
Nasaud');
```

5.4.2. Introducerea datelor în tabelul LOCALITATI

```
INSERT INTO LOCALITATI (idLocalitate, numeLocalitate, codPostal, codJudet) VALUES (LOCALITATI_idLocalitate_SEQ.NEXTVAL, 'Alba Iulia', LOCALITATI_codPostal_SEQ.NEXTVAL, 1);

INSERT INTO LOCALITATI (idLocalitate, numeLocalitate, codPostal, codJudet) VALUES (LOCALITATI_idLocalitate_SEQ.NEXTVAL, 'Arad', LOCALITATI_codPostal_SEQ.NEXTVAL, 2);

INSERT INTO LOCALITATI (idLocalitate, numeLocalitate, codPostal, codJudet) VALUES (LOCALITATI_idLocalitate_SEQ.NEXTVAL, 'Pitesti', LOCALITATI_codPostal_SEQ.NEXTVAL, 3);

INSERT INTO LOCALITATI (idLocalitate, numeLocalitate, codPostal, codJudet) VALUES (LOCALITATI_idLocalitate_SEQ.NEXTVAL, 'Bacau', LOCALITATI_codPostal_SEQ.NEXTVAL, 4);

INSERT INTO LOCALITATI (idLocalitate, numeLocalitate, codPostal, codJudet) VALUES (LOCALITATI_idLocalitate_SEQ.NEXTVAL, 'Oradea', LOCALITATI_codPostal_SEQ.NEXTVAL, 5);

INSERT INTO LOCALITATI (idLocalitate, numeLocalitate, codPostal, codJudet) VALUES (LOCALITATI_idLocalitate_SEQ.NEXTVAL, 'Bistrita', LOCALITATI_codPostal_SEQ.NEXTVAL, 6);
```

5.4.3. Introducerea datelor în tabelul SPITALE

```
INSERT INTO SPITALE (idSpital, numeSpital, idLocalitate, strada, numar, telefon, email)

VALUES(SPITALE_idSpital_SEQ.NEXTVAL, 'Spitalul Judetean de Urgenta Alba', 1, 'Bd. Republicii', 25, '0235001002', 'contact@sjab.ro');

INSERT INTO SPITALE (idSpital, numeSpital, idLocalitate, strada, numar, telefon, email)

VALUES(SPITALE_idSpital_SEQ.NEXTVAL, 'Spitalul Judetean de Urgenta Arad', 2, 'Sos. Nationala', 123, '0235002003', 'contact@sjar.ro');

INSERT INTO SPITALE (idSpital, numeSpital, idLocalitate, strada, numar, telefon, email)

VALUES(SPITALE_idSpital_SEQ.NEXTVAL, 'Spitalul Judetean de Urgenta Arges', 3, 'Castanilor', 25, '0235003004', 'contact@sjag.ro');

INSERT INTO SPITALE (idSpital, numeSpital, idLocalitate, strada, numar, telefon, email)

VALUES(SPITALE_idSpital_SEQ.NEXTVAL, 'Spitalul Judetean de Urgenta Bacau', 4, 'Libertatii', 25, '0235004005', 'contact@sjbc.ro');

INSERT INTO SPITALE (idSpital, numeSpital, idLocalitate, strada, numar, telefon, email)

VALUES(SPITALE_idSpital_SEQ.NEXTVAL, 'Spitalul Judetean de Urgenta Bihor', 5, 'Stefan cel Mare', 25, '0235005006', 'contact@sjbh.ro');
```

```
INSERT INTO SPITALE (idSpital, numeSpital, idLocalitate, strada, numar, telefon, email) VALUES(SPITALE_idSpital_SEQ.NEXTVAL, 'Spitalul Judetean de Urgenta Bistita-Nasaud', 6, '1 Decembrie', 25, '0235006007', 'contact@sjbn.ro');
```

5.4.4. Introducerea datelor în tabelul SECTII

```
INSERT INTO SECTII (codSectie, numeSectie) VALUES (SECTII_codSectie_SEQ.NEXTVAL, 'Alergologie si imunologie clinica');

INSERT INTO SECTII (codSectie, numeSectie) VALUES (SECTII_codSectie_SEQ.NEXTVAL, 'ATI');

INSERT INTO SECTII (codSectie, numeSectie) VALUES (SECTII_codSectie_SEQ.NEXTVAL, 'Cardiologie');

INSERT INTO SECTII (codSectie, numeSectie) VALUES (SECTII_codSectie_SEQ.NEXTVAL, 'Chirurgie');

INSERT INTO SECTII (codSectie, numeSectie) VALUES (SECTII_codSectie_SEQ.NEXTVAL, 'Endocrinologie');

INSERT INTO SECTII (codSectie, numeSectie) VALUES (SECTII_codSectie_SEQ.NEXTVAL, 'Epiedemiologie');
```

5.4.5. Introducerea datelor în tabelul SECTII SPITALE

```
INSERT INTO SECTII_SPITALE (idSectie, idSpital, codSectie, corp, etaj, telefon, email) VALUES (SECTII_SPITALE_idSectie_SEQ.NEXTVAL, 10001, 1, 'A', 0, '0235044041', 'alergie@sjab.ro');

INSERT INTO SECTII_SPITALE (idSectie, idSpital, codSectie, corp, etaj, telefon, email) VALUES (SECTII_SPITALE_idSectie_SEQ.NEXTVAL, 10001, 2, 'A', 1, null, 'ati@sjab.ro');

INSERT INTO SECTII_SPITALE (idSectie, idSpital, codSectie, corp, etaj, telefon, email) VALUES (SECTII_SPITALE_idSectie_SEQ.NEXTVAL, 10002, 4, 'A', 1, null, null);

INSERT INTO SECTII_SPITALE (idSectie, idSpital, codSectie, corp, etaj, telefon, email) VALUES (SECTII_SPITALE_idSectie_SEQ.NEXTVAL, 10002, 10, 'B', 1, null, null);

INSERT INTO SECTII_SPITALE (idSectie, idSpital, codSectie, corp, etaj, telefon, email) VALUES (SECTII_SPITALE_idSectie_SEQ.NEXTVAL, 10003, 17, null, 1, '0235054041', null);

INSERT INTO SECTII_SPITALE (idSectie, idSpital, codSectie, corp, etaj, telefon, email) VALUES (SECTII_SPITALE_idSectie_SEQ.NEXTVAL, 10004, 4, null, 0, null, null);
```

5.4.6. Introducerea datelor în tabelul FUNCTII

```
INSERT INTO FUNCTII (codFunctie, numeFunctie) VALUES (FUNCTII_codFunctie_SEQ.NEXTVAL, 'Medic chirurg');

INSERT INTO FUNCTII (codFunctie, numeFunctie) VALUES (FUNCTII_codFunctie_SEQ.NEXTVAL, 'Medic primar');

INSERT INTO FUNCTII (codFunctie, numeFunctie) VALUES (FUNCTII_codFunctie_SEQ.NEXTVAL, 'Medic anestezist');

INSERT INTO FUNCTII (codFunctie, numeFunctie) VALUES (FUNCTII_codFunctie_SEQ.NEXTVAL, 'Medic rezident');

INSERT INTO FUNCTII (codFunctie, numeFunctie) VALUES (FUNCTII_codFunctie_SEQ.NEXTVAL, 'Medic specialist');
```

5.4.7. Introducerea datelor în tabelul DOCTORI

```
INSERT INTO DOCTORI (codParafa, nume, prenume, cnp, telefon, email, codFunctie, idSectie)

VALUES (DOCTORI_codParafa_SEQ.NEXTVAL, 'Popescu', 'Ioana', '2980807452812', '0741000000', 'medic1@gmail.com', 1, 100001);

INSERT INTO DOCTORI (codParafa, nume, prenume, cnp, telefon, email, codFunctie, idSectie)

VALUES (DOCTORI_codParafa_SEQ.NEXTVAL, 'Radu', 'Adrian', '1890606340497', '0741000001', 'medic2@gmail.com', 2, 100002);

INSERT INTO DOCTORI (codParafa, nume, prenume, cnp, telefon, email, codFunctie, idSectie)

VALUES (DOCTORI_codParafa_SEQ.NEXTVAL, 'Maxim', 'Gabriel', '1910817408864', '0741000002', 'medic3@gmail.com', 3, 100003);

INSERT INTO DOCTORI (codParafa, nume, prenume, cnp, telefon, email, codFunctie, idSectie)

VALUES (DOCTORI_codParafa_SEQ.NEXTVAL, 'Tanase', 'Maria', '2890325319781', '0741000003', 'medic4@gmail.com', 4, 100004);

INSERT INTO DOCTORI (codParafa, nume, prenume, cnp, telefon, email, codFunctie, idSectie)

VALUES (DOCTORI_codParafa_SEQ.NEXTVAL, 'Rascanu', 'Gabriela', '2780425329582', '0741000004', 'medic5@gmail.com', 5, 100005);

INSERT INTO DOCTORI (codParafa, nume, prenume, cnp, telefon, email, codFunctie, idSectie)

VALUES (DOCTORI_codParafa_SEQ.NEXTVAL, 'Popa', 'Marcel', '1870620178249', '0741000005', 'medic6@gmail.com', 1, 100006);
```

5.4.8. Introducerea datelor în tabelul PACIENTI

```
INSERT INTO PACIENTI (idPacient, nume, prenume, cnp, strada, numar, localitate, telefon, email, asigurat, dataInternare, dataExternare)

VALUES (PACIENTI_idPacient_SEQ.NEXTVAL, 'Dimitrina', 'Gratian', '1600501152003', 'Aleea Vlad Tepes', 250, 'Radauti', '0710590770', null, 0, TO_DATE('25/01/2020', 'DD/MM/YYYY'), null);

INSERT INTO PACIENTI (idPacient, nume, prenume, CNP, strada, numar, localitate, telefon, email, asigurat, dataInternare, dataExternare)

VALUES (PACIENTI_idPacient_SEQ.NEXTVAL, 'Pompilia', 'Catinca', '2580716089546', 'Splaiul Jiului', 203, 'Mun. Pitesti', '0710860757', null, 0, TO_DATE('31/01/2020', 'DD/MM/YYYY'), null);

INSERT INTO PACIENTI (idPacient, nume, prenume, CNP, strada, numar, localitate, telefon, email, asigurat, dataInternare, dataExternare)

VALUES (PACIENTI_idPacient_SEQ.NEXTVAL, 'Natasa', 'Tiberiu', '1510637943602', 'Splaiul Ion Creanga', 52, 'Galati', '0722330302', null, 1, TO_DATE('04/02/2020', 'DD/MM/YYYY'), null);

INSERT INTO PACIENTI (idPacient, nume, prenume, CNP, strada, numar, localitate, telefon, email, asigurat, dataInternare, dataExternare)

VALUES (PACIENTI_idPacient_SEQ.NEXTVAL, 'Geanina', 'Roza', '6001269995931', 'Splaiul Eroilor', 44, 'Mun. Bragadiru', '0792060082', null, 0, TO_DATE('08/04/2020', 'DD/MM/YYYY'), null);

INSERT INTO PACIENTI (idPacient, nume, prenume, CNP, strada, numar, localitate, telefon, email, asigurat, dataInternare, dataExternare)

VALUES (PACIENTI_idPacient_SEQ.NEXTVAL, 'Mihail', 'George', '1760422603709', 'Aleea Padurii', 49, 'Ardud', '0743717625', null, 1, TO_DATE('12/04/2020', 'DD/MM/YYYY'), null);

INSERT INTO PACIENTI (idPacient, nume, prenume, CNP, strada, numar, localitate, telefon, email, asigurat, dataInternare, dataExternare)

VALUES (PACIENTI_idPacient_SEQ.NEXTVAL, 'Paul', 'Achim', '1781214391087', 'Aleea Piersicului', 79, 'Mun. Isaccea', '0264226364', null, 1, TO_DATE('17/04/2020', 'DD/MM/YYYY'), null);
```

5.4.9. Introducerea datelor în tabelul CONSULTATII

```
INSERT INTO CONSULTATII (idConsultatie, codParafa, idPacient, dataConsultatie) VALUES (CONSULTATII_idConsultatie_SEQ.NEXTVAL, 100000, 1, TO_DATE('25/01/2020', 'DD/MM/YYYY')+1);

INSERT INTO CONSULTATII (idConsultatie, codParafa, idPacient, dataConsultatie) VALUES (CONSULTATII_idConsultatie_SEQ.NEXTVAL, 100066, 2, TO_DATE('31/01/2020', 'DD/MM/YYYY')+1);

INSERT INTO CONSULTATII (idConsultatie, codParafa, idPacient, dataConsultatie) VALUES (CONSULTATII_idConsultatie_SEQ.NEXTVAL, 100001, 3, TO_DATE('04/02/2020', 'DD/MM/YYYY')+1);
```

```
INSERT INTO CONSULTATII (idConsultatie, codParafa, idPacient, dataConsultatie) VALUES (CONSULTATII_idConsultatie_SEQ.NEXTVAL, 100010, 4, TO_DATE('08/04/2020', 'DD/MM/YYYY')+1);

INSERT INTO CONSULTATII (idConsultatie, codParafa, idPacient, dataConsultatie) VALUES (CONSULTATII_idConsultatie_SEQ.NEXTVAL, 100022, 5, TO_DATE('12/04/2020', 'DD/MM/YYYY')+1);

INSERT INTO CONSULTATII (idConsultatie, codParafa, idPacient, dataConsultatie) VALUES (CONSULTATII_idConsultatie_SEQ.NEXTVAL, 100028, 6, TO_DATE('17/04/2020', 'DD/MM/YYYY')+1);
```

5.4.10. Introducerea datelor în tabelul BOLI

```
INSERT INTO BOLI (codBoala, denumireBoala) VALUES(BOLI_codBoala_SEQ.NEXTVAL, 'Alergie');

INSERT INTO BOLI (codBoala, denumireBoala) VALUES(BOLI_codBoala_SEQ.NEXTVAL, 'Intoleranta');

INSERT INTO BOLI (codBoala, denumireBoala) VALUES(BOLI_codBoala_SEQ.NEXTVAL, 'Insuficienta respiratorie acuta');

INSERT INTO BOLI (codBoala, denumireBoala) VALUES(BOLI_codBoala_SEQ.NEXTVAL, 'COVID-19');

INSERT INTO BOLI (codBoala, denumireBoala) VALUES(BOLI_codBoala_SEQ.NEXTVAL, 'Boala coronariana');

INSERT INTO BOLI (codBoala, denumireBoala) VALUES(BOLI_codBoala_SEQ.NEXTVAL, 'AVC');
```

5.4.11. Introducerea datelor în tabelul SIMPTOME

```
INSERT INTO SIMPTOME (codSimptom, denumireSimptom) VALUES
(SIMPTOME_codSimptom_SEQ.NEXTVAL, 'Febra cu fiori reci');

INSERT INTO SIMPTOME (codSimptom, denumireSimptom) VALUES
(SIMPTOME_codSimptom_SEQ.NEXTVAL, 'Febra persistenta');

INSERT INTO SIMPTOME (codSimptom, denumireSimptom) VALUES
(SIMPTOME_codSimptom_SEQ.NEXTVAL, 'Cefalee');

INSERT INTO SIMPTOME (codSimptom, denumireSimptom) VALUES
(SIMPTOME_codSimptom_SEQ.NEXTVAL, 'Durere acuta');

INSERT INTO SIMPTOME (codSimptom, denumireSimptom) VALUES
(SIMPTOME_codSimptom_SEQ.NEXTVAL, 'Sincopa');

INSERT INTO SIMPTOME (codSimptom, denumireSimptom) VALUES
(SIMPTOME_codSimptom_SEQ.NEXTVAL, 'Senilitate');
```

5.4.12. Introducerea datelor în tabelul DIAGNOSTICE

```
INSERT INTO DIAGNOSTICE (idDiagnostic, idConsultatie, codBoala) VALUES
(DIAGNOSTICE_idDiagnostic_SEQ.NEXTVAL, 1, 1);

INSERT INTO DIAGNOSTICE (idDiagnostic, idConsultatie, codBoala) VALUES
(DIAGNOSTICE_idDiagnostic_SEQ.NEXTVAL, 2, 2);

INSERT INTO DIAGNOSTICE (idDiagnostic, idConsultatie, codBoala) VALUES
(DIAGNOSTICE_idDiagnostic_SEQ.NEXTVAL, 3, 3);

INSERT INTO DIAGNOSTICE (idDiagnostic, idConsultatie, codBoala) VALUES
(DIAGNOSTICE_idDiagnostic_SEQ.NEXTVAL, 4, 4);

INSERT INTO DIAGNOSTICE (idDiagnostic, idConsultatie, codBoala) VALUES
(DIAGNOSTICE_idDiagnostic_SEQ.NEXTVAL, 5, 5);

INSERT INTO DIAGNOSTICE (idDiagnostic, idConsultatie, codBoala) VALUES
(DIAGNOSTICE_idDiagnostic_SEQ.NEXTVAL, 6, 6);
```

5.4.13. Introducerea datelor în tabelul DIAGNOSTICE_SIMPTOME

```
INSERT INTO DIAGNOSTICE_SIMPTOME (idDiagnostic, codSimptom) VALUES (100000, 7);

INSERT INTO DIAGNOSTICE_SIMPTOME (idDiagnostic, codSimptom) VALUES (100000, 28);

INSERT INTO DIAGNOSTICE_SIMPTOME (idDiagnostic, codSimptom) VALUES (100001, 10);

INSERT INTO DIAGNOSTICE_SIMPTOME (idDiagnostic, codSimptom) VALUES (100001, 32);

INSERT INTO DIAGNOSTICE_SIMPTOME (idDiagnostic, codSimptom) VALUES (100002, 28);

INSERT INTO DIAGNOSTICE_SIMPTOME (idDiagnostic, codSimptom) VALUES (100002, 29);
```

5.4.14. Introducerea datelor în tabelul TRATAMENTE

```
INSERT INTO TRATAMENTE (idTratament, codBoala, perioadaAdministrare, indicatii)
VALUES(TRATAMENTE_idTratament_SEQ.NEXTVAL, 1, 5, null);

INSERT INTO TRATAMENTE (idTratament, codBoala, perioadaAdministrare, indicatii)
VALUES(TRATAMENTE_idTratament_SEQ.NEXTVAL, 2, 10, 'nu este recomandat formelor usoare');

INSERT INTO TRATAMENTE (idTratament, codBoala, perioadaAdministrare, indicatii)
VALUES(TRATAMENTE_idTratament_SEQ.NEXTVAL, 3, 7, null);

INSERT INTO TRATAMENTE (idTratament, codBoala, perioadaAdministrare, indicatii)
VALUES(TRATAMENTE_idTratament_SEQ.NEXTVAL, 4, 5, null);
```

```
INSERT INTO TRATAMENTE (idTratament, codBoala, perioadaAdministrare, indicatii) VALUES(TRATAMENTE_idTratament_SEQ.NEXTVAL, 5, 5, null);

INSERT INTO TRATAMENTE (idTratament, codBoala, perioadaAdministrare, indicatii) VALUES(TRATAMENTE_idTratament_SEQ.NEXTVAL, 6, 10, 'nu este recomandat formelor usoare');
```

5.4.15. Introducerea datelor în tabelul MEDICAMENTE

```
INSERT INTO MEDICAMENTE (idMedicament, numeMedicament, tipMedicament, dozaUnitate) VALUES(MEDICAMENTE_idMedicament_SEQ.NEXTVAL, 'Apixabanum', 'anticoagulant', 250);

INSERT INTO MEDICAMENTE (idMedicament, numeMedicament, tipMedicament, dozaUnitate) VALUES(MEDICAMENTE_idMedicament_SEQ.NEXTVAL, 'FLUOROURACIL ', 'perfuzie', 100);

INSERT INTO MEDICAMENTE (idMedicament, numeMedicament, tipMedicament, dozaUnitate) VALUES(MEDICAMENTE_idMedicament_SEQ.NEXTVAL, 'Eteri', 'anestezic', 50);

INSERT INTO MEDICAMENTE (idMedicament, numeMedicament, tipMedicament, dozaUnitate) VALUES(MEDICAMENTE_idMedicament_SEQ.NEXTVAL, 'Tetracicline', 'antibiotic', 1000);

INSERT INTO MEDICAMENTE (idMedicament, numeMedicament, tipMedicament, dozaUnitate) VALUES(MEDICAMENTE_idMedicament_SEQ.NEXTVAL, 'Cloramfenicol', 'antibiotic', 500);

INSERT INTO MEDICAMENTE (idMedicament, numeMedicament, tipMedicament, dozaUnitate) VALUES(MEDICAMENTE_idMedicament_SEQ.NEXTVAL, 'Ampicillinum', 'antibiotic', 250);
```

5.4.16. Introducerea datelor în tabelul TRATAMENTE_ADMINISTRATE

```
INSERT INTO TRATAMENTE_ADMINISTRATE (idTratament, idMedicament, doza) VALUES (100001, 27, 1);

INSERT INTO TRATAMENTE_ADMINISTRATE (idTratament, idMedicament, doza) VALUES (100002, 26, 1);

INSERT INTO TRATAMENTE_ADMINISTRATE (idTratament, idMedicament, doza) VALUES (100003, 25, 3);

INSERT INTO TRATAMENTE_ADMINISTRATE (idTratament, idMedicament, doza) VALUES (100004, 24, 1);

INSERT INTO TRATAMENTE_ADMINISTRATE (idTratament, idMedicament, doza) VALUES (100005, 23, 2);

INSERT INTO TRATAMENTE_ADMINISTRATE (idTratament, idMedicament, doza) VALUES (100006, 22, 1);
```