**ROMÂNIA**

MINISTERUL EDUCAŢIEI

INSPECTORATUL ȘCOLAR AL JUDEȚULUI BACĂU

**COLEGIUL NAȚIONAL „DIMITRIE CANTEMIR“, ONEŞTI**

Str. dr. Victor Babeș, nr.12, cod poștal 601119

Tel/Fax (+4)0234 311046

e-mail:contact@dcantemir.ro website: www.dcantemir.ro

**Nr. ....................../............................................**

LUCRARE PENTRU OBȚINEREA ATESTATULUI PROFESIONAL

Profil: real

Specializarea: matematică-informatică, intensiv informatică

Candidat: Luchian Eliza Raluca

Profesor îndrumător:

Piștea Daniela-Geanina

**MAI 2023**

**Temă proiect: Clinica medicală**

**Clinica**

**YDAHealth**

****

Cuprins

[I. Argument i](#_Toc134730882)

[I.1.Alegerea temei proiectului i](#_Toc134730883)

[II. Noțiuni teoretice pentru baze de date 3](#_Toc134730884)

[II.1.Date generale 3](#_Toc134730885)

[II.2. Entități. Instanțe. Atribute. Identificator unic 4](#_Toc134730886)

[II.3. Relaţii între entităţi 5](#_Toc134730887)

[II.3.1 Relațíi ierarhice 5](#_Toc134730888)

[II.3..2 Relațíi recursive 5](#_Toc134730889)

[II.4. Rezolvarea relațiilor many-to-many 9](#_Toc134730890)

[II.5. Cele trei forme de normalizare 10](#_Toc134730891)

[II.6. Mapare 12](#_Toc134730892)

[III. Modelarea bazei de date 13](#_Toc134730893)

[III.1. Modelare conceptuală 13](#_Toc134730894)

[III.1.1. Scenariu 13](#_Toc134730895)

[III.1.2.Reguli structurale și procedurale 14](#_Toc134730896)

[III.1.3.Rezolvare relații many-to-many 15](#_Toc134730897)

[III.1.4. Normalizări 17](#_Toc134730898)

[III.1.5. Relații prezente în ERD 19](#_Toc134730899)

[III.1.6. Modelul conceptual inițial și final 20](#_Toc134730900)

[III. 2 Modelarea fizică 21](#_Toc134730901)

[III.2.1. Maparea entităților. Maparea relațiilor 21](#_Toc134730902)

[III.2.2.Implementarea tabelelor în Oracle Application Express 23](#_Toc134730903)

[III.3 Interogări 29](#_Toc134730904)

[IV.Concluzii ii](#_Toc134730905)

[V.Bibliografie iii](#_Toc134730906)

# I. Argument

În prezent, bazele de date se strecoară în viețile noastre din ce în ce mai mult și stau în spatele multor sisteme, societăți și concepte din jurul nostru. Bazele de date sunt folosite pentru stocarea, întreținerea și accesarea oricărui tip de date, colectând informații despre oameni, locuri sau lucruri, fiind practic o colecție organizată de informații. Am ales astfel să îmi orientez realizarea proiectului folosindu-mă de bazele de date ca instrument de lucru, având o aplicabilitate vastă, de la companiile care folosesc datele stocate în baze de date pentru a lua decizii de afaceri informate, la clienții acestora a căror informații trebuie înregistrate, păstrate pentru a putea fi ulterior procesate și chiar până la platformele de social media unde avem nevoie de a stoca informații personale, sau despre utilizatori în idea recomandării conținutului media și îmbunătățirii experienței în mediul online. Datele sunt peste tot și așa cum acestea există, este nevoie ca și diferite sisteme de stocare a lor să fie concepute pentru a putea le putea organiza și folosi. Construirea unei baze de date este o activitate foarte amuzantă, care oferă celor ce o construiesc o senzație apropiată gândirii și schițării propriei afaceri, fiind totodată și o unealtă oportună pentru procesul de învățare al elevilor.

## I.1.Alegerea temei proiectului

Domeniul medical este unul fascinant, cu atât mai mult cât a devenit o situație cotidiană ca cele două științe -medicina și tehnologia- să se mixeze, folosindu-se una de alta pentru a crea diferite inovații, pentru a putea rezolva problemele societății moderne, încât s-ar putea spune că progresele realizate pe anumite ramuri nu ar fi fost posibil fără vreuna dintre cele două. La fel ca orice firmă, o clinică sau un cabinet medical necesită o gestiune foarte bună a datelor pe care le deține, de la funcționarea clinicii, până la pacienții doctorilor, controalele medicale nefiind singurele prioritare, ci și birocrația aferentă, prin gruparea informațiilor în mod favorabil. Există multe clinici care pot face progrese prin construirea sau monitorizarea întregii afaceri într-un mod tehnic, iar modelele conceptuale ușurează cooperarea dintre programatori și clienții lor, fiind o schemă potrivită pentru a schița dorințele întreprinzătorului și pentru programator în idea implementării cerințelor. Astfel, consider că alegerea construirii unei baze de date care să susțină funcționarea corectă a unei clinici aduce în vedere o situație reală, existentă în contextul afacerilor și o îmbunătățire domeniului medical la finalul zilei.

# II. Noțiuni teoretice pentru baze de date

## II.1.Date generale

O bază de date este o colecție organizată de date sau informații electronice stocate pe un computer. De obicei, bazele de date sunt gestionate de sistemul de gestionare a bazelor de date „DBMS”. Datele, DBMS și aplicațiile asociate sunt adunate pentru a forma un sistem de bază de date sau pur și simplu o bază de date.

Datele din cele mai obișnuite tipuri de baze de date sunt de obicei organizate în rânduri și coloane ale diferitelor tabele pentru a optimiza procesarea datelor și regăsirea datelor. Puteți accesa, gestiona, edita, actualiza, controla și organiza cu ușurință datele dvs. Cele mai multe baze de date utilizează limbajul de interogare structurat (SQL) pentru a scrie și a interoga datele.

Bazele de date au evoluat semnificativ de la introducerea lor la începutul anilor 1960. Deși simple, aceste sisteme erau inflexibile. Bazele de date relaționale au devenit populare în anii 1980, urmate de bazele de date orientate pe obiecte în anii 1990. Apoi, creșterea internetului și nevoia de procesare rapidă a datelor nestructurate au dat naștere bazelor de date NoSQL. Astăzi, bazele de date cloud și autonome revoluționează modul în care datele sunt colectate, stocate, gestionate și utilizate. Analiza datelor este uneori un proces dificil și consumator de timp, dar este un pas necesar.

Există multe tipuri diferite de baze de date, inclusiv baze de date relaționale, baze de date orientate pe obiecte și baze de date NoSQL, și pot fi utilizate într-o varietate de setări, inclusiv organizații de afaceri, științifice și guvernamentale.

În concluzie, putem evidenția câteva caracteristici ale ERD.

* Sunt instrumente de proiectare.
* Reprezentarea grafică a unui sistem de date.
* Furnizează un model conceptual general al bazei de date.
* Ajută utilizatorii să înțeleagă datele și relațiile dintre date.
* Implementare autonomă.

Principalele elemente care compun un ERD și convențiile aferente sunt descrise mai jos.

## II.2. Entități. Instanțe. Atribute. Identificator unic

În contextul modelării afacerii, o entitate poate fi un obiect, persoană, eveniment sau concept care are o semnificație semnificativă și trebuie să fie colectată și stocată într-o bază de date. Entitățile pot fi reale și concrete, cum ar fi: B. Cum ar fi clădiri, oameni sau abstracții. B. Activitate sau Proces. În modelul conceptual al bazei de date (ERD), entitățile sunt reprezentate prin dreptunghiuri rotunjite, iar numele lor (întotdeauna substantivele singulare) sunt scrise cu majuscule deasupra dreptunghiului. Fiecare entitate reprezintă o clasă de obiecte cu mai multe instanțe. O instanță este un obiect, persoană, eveniment sau altă entitate specifică din aceeași clasă. De exemplu, Elevul X din Anul 9 A al Liceului de Informatică din Orașul Y este o instanță a entității ELEV.

Atributele sunt informații specifice care ajută la identificarea, clasificarea, cuantificarea sau exprimarea stării unei instanțe a unei entități. Atributele sunt detalii specifice care trebuie cunoscute și stocate în baza de date. De exemplu, atributele entității ELEV sunt prenumele, prenumele, adresa, numărul de telefon, adresa de e-mail, data nașterii și așa mai departe. În modelul conceptual al bazei de date, atributele sunt scrise cu litere mici sub numele entității. Atributele sunt obligatorii sau opționale. Atributele obligatorii necesită o valoare pentru fiecare instanță a entității, iar atributele opționale pot să nu aibă o valoare pentru unele instanțe. În ERD, atributele obligatorii sunt marcate cu un asterisc \*, iar atributele opționale sunt marcate cu un cerc. Atributele care definesc în mod unic o instanță a unei entități sunt numite identificatori unici (UID). Un UID poate consta dintr-un singur atribut, cum ar fi: B. Codul numeric personal al entității ELEV sau o combinație de două sau mai multe atribute. Atributele care fac parte din UID suntîntotdeauna obligatorii și sunt marcate cu caracterul # în ERD.



Fig I.1. Entitate

## II.3. Relaţii între entităţi

### II.3.1 Relațíi ierarhice

### II.3..2 Relațíi recursive

O conexiune reprezintă o asociere sau o legătură între două sau mai multe entități, având o semnificație importantă pentru afacerea modelată. Această conexiune este bidirecțională și poate lega două entități sau o entitate cu ea însăși. De exemplu, elevii studiază mai multe discipline, iar ao disciplină este studiată de către elevi.

Orice relaţie este caracterizată de următoarele elemente:

- numele relaţiei

- opţionalitatea relaţiei

- gradul (cardinalitatea) relaţiei.

* Cardinalitatea relației este dată de numărul de instanțe ale entității din partea dreaptă a relației care pot intra în relație cu o instanță a entității din partea stângă a relației. Adică va trebui să răspundem la întrebări de genul: La câte echipe poate juca un jucător? Răspunsurile posibile sunt unul și numai unul, sau unul sau mai mulți.
* Relatiile sunt obligatorii sau optionale. Optionalitatea se poate determina raspunzand la intrebari ca de pilda: Trebuie ca fiecare angajat sa aiba o slujba? Este aceasta o relatie optionala sau obligatorie pentru angajat? Pot angajatii sa aiba mai multe slujbe?

Într-o diagramă de relații între entități, o relație este prezentată printr-o linie care leagă două entități. Deoarece relația este bidirecțională, linia care leagă cele două unități este formată din două segmente separate, câte unul pentru fiecare unitate. Un tip segmentar care provine dintr-o entiate indică opționalitatea relației dintre această entitate și entitatea de pe cealaltă parte a relației. Dacă acest segment este continuu, este o relație obligatorie, o linie întreruptă indică o relație opțională.

Modul în care o linie se termină spre o entitate este important. Dacă se termină printr-o linie simplă, înseamnă că o instanţă şi numai una a acestei entităţi este în relaţie cu o instanţă a celeilalte entităţi.

Dacă linia se termină cu trei linii (picior de cioară) înseamnă că mai multe instanţe ale entităţii pot corespunde unei instanţe a celeilalte entităţi.

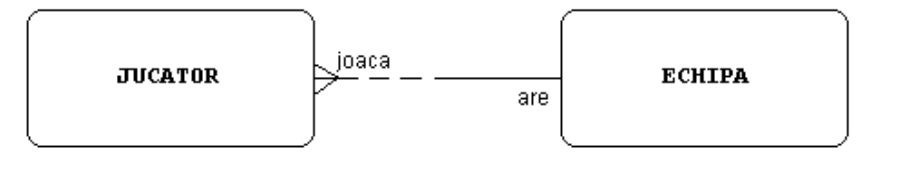


Fig I.2. Reprezentare relații

Variantele de relaţii ce pot exista între două entităţi sunt prezentate mai jos:

- relaţii one-to-one – acest tip de relaţie este destul de rar întâlnit. Uneori astfel de relaţii pot fi modelate transformând una dintre entităţi în atribut al celeilalte entităţi.

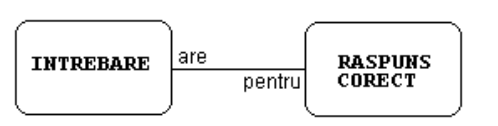


Fig I.2.1 Relație one-to-one

- relaţii one-to-many – sunt cele mai întâlnite tipuri de relaţii.

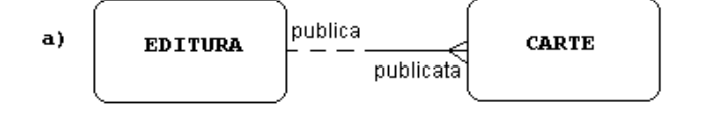


Fig I.2.2 Relație one-to-many

-relaţii many-to-many – aceste tipuri de relaţii apar în prima fază a proiectării bazei de date, însă ele trebuie să fie ulterior eliminate.

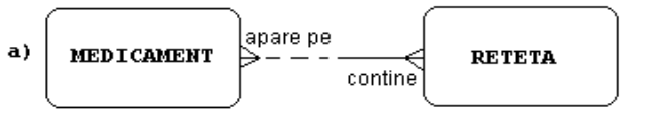


Fig I.2.3 Relație many-to-many

3.1

O relație ierarhică într-o bază de date se referă la un model de date în care datele sunt organizate într-o structură arborescentă, fiecare nod părinte având unul sau mai multe noduri copil. Acest tip de relație se mai numește și relație părinte-copil. Într-o bază de date ierarhică, fiecare înregistrare este organizată într-o structură arborescentă, cu un nod rădăcină în partea de sus și noduri copil la nivelurile inferioare. Fiecare nod din ierarhie este conectat la unul sau mai multe noduri copil, dar fiecare nod copil poate avea doar un nod părinte.

Acest tip de model de date este utilizat în mod obișnuit în aplicațiile care au o structură ierarhică naturală. B. Sistem de fișiere, organigramă sau categorii de produse. De exemplu, un sistem de vânzare cu amănuntul poate organiza produsele în categorii, poate crea subcategorii în cadrul fiecărei categorii și poate crea relații ierarhice între date.

Bazele de date ierarhice au avantajul de a fi simple și eficiente deoarece interogările pot fi optimizate pentru recuperarea rapidă a datelor într-o structură arborescentă. Cu toate acestea, adăugarea și eliminarea nodurilor poate fi dificilă și necesită modificări semnificative ale modelului de date, ceea ce poate limita flexibilitatea.

Spre deosebire de bazele de date ierarhice, bazele de date relaționale folosesc un model de date plat în care datele sunt organizate în tabele și legate prin relații definite prin chei primare și externe. Acest model de date este mai flexibil și poate găzdui relații complexe între entitățile de date, dar este posibil să nu fie eficient pentru anumite tipuri de interogări de date.

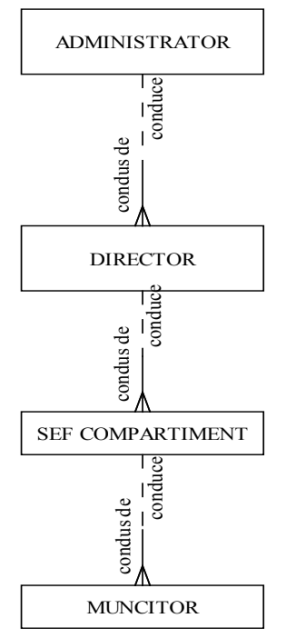


Fig I.3. Relație ierarhică

3.2.

Relațiile recursive în bazele de date sunt relații între o entitate și ea însăși. Acestea sunt adesea folosite în cazul structurilor ierarhice, cum ar fi arborii genealogici, organigramele și alte structuri similare.

În relațiile recursive, o entitate poate fi legată de alta entitate prin intermediul unei legături de tipul "părinte-copil". Aceasta înseamnă că entitatea părinte poate avea mai mulți copii, dar fiecare copil poate avea un singur părinte. De exemplu, o persoană poate fi părintele altei persoane, care poate fi părintele altei persoane și așa mai departe. În acest caz, entitatea "persoană" este legată de ea însăși prin intermediul relației "părinte-copil".

Utilizarea relațiilor recursive poate fi utilă în cazul în care este necesară organizarea și stocarea datelor ierarhice. Cu toate acestea, utilizarea lor poate fi uneori dificilă din punct de vedere al interogărilor, deoarece trebuie să fie luate în considerare toate nivelurile ierarhice în interogări.

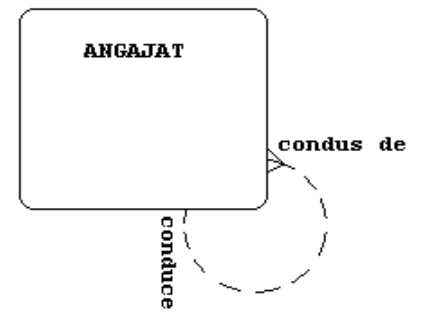


Fig I.4. Relație recursivă

## II.4. Rezolvarea relațiilor many-to-many

După cum am precizat mai devreme relaţiile many-to-many pot apărea într-o primă fază a proiectării bazei de date însă ele nu au voie să apară în schema finală. Rezolvarea unei relaţii many-to-many constă în introducerea unei noi entităţi numită entitate de intersecţie, pe care o legăm de entităţile originale prin câte o relaţie one-to-many.

Paşii în rezolvarea unei relaţii many-to-many sunt următorii:

* se găseşte entitatea de intersecţie, pentru exemplul nostru vom introduce entitatea INSCRIERE.
* crearea noilor relaţii

o opţionalitatea: relaţiile care pleacă din entitatea de intersecţie sunt întotdeauna obligatorii în această parte. În partea dinspre entităţile originale, relaţiile vor păstra

opţionalitatea relaţiilor iniţiale.

o cardinalitatea: ambele relaţii sunt de tip one-to-many, iar partea cu many va fi întotdeauna înspre entitatea de intersecţie.

o numele noilor relaţii.

* adăugarea de atribute în cadrul entităţii de intersecţie, dacă acestea există În exemplul nostru ne poate interesa să zicem data la care s-a înscris un student la un curs, data la care a finalizat cursul precum şi nota obţinută la sfârşitul cursului.
* stabilirea identificatorului unic pentru entitatea de intersecţie: dacă entitatea de intersecţie nu are un identificator unic propriu, atunci acesta se poate forma din identificatorii unici ai entităţilor iniţiale la care putem adăuga atribute ale entităţii de intersecţie. În exemplul nostru, identificatorul unic al entităţii de intersecţie este format din id-ul studentului, id-ul cursului şi data înscrierii la curs. Faptul că identificatorul unic al unei entităţi preia identificatorul unic din altă entitate cu care este legată este reprezentat grafic prin bararea relaţiei respective, înspre entitatea care preia UID-ul celeilalte entităţi.

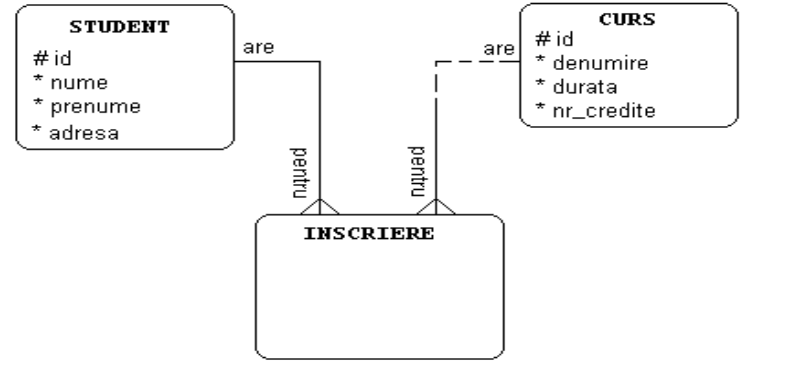


Fig I.5. Rezolvare relație many-to-many între STUDENT și CURS.

## II.5. Cele trei forme de normalizare

Prima formă normală (1NF) este o regulă de bază pentru modelarea datelor în bazele de date relaționale. Aceasta specifică că o relație trebuie să fie organizată astfel încât fiecare valoare atomică să fie stocată într-o singură celulă a unei tabele.Mai precis, o relație este considerată în 1NF dacă fiecare coloană conține numai valori atomice, iar nicio coloană nu poate conține valori multiple sau grupuri de valori. De asemenea, fiecare înregistrare trebuie să fie unică și să aibă o cheie primară unică pentru identificare.

De exemplu, să presupunem că avem o tabelă "Comenzi" care conține coloanele "ID comandă", "Produs", "Cantitate" și "Preț total". Pentru a respecta 1NF, ar trebui să ne asigurăm că fiecare înregistrare din tabelă are o singură valoare pentru fiecare dintre aceste coloane și că nu există grupuri de valori pentru produsele comandate sau cantitățile aferente. În general, 1NF este considerată prima etapă importantă în proiectarea bazei de date, deoarece permite structurarea datelor într-un mod coerent și permite utilizarea mai eficientă a informațiilor stocate într-o bază de date.

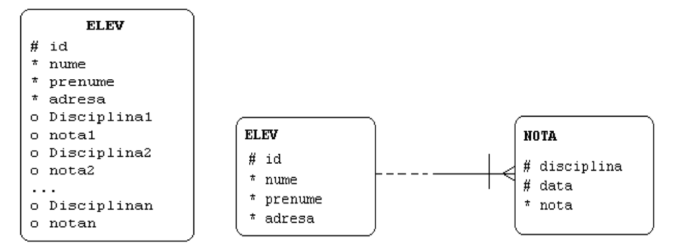


Fig I.6.1 Prima formă de normalizare

A doua formă normală (2NF) este un proces de normalizare a bazelor de date care are ca scop eliminarea redundanței și a dependențelor dintre atributele ne-cheie dintr-o tabelă.

Un tabel se află în 2NF dacă este deja în 1NF și toate atributele ne-cheie depind de întreaga cheie primară a tabelului. Cu alte cuvinte, dacă o tabelă are chei primare compuse, 2NF necesită ca fiecare atribut ne-cheie să depindă de întreaga cheie compusă, nu doar de o parte a acesteia. 2NF poate fi obținută prin dezmembrarea tabelei în altele mai mici, pe baza dependențelor funcționale, astfel încât fiecare tabel să reprezinte o singură entitate cu o cheie primară unică. Rezultatul este că datele nu mai sunt duplicate în mai multe rânduri și tabele, ceea ce duce la un stocare mai eficientă și la o întreținere mai ușoară a bazei de date.

De exemplu, să considerăm o tabelă care listează comenzile clienților, cu coloane pentru ID-ul comenzii, ID-ul clientului, numele produsului și prețul produsului. Dacă numele și prețul produsului depind doar de ID-ul comenzii, atunci tabela nu se află în 2NF. Pentru a obține 2NF, tabela ar trebui să fie împărțită în două tabele separate, una pentru comenzile și una pentru produsele clienților, cu o relație cheie străină care le leagă.

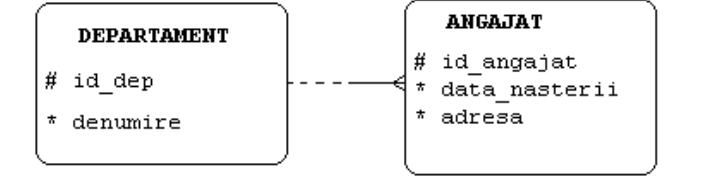
 

Fig I.6.2 A doua formă de normalizare

A treia formă normală (3NF) este o regulă de normalizare a bazei de date relaționale care se concentrează pe eliminarea dependențelor tranzitive din baza de date. O dependență tranzitivă apare atunci când există o relație indirectă între două sau mai multe coloane dintr-o tabelă prin intermediul unei alte coloane. Mai precis, atunci când o coloană dintr-o tabelă este dependentă de o altă coloană care, la rândul său, este dependentă de o altă coloană, aceasta poate fi considerată o dependență tranzitivă.

De exemplu, să presupunem că avem o tabelă "Comenzi" care conține coloanele "ID comandă", "Nume client", "Adresă client" și "Cod poștal". În cazul în care adresa clientului și codul poștal sunt dependente de numele clientului și nu de ID-ul comenzii, atunci avem o dependență tranzitivă între aceste coloane.

Pentru a respecta 3NF, ar trebui să separăm coloanele dependente tranzitiv într-o altă tabelă și să le legăm prin intermediul unei chei externe. În acest caz, ar trebui să avem o tabelă "Clienti" care conține coloanele "Nume client", "Adresă client" și "Cod poștal", iar tabelul "Comenzi" ar trebui să conțină doar coloanele "ID comandă" și "Nume client", legate prin intermediul unei chei externe.

Astfel, eliminând dependențele tranzitive din baza de date, putem îmbunătăți performanța și eficiența bazei de date, reducând redundanța și evitând anomalii de actualizare și ștergere care ar putea afecta integritatea datelor.

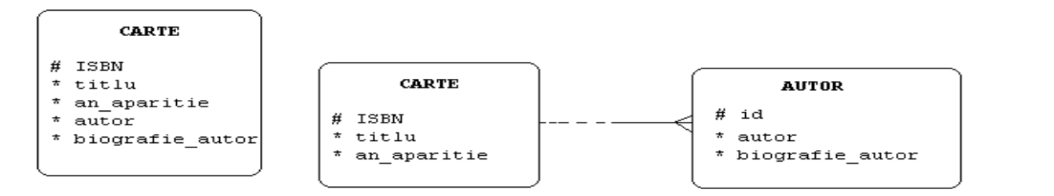


Fig I.6.3 A treia formă de normalizare

## II.6. Mapare

Mapping, sau mapare în limba română, este procesul prin care se stabilește o relație între două seturi de date, astfel încât să se poată transfera informația între ele.

În contextul bazelor de date, mapping-ul se referă la identificarea legăturilor dintre tabelele din baza de date și stabilirea modului în care acestea sunt conectate între ele. Acest proces este important pentru a permite interogarea datelor din mai multe tabele și pentru a asigura coerența datelor în baza de date.

Există mai multe tipuri de mapping utilizate în bazele de date, inclusiv mapping-ul obiect-relațional, care se referă la procesul de transformare a obiectelor din codul sursă al aplicației în date stocate în baza de date și invers. De asemenea, există și mapping-ul relațional-obiect (ROM), care se referă la procesul de transformare a datelor stocate în baza de date în obiecte utilizate de codul sursă al aplicației.

Mapping-ul poate fi realizat manual sau prin intermediul unor unelte specializate, cum ar fi ORM sau ROM. Indiferent de metoda folosită, este important ca mapping-ul să fie realizat cu atenție pentru a evita erorile și pentru a asigura că baza de date este coerentă și eficientă în utilizare.

# III. Modelarea bazei de date

## III.1. Modelare conceptuală

### III.1.1. Scenariu

Cabinetul YDA-health solicită printr-un contract crearea unei baze de date care să asigure gestionarea corectă a datelor pentru buna funcționare a clinicii. Deținătorul clinicii, doctorul Josef are nevoie de o bază de date în care fiecare pacient să fie identificat printr-un ID conceput de cabinet și CNP-ul pacientului. Se memorează despre aceștia numele și prenumele, vârsta și opțional adresa acestuia și sexul. Fiecărui pacient îi este atribuit un doctor sau mai mulți, la fel și în cazul doctorilor, care pot consulta unul sau mai mulți pacienți.

În cadrul clinicii lucrează  mulți doctori cu specializări diferite. Despre aceștia se memorează numele, domeniul specializării, experiența de muncă de când au obținut specializarea și data angajării în cadrul clinicii, numărul de telefon, salariul și opțional prima. Fiecare  doctor are la dispoziție un grup de asistenți și infirmieri care îl ajută în cadrul consultului clinic. Despre fiecare asistent se cunoaște gradul și funcția deținută(exemplu: asistent șef, asistent general), numele, vârsta, număr de telefon, salariu, prima în cazul asistenților și la fel ca doctorii o insignă proprie dată de cabinet prin care se identifică. Aceleași lucruri înafară de funcție se rețin și în cazul infirmierilor.

În cadrul unui consult este posibil să se emită  unul sau mai multe tratamente pentru pacienți. Va fi nevoie de reținerea diagnosticului, data începerii administrării, codul de înregistrare a tratamentului ca UID principal și programul de administrare. Tratamentele trebuie neapărat să conțină una sau mai multe medicamente, dar medicamentele pot fi eliberate și fără tratament.  Anumite medicamente pot fi disponibile în cabinet și pentru acestea se cunoaște cantitatea disponibilă în cabinet, si gramajele sau miligramele în funcție de vârstă,  cantitatea ce trebuie comandată. Există distribuitori care pot furniza medicamente cabinetelor clinicii  și echipamente medicale, clinica neputând exista fără ei. Echipamentele au un nume de reținut și un număr deja prezent în cabinete. Despre furnizori se cunoaște că furnizează mai multor cabinete din cadrul clinicii, acestea putând avea și mai mulți furnizori. Reținem astfel id-ul firmei, nr de telefon și opțional adresa sediului, produsul si cantitatea produsul pe care îl livrează, împreună cu id-ul produsului.

Întreținerea necesită mai multe cheltuieli, care trebuie urmărite pentru calcularea bugetului. Se vor reține costurile facilităților(apă, curent, lumină), dar și ale produselor(medicamente, echipamente), comandate.

### III.1.2.Reguli structurale și procedurale

Există totuşi reguli pentru implementarea cărora va trebui să scriem programe speciale folosind limbaje specializate specifice SGBD-ului utilizat. Acest tip de reguli se numesc numite reguli procedurale. În Oracle acest tip de programe se vor scrie folosind limbajul PL/SQL şi se numesc declanşatoare (triggere).

Exemple posibile

Un pacient își poate face un card de clinică și să devină membru, lucru care va genera reduceri pentru consultații și anumite beneficii. De asemenea există anumite condiții de vârstă pentru a putea realiza înscrierea ca membru.

Furnizorii pot face reducerui și promoții pentru comenzi care depășesc un anumit număr de echipamente sau medicamente comandate.

Tipuri de atribute:

\* = mandatorii

* = opțional

# = UID, pentru identificare

Exemple din ERD:

#id\_pacient

\*prenume

* prima

Pentru ca modelul conceptual să fie complet se definesc reguli structurale (-indică tipuri de informații ce vor fi stocate și cum relaționează ele).

Exemple

În entitatea pacient atributul sex este optional. Completarea acestui câmp se realizează doar cu valori de tip caracter( F sau M), sau null în cazul necompletării. În entitatea medicamente există atributul concentrație, care este opțional, deoarece nu toate medicamentele au precizată concentrația, sau pur și simplu nu au așa ceva. Tot în această entitate există atributul untitate de măsură. Acesta poate fi completat fie cu numărul de bucăți în cazul pastilelor, sau cu numărul de mililitri în cazul siropurilor.

### III.1.3.Rezolvare relații many-to-many

Rezolvarea unei relaţii many-to-many constă în introducerea unei noi entităţi numită entitate de intersecţie, pe care o legăm de entităţile originale prin câte o relaţie one-to-many.

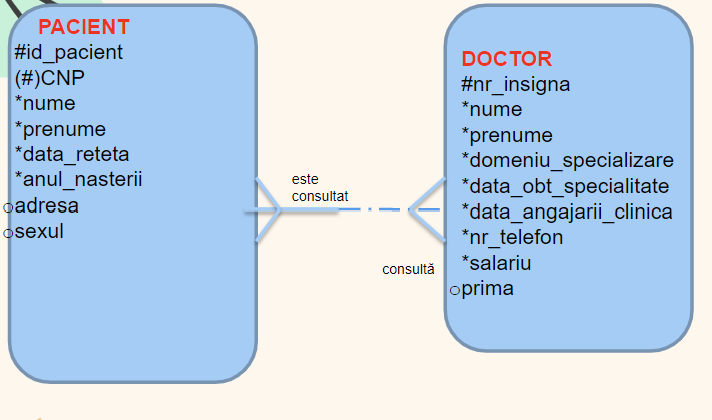


Fig I.1.1. Relație înainte de rezolvare

Pentru a rezolva relația dintre entitatea PACIENT și entitatea DOCTOR vom folosi entitatea de intersecție PROGRAMARE, pacientul si doctorul urmând să se întâlnească în urma unei programări, astfel evitându-se cazul în care un pacient ar fi atribuit în același timp mai multor doctori, sau alte greșeli de distribuire a doctorilor pentru pacienți.

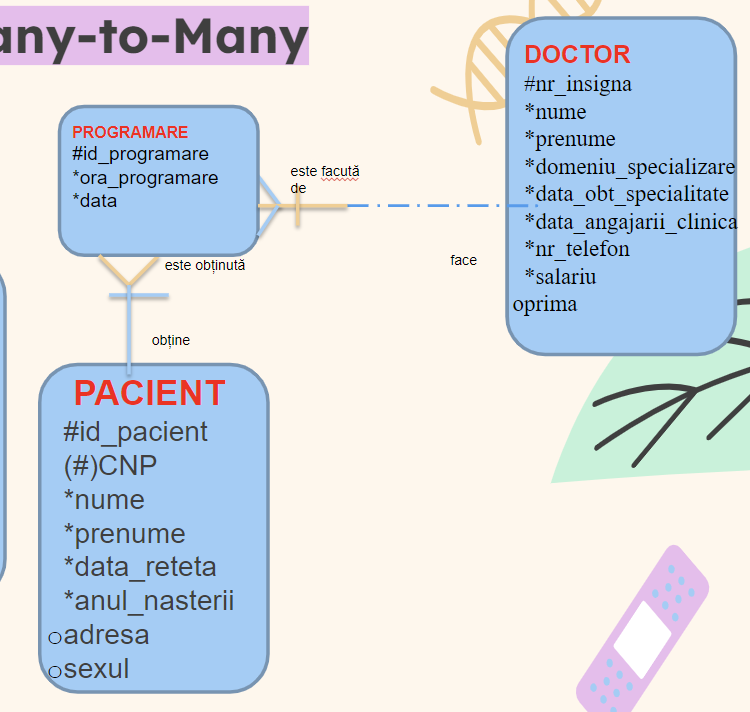


Fig I.1.2. Rezolvare

Pentru rezolvarea relației dintre entitățile CABINET și FURNIZOR este nevoie de entitatea de intersecție CONTRACT, orice achiziție sau vânzare realizată între anumiți furnizori și anumite cabinete se înregistrează și efectuează pe baza unui contract vânzare-cumpărare care să specifice data la care se emite și care va primi UID-urile furnizorilor și a cabinetelor.

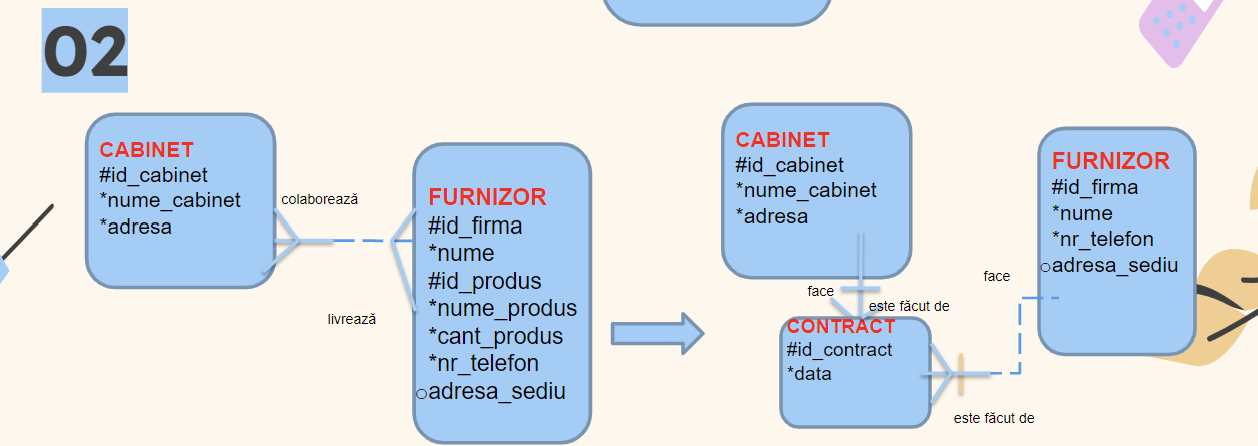


Fig I.2. Prima formă de normalizare

În cazul relației dintre entitatea PACIENT și entitatea TRATAMENT este nevoie de entitatea CONSULT care ajută la distribuirea corectă a tratamentelor care aparțin unui anumit consult, pentru a ști care pacient primește anumite tratamente și acestea în urma căror consulturi au fost atribuite.

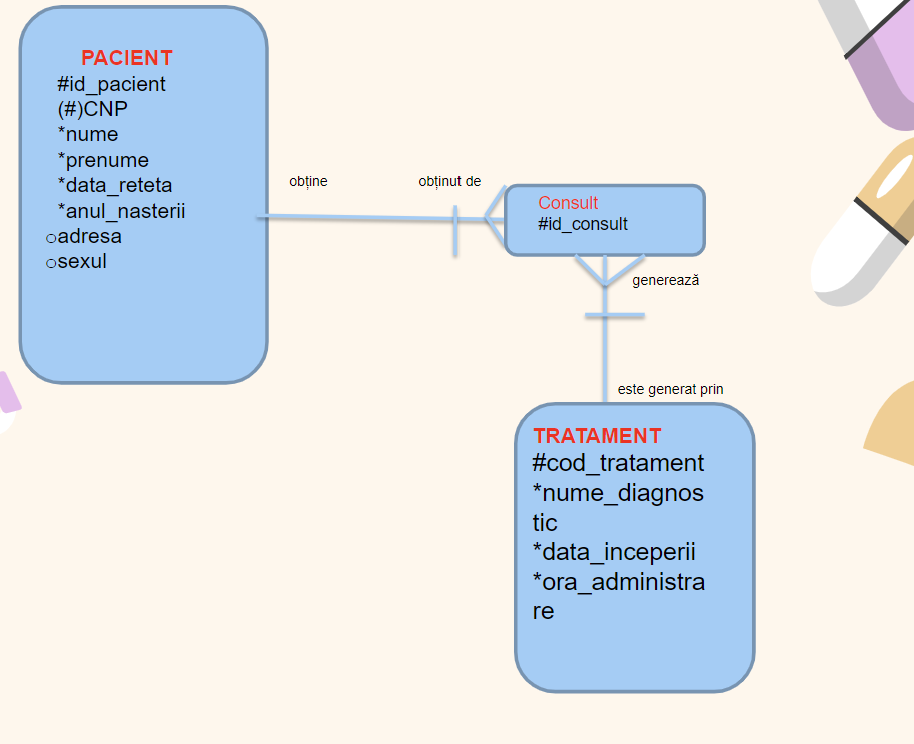


Fig I.3. Prima formă de normalizare

### III.1.4. Normalizări

Prima formă de normalizare presupune neexistarea atributelor cu valori multiple.

În cazul nostru este vorba despre pacient, care are atributul data\_reteta, care este problema. Un pacient poate avea mai multe rețete cu diferite scopuri. De aceea se recomandă crearea unei entități noi cu numele RETETA.

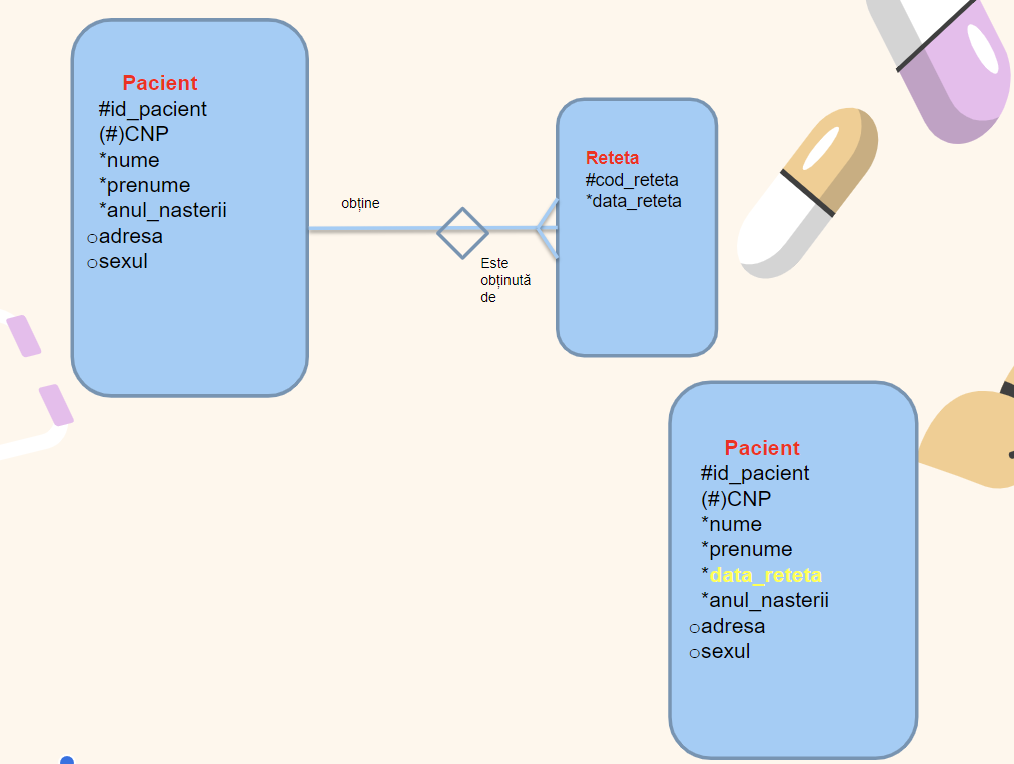


Fig II.1. Prima formă de normalizare

A doua formă de normalizare presupune ca toate atributele ce nu sunt UID-uri să depindă de UID-ul total. În cazul nostru este vorba despre id\_utilitate cu care atributele produselor nu au legătură. Cabinetul are cheltuieli pe utilități, dar utilitățile nu intră în categoria produselor ca marfă achiziționate, astfel nu este potrivit ca acest UID să rămână în această entitate, deoarece, în cazul unei modificări în cadrul utilităților, vor trebui făcute modificări în multiple locuri în care nu este nevoie. Soluția este crearea unei noi entități cu respectivul UID și atributele corespunzătoare. La fel procedăm și în cazul furnizorilor pentru id\_produs, datele furnizorilor fiind irelevante pt acest UID. Creăm astfel entitatea PRODUS.

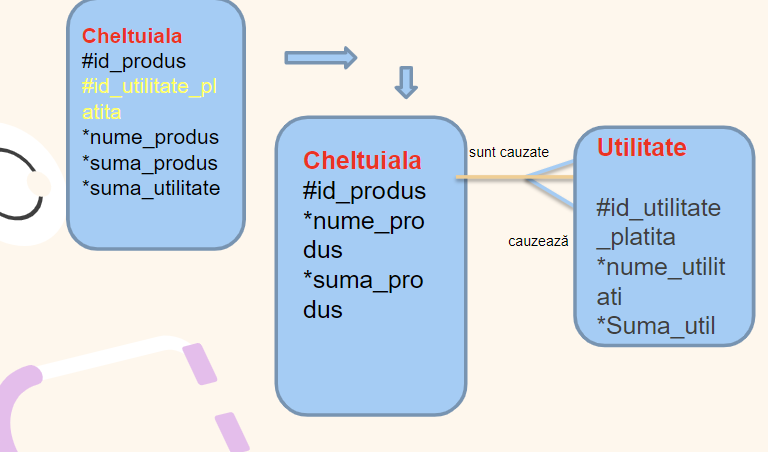


Fig II.2.1. A doua formă de normalizare

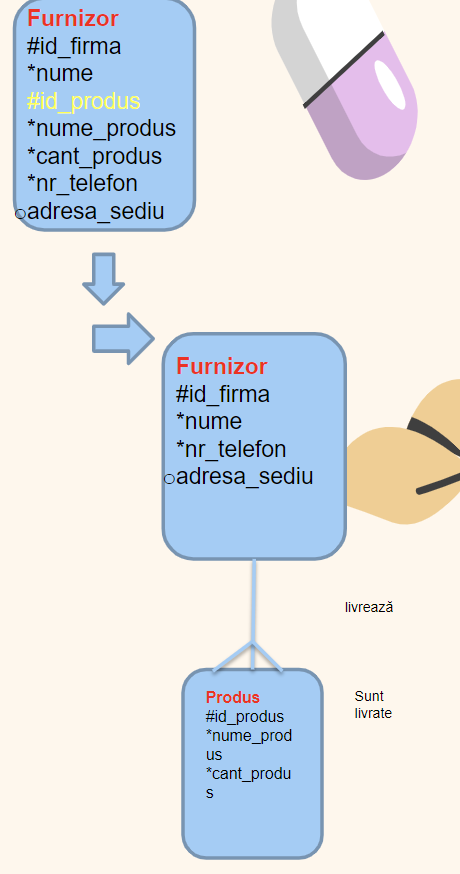


Fig II.2.2. A doua formă de normalizare.Model 2.

A treia formă de normalizare presupune neexistarea atributelor ce nu sunt UID-uri care să depindă de alte atribute ce nu sunt UID-uri. În cazul nostru vorbim de \*data\_obt\_specialitate și de domeniu\_specializare, soluționarea fiind crearea unei alte entități cu numele SPECIALIZARE.

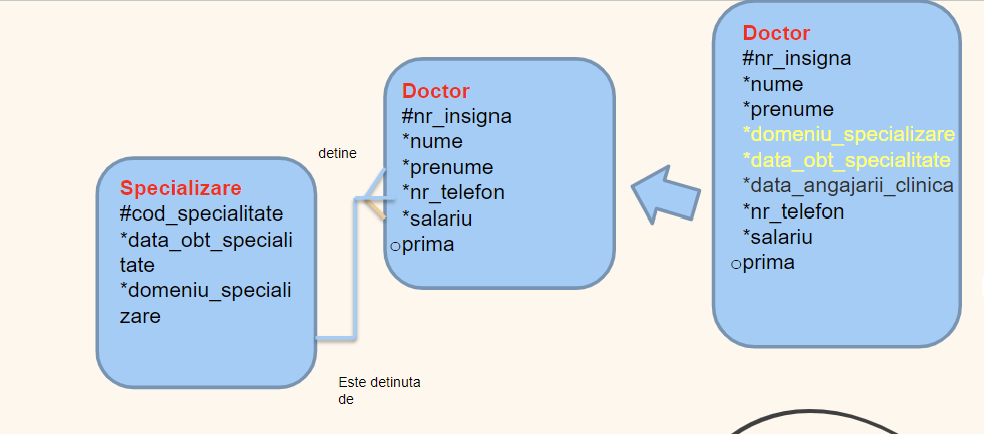


Fig II.3. A treia formă de normalizare

### III.1.5. Relații prezente în ERD

|  |  |
| --- | --- |
| Many to Many | Fiecare pacient este consultat de unul sau mai mulți doctori. Fiecare doctor consultă unul sau mai mulți pacienți. |

|  |  |
| --- | --- |
| One to many | Fiecare cabinet are una sau mai multe cheltuieli. Cheltuielile aparțin doar unui cabinet. |
| Ierarhie | Fiecare infirmier este coordonat de un asistent. Fiecare asistent este coordonat de un doctor. Se transmit UID-ul doctorilor către asistenți și cele ale asistentilor către infirmieri prin liniile barate. |

|  |  |
| --- | --- |
| Recursive | Există asistenți șefi care conduc și se ocupă de toti asistenții de pe specializarea respectivă. |

### III.1.6. Modelul conceptual inițial și final

## III. 2 Modelarea fizică

### III.2.1. Maparea entităților. Maparea relațiilor

Entitatea SPECIALIZARE devine tabela SPECIALIZARI

|  |  |  |  |
| --- | --- | --- | --- |
| Numele coloanei | Tip | Tip cheie | Opționalitatea |
| cse | Number | Pk | \* |
| doe | Varchar2 |  | \* |
| dse | Varchar2 |  | \* |

Entitatea DOCTOR devine tabela DOCTORI

|  |  |  |  |
| --- | --- | --- | --- |
| Numele coloanei | Tip | Tip cheie | Opționalitatea |
| nia | Number | Pk | \* |
| nume | Varchar2 |  | \* |
| prenume | Varchar2 |  | \* |
| ntn | Number |  | \* |
| salariu | Number |  | \* |
| prima | Number |  |  |

Entitatea CABINET devine tabela CABINETE

|  |  |  |  |
| --- | --- | --- | --- |
| Numele coloanei | Tip | Tip cheie | Opționalitatea |
| ict | Number | Pk | \* |
| nct | Varchar2 |  | \* |
| adresa | Varchar2 |  | \* |

Enitatea FURNIZOR devine tabela FURNIZORI

|  |  |  |  |
| --- | --- | --- | --- |
| Numele coloanei | Tip | Tip cheie | Opționalitatea |
| ifa | Number | Pk | \* |
| nume | Varchar2 |  | \* |
| ntn | Number |  | \* |
| asu | Varchar2 |  |  |

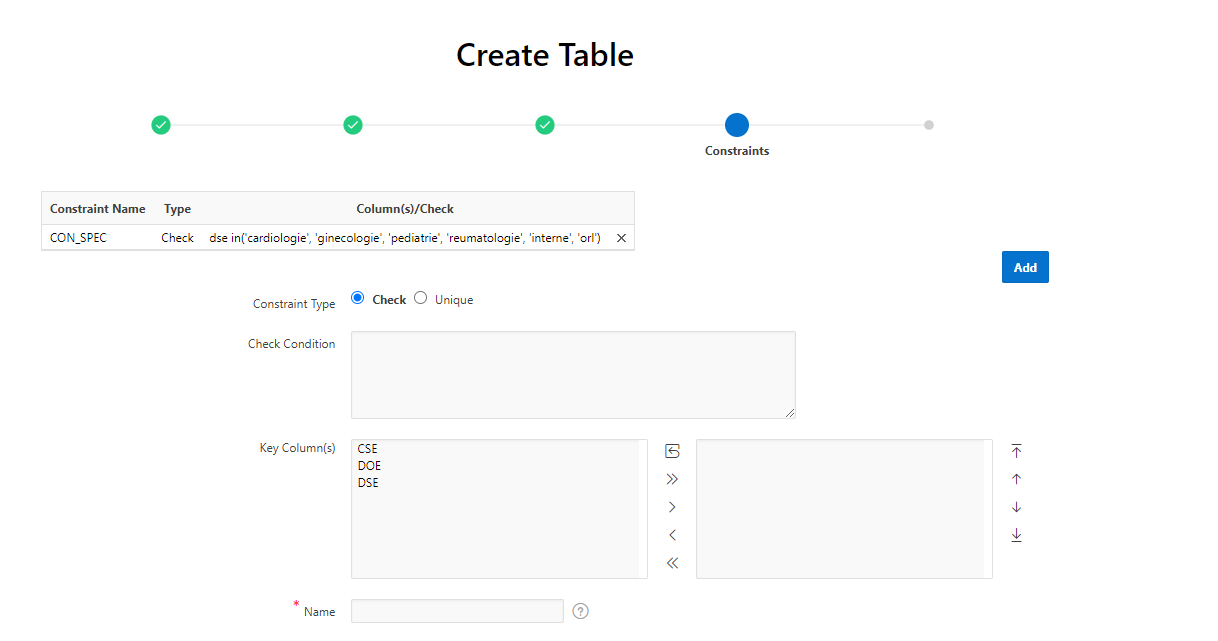
Entitatea Contract devine tabela Contracte

|  |  |  |  |
| --- | --- | --- | --- |
| Numele coloanei | Tip | Tip cheie | Opționalitatea |
| icrt | Number | Pk | \* |
| data | Varchar2 |  | \* |
| ifa | Number | Fk | \* |
| ict | Number | Fk | \* |

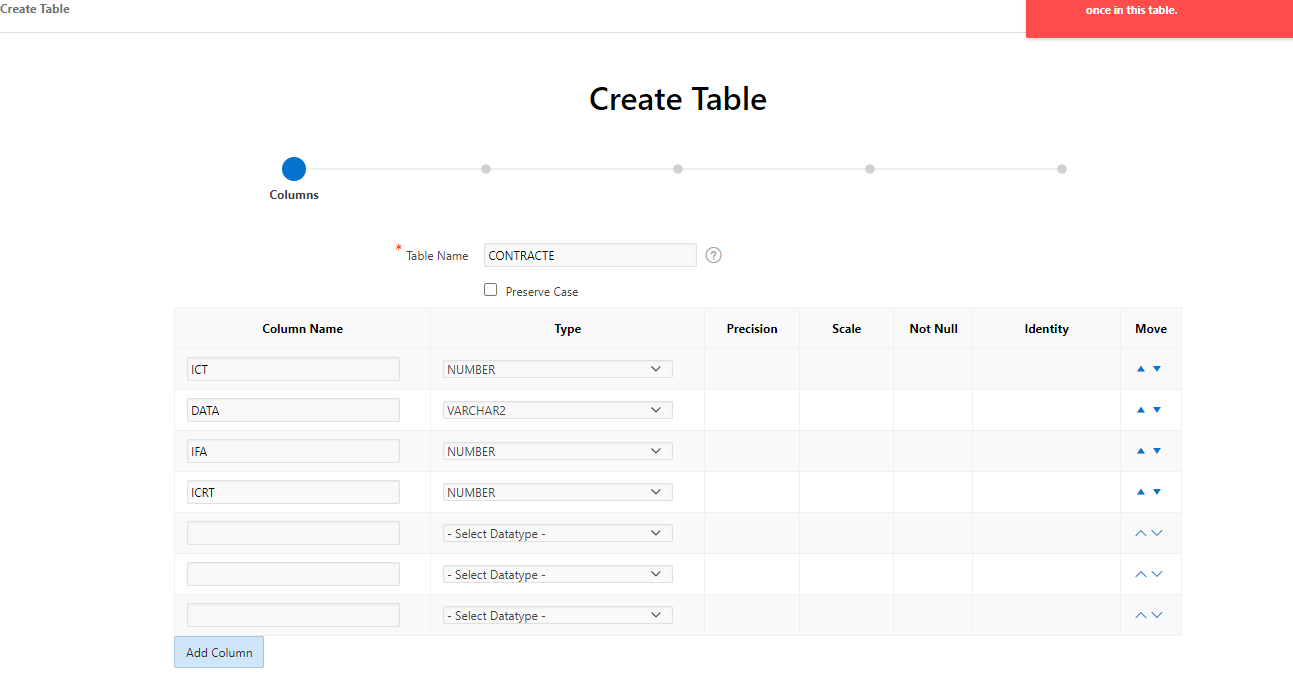
### III.2.2.Implementarea tabelelor în Oracle Application Express

* Procesul de implementare

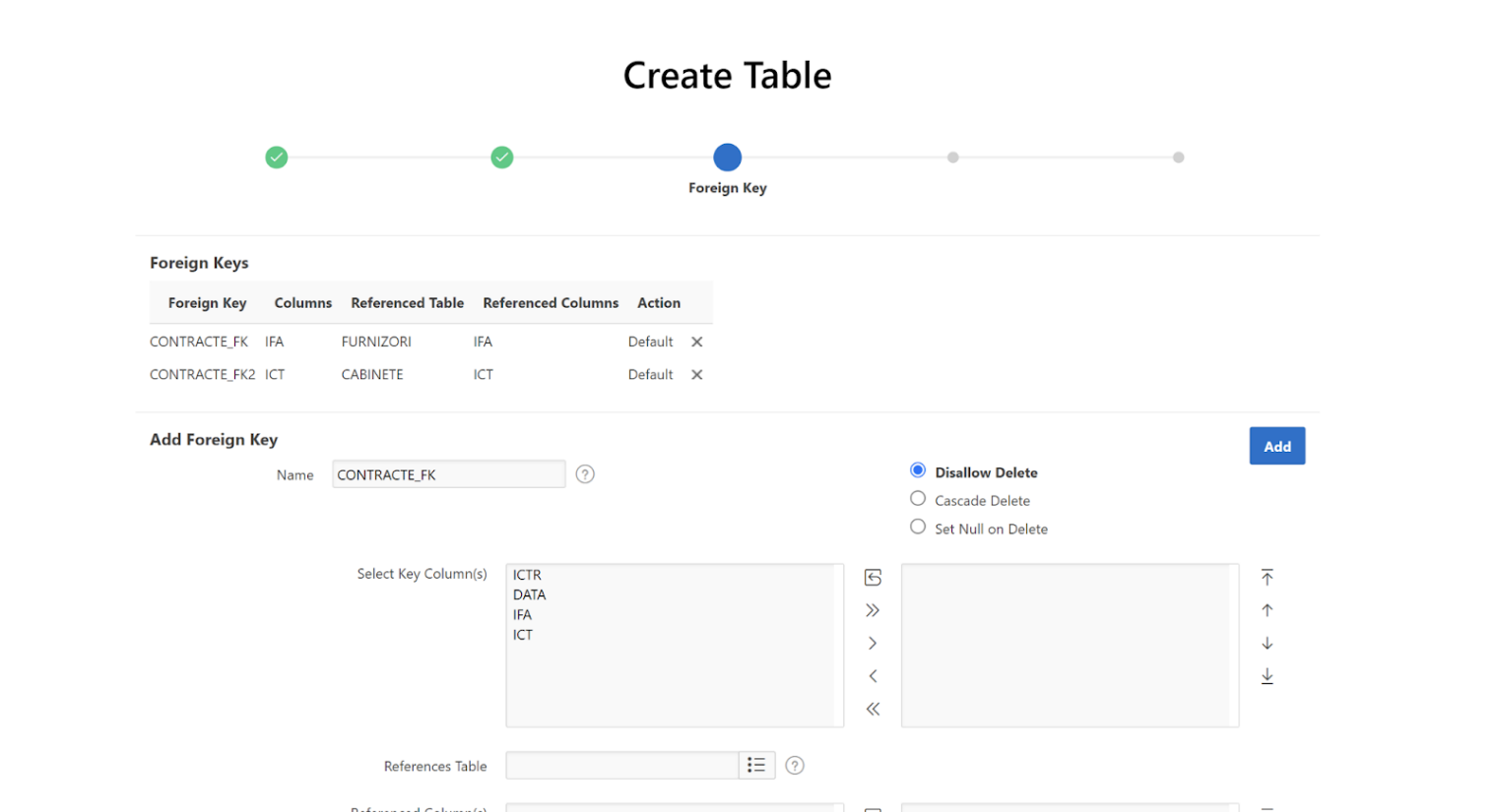
Pentru tabela SPECIALIZARI se impune o constrângere pentru câmpul dse, adică domeniul specializări, care poate fi ales dintr-o listă de specializări deja conceput la introducerea datelor în tabel.



Imaginea următoare prezintă introducerea tipurilor de câmpuri din procesul de creare a tabelelor.

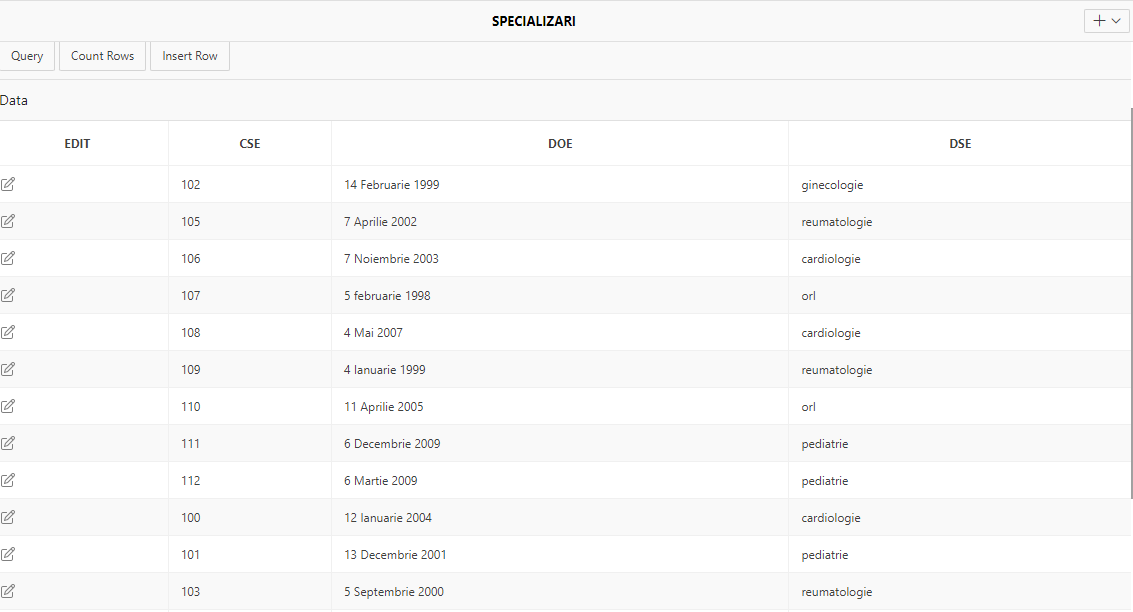


Pentru entitatea CONTRACT a fost nevoie de crearea a două chei străine, deoarece este entitate de intersecție pentru rezolvarea relației many-to-many între entitățile FURNIZOR și CABINET.

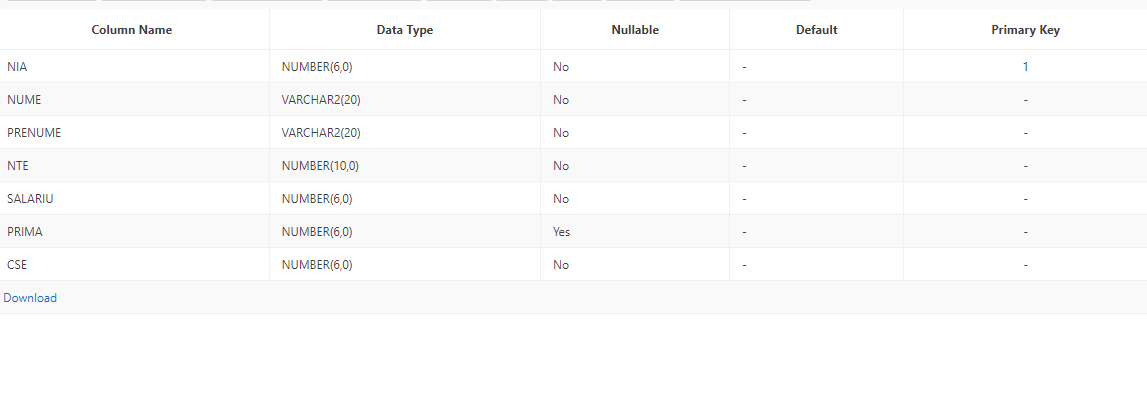


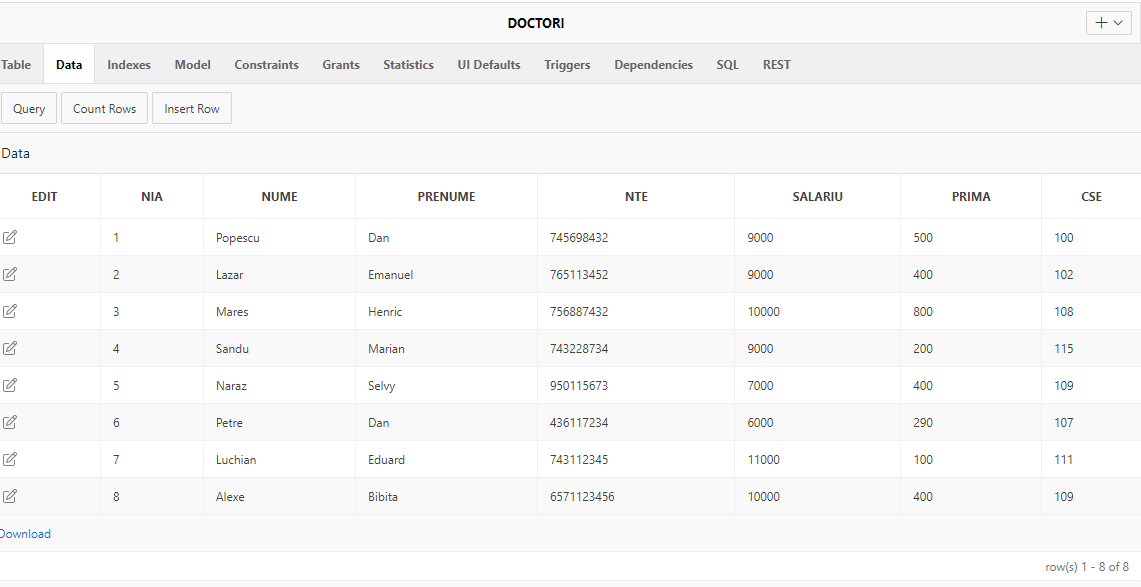
* Tabele create

Entitatea SPECIALIZARE Tabela SPECIALIZARI

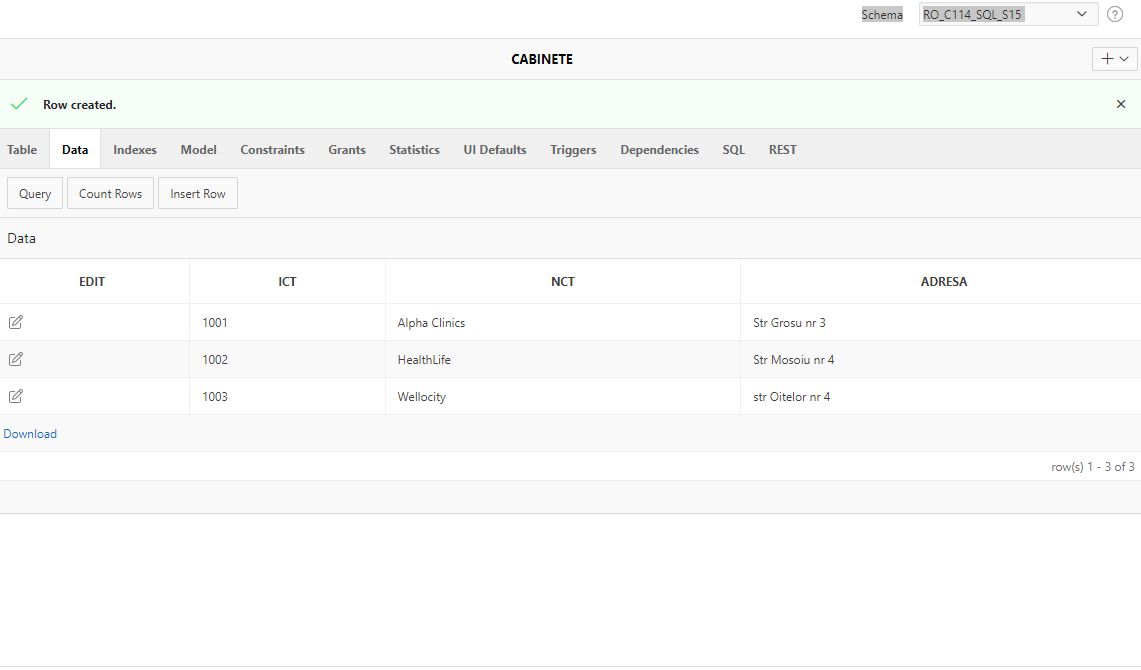


Entitatea DOCTOR Tabela DOCTORI

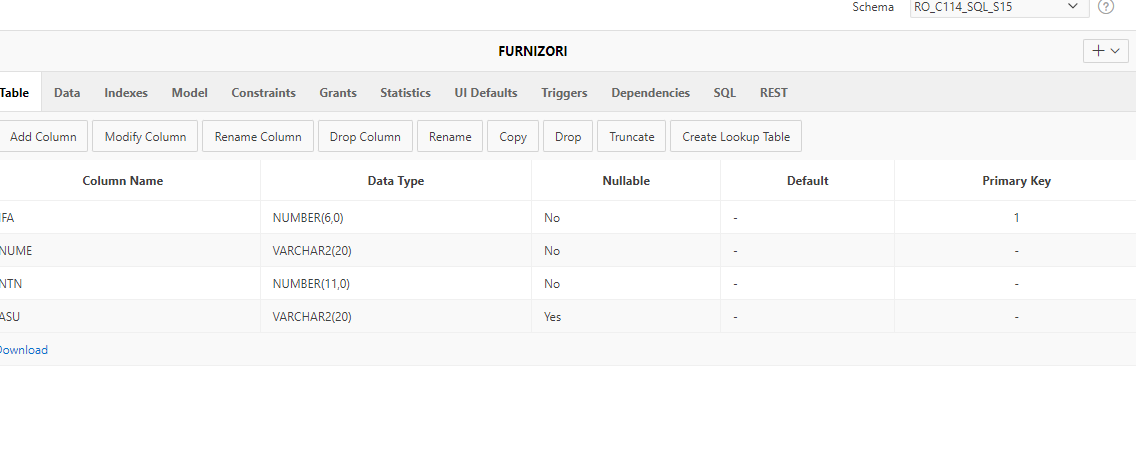


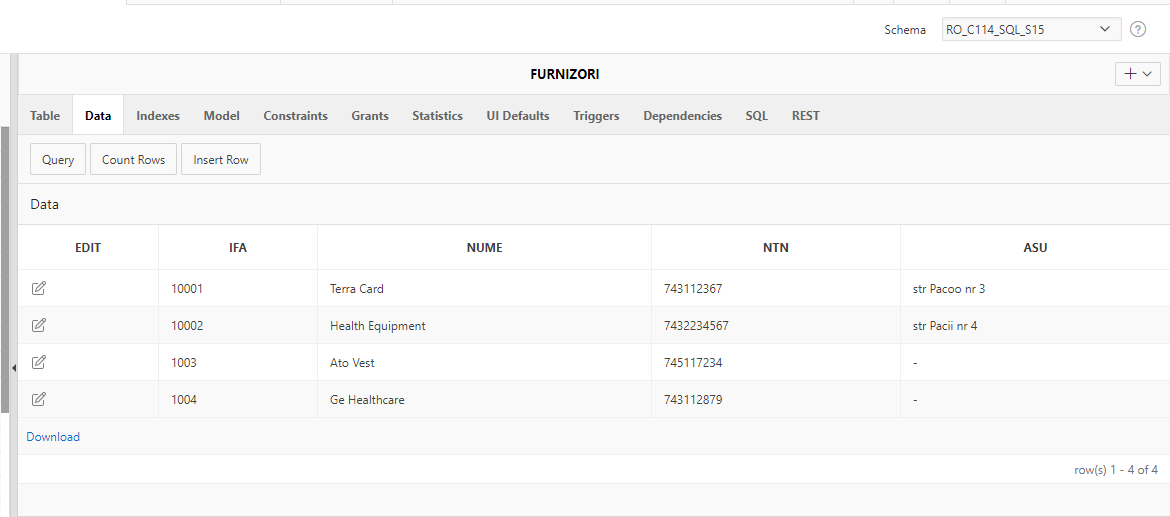


Entitatea CABINET Tabela CABINETE

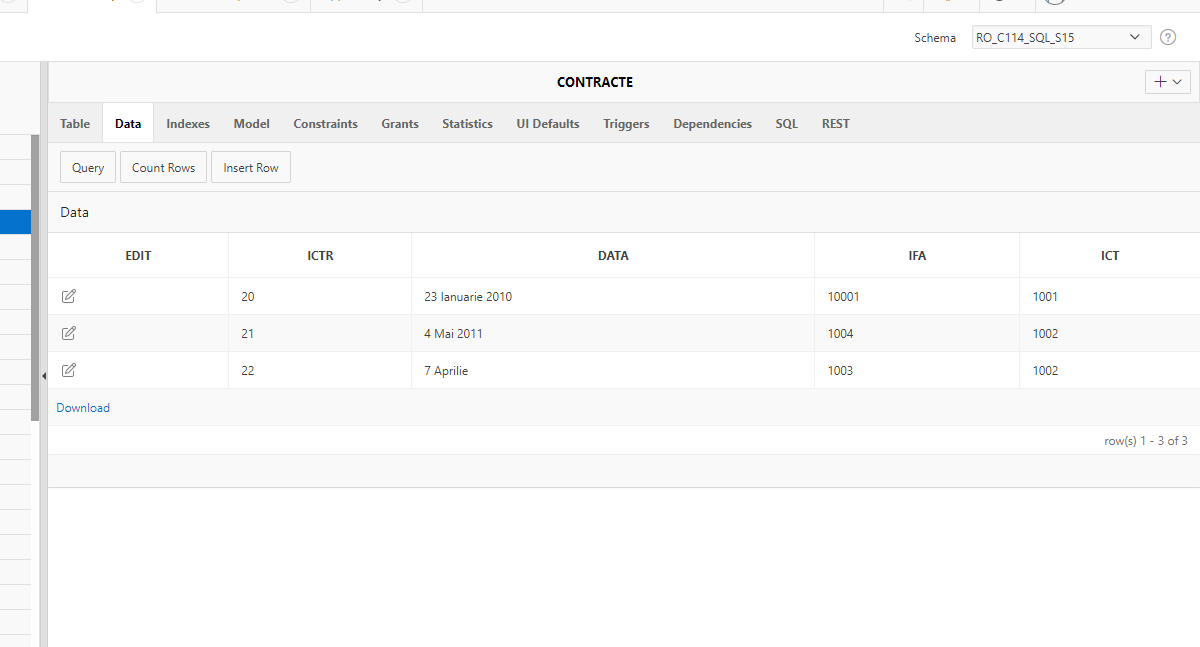


Entitatea FURNIZOR Tabela FURNIZORI





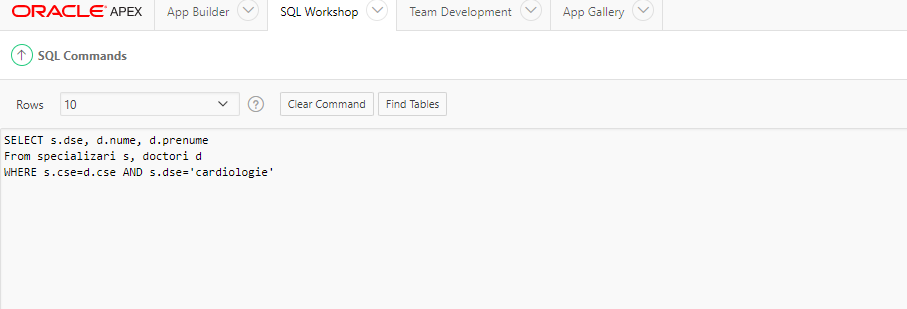
Entitatea CONTRACT Tabela CONTRACTE

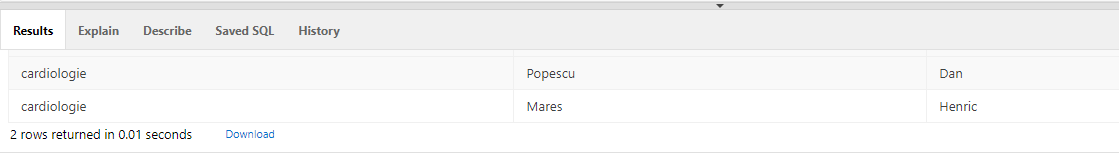


# 

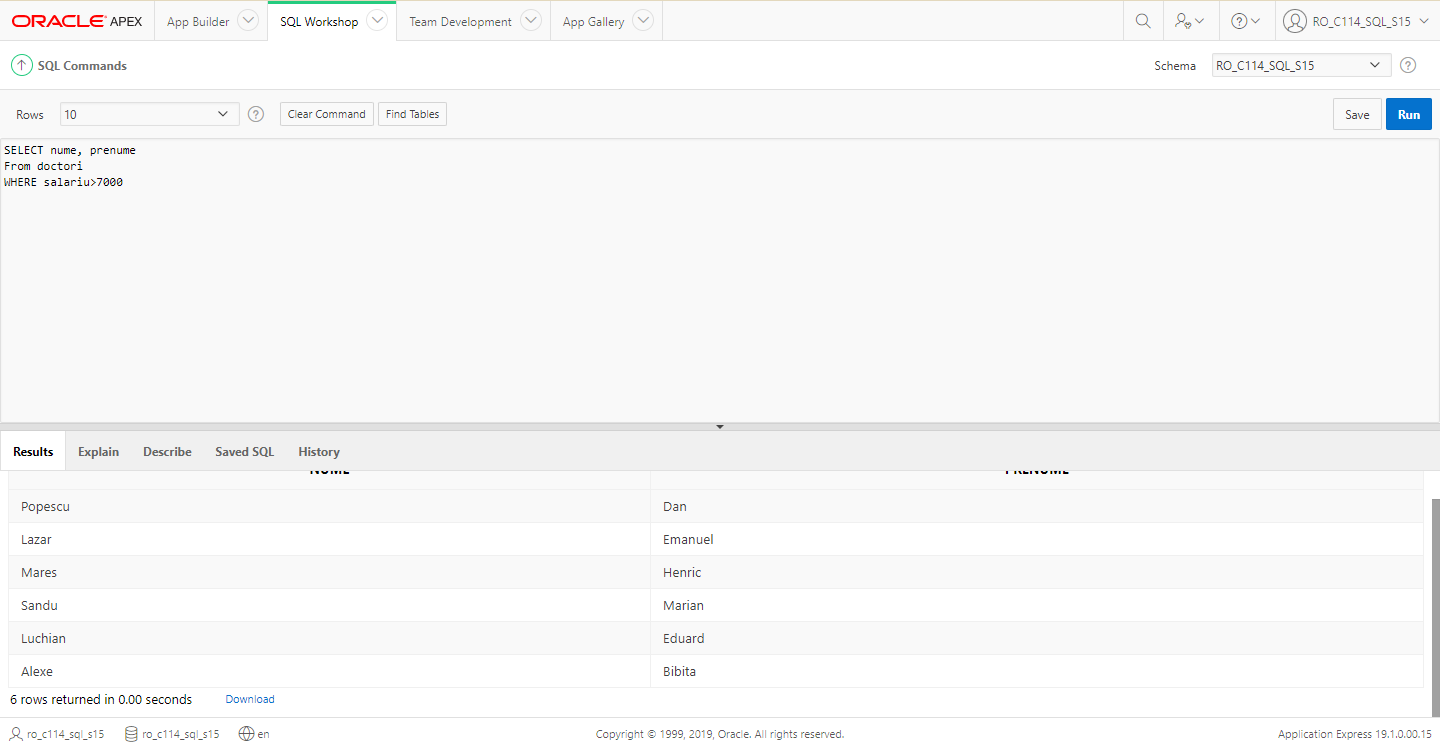
# III.3 Interogări

1. Aici s-a realizat un EquiJoin, deoarece se încearcă afișarea numelor și prenumelor doctorilor care au specializarea “cardiologie”.

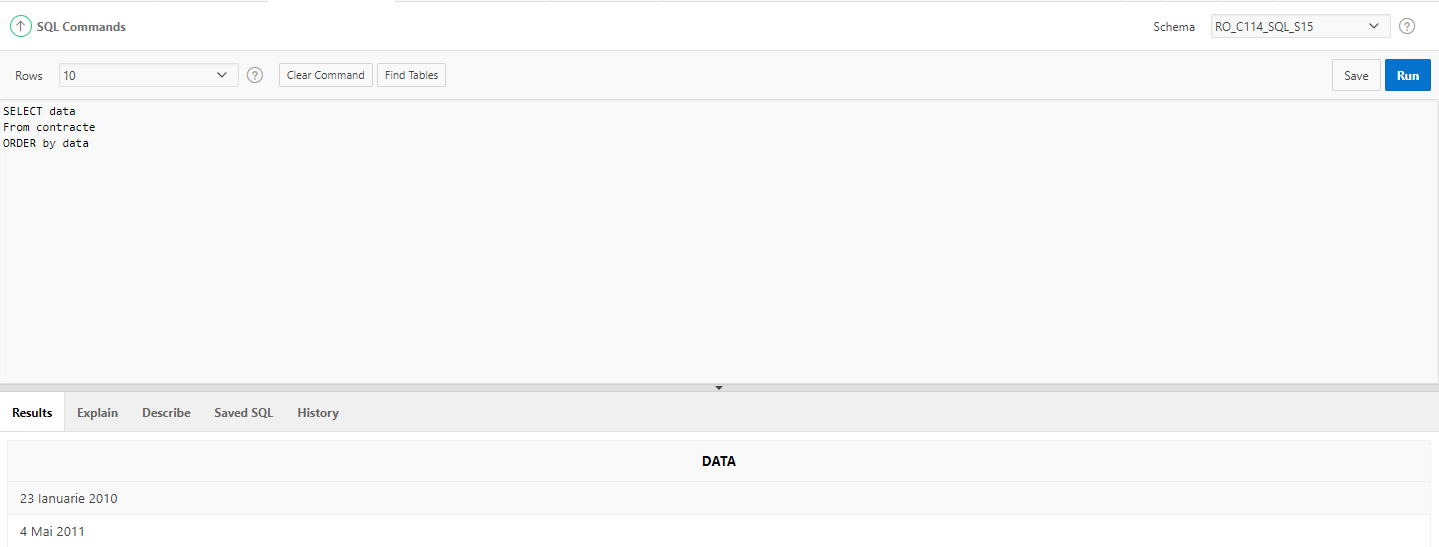




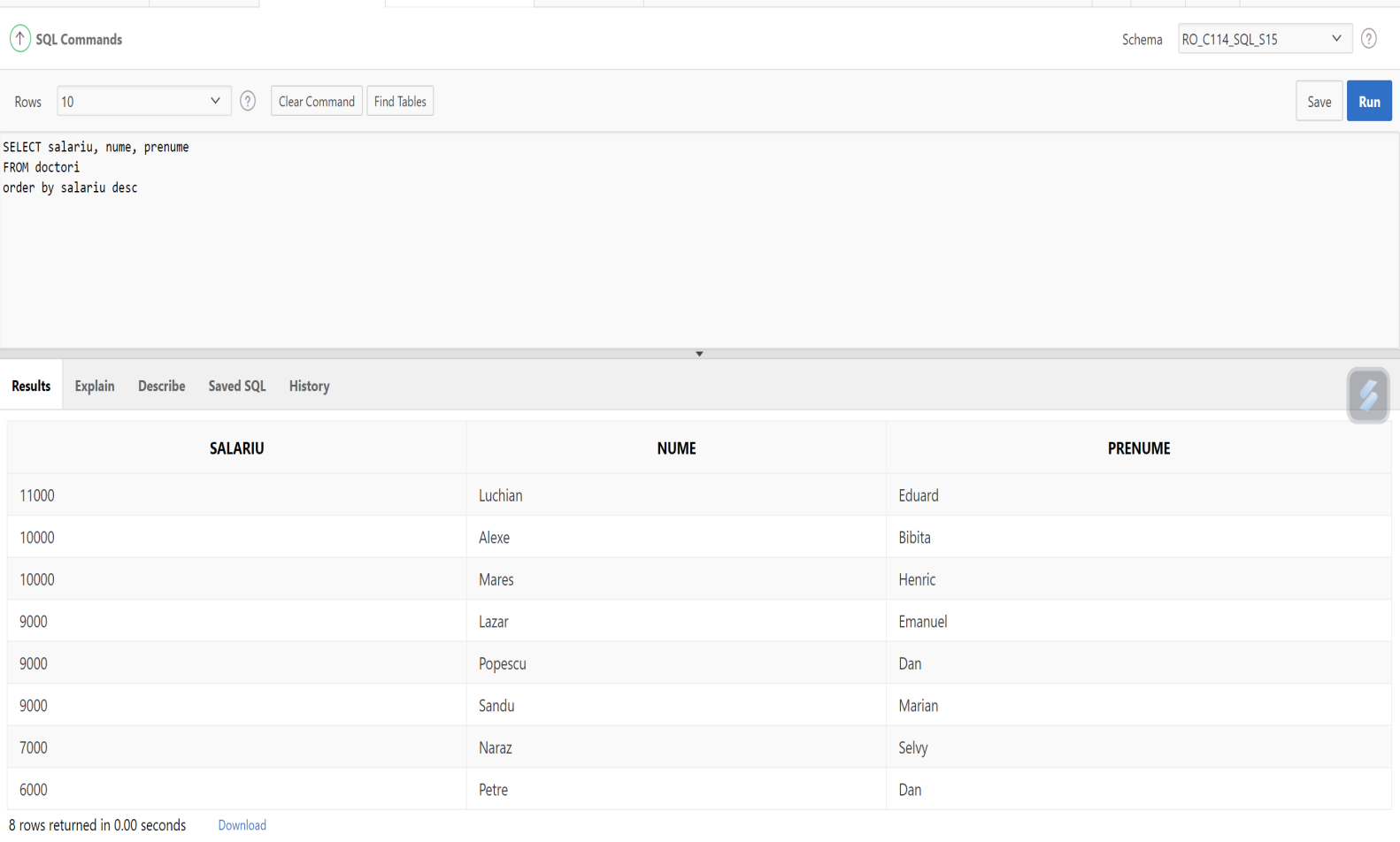
2. Aici s-a realizat o interogare prin care se caută doctorii cu salariul mai mare de 7000 de lei și se afișează numele și prenumele acestora



3. Următoarea interogare ordonează contractele dintre furnizori și cabinete după data emiterii contractului



4. Următoarea interogare ordonează doctorii după salariile lor în mod descrescător



# IV.Concluzii

Bazele de date sunt esențiale pentru stocarea, gestionarea și accesarea informațiilor în mod eficient. O structură bine proiectată a bazei de date poate elimina redundanța și poate îmbunătăți performanța și securitatea. Normalizarea bazei de date este un proces important pentru a asigura integritatea și coerența datelor și pentru a minimiza redundanța. În plus, o baza de date trebuie sa fie proiectată pentru a putea face față creșterii volumului de date și a cerințelor de accesare. Cu toate acestea, proiectarea și întreținerea unei baze de date pot fi complexe și necesită expertiză în domeniu pentru a fi realizate în mod corespunzător. Realizarea acestui proiect a fost merituoasă pentru cantitatea de informații pe care am obținut-o și abilitățile dobândite.

# V.Bibliografie

<https://www.oracle.com/ro/database/what-is-database/>

<https://support.microsoft.com/ro-ro/office/no%C8%9Biuni-elementare-despre-bazele-de-date-a849ac16-07c7-4a31-9948-3c8c94a7c204#:~:text=O%20baz%C4%83%20de%20date%20este,sau%20o%20foaie%20de%20calcul>.

-Manual informatică Oracle clasa

 a XII-a-