



UNIVERSITATEA TEHNICĂ „GHEORGHE ASACHI” IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

DISCIPLINA BAZE DE DATE

Monitorizarea sistemului medical într-un spital de chirurgie

Nume: Ștefan Alexandra

Grupa: 1306B

Coordonator: Sorin Avram

IAȘI, 2020

Titlu temă: Monitorizarea sistemului medical într-un spital de chirurgie

Tema are în vedere analiza, proiectarea și implementarea unei baze de date care să modeleze într-o manieră cât mai realistă activitatea medicală dintr-un spital de chirurgie, mai exact ceea ce este legat de medici, pacienți și alte cadre medicale, împreună cu acțiunile principale pe care le desfășoară.

Descriere a cerințelor și a modului de organizare a temei

Gestionarea activității dintr-un spital necesită abilitatea de a munci intens și de a fi atent la detalii atunci când vine vorba de ținerea în evidență a numeroase documente medicale și a nevoilor cadrelor medicale, pacienților și celor „din spate”, care se ocupă cu aprovizionarea și managementul bugetului spitalului respectiv (tema nu vizează însă ultima mențiune decât într-o mică măsură) și a relațiilor dintre toți aceștia.

Un spital dedicat chirurgiei prezintă diverse *departamente* (de chirurgie endocrină, chirurgie oncologică, chirurgie cardiotoracică, chirurgie plastică, neurochirurgie etc.) pentru care se precizează pe rând, pentru identificare, un id, un nume și un șef de secție, care primește toate rapoartele referitoare la starea pacienților din secția de care aparține.

În fiecare *departament* există mai mulți *specialiști* (fiecare cu un id, un nume, un email, un număr de telefon și un salariu), care se ocupă cu consultarea și tratarea în parte a fiecărui *pacient*.

Totodată, în mod evident, la spital se prezintă numeroși *pacienți* cu afecțiuni diverse (fiecare cu un cod de asigurat (de pe cardul de sanatate), un cnp, un nume, o adresă de domiciliu și un număr de telefon pentru contact – poate fi al său sau al celui care are grijă de el, în cazul minorilor spre exemplu).

Pentru *pacienți* se pot planifica *consultații* realizate de *specialiști* (fiecare cu un id, o dată dedicată din an și un număr de cabinet pentru orientarea fizică în spital) și *intervenții* chirurgicale realizate de *specialiști* (fiecare cu un id, un nume, o dată de început, o dată de final – ce poate fi adăugată oricând, un cost total și o sală de operație rezervată).

Intervențiile prevăd *internări* (fiecare cu un id și un număr de cameră, o dată de internare și o dată de externare – ce poate fi adăugată oricând), supravegheate de *asistenți medicali* (fiecare cu un id, un nume, un număr de telefon, un statut medical neobligatoriu de menționat și un salariu) ce administrează *tratamente* post-operatorii pacienților (fiecare cu un cod, un nume, o marcă, o descriere scurtă, o perioadă de administrare, o primă oră de administrare zilnică și un dozaj zilnic). Datele calendaristice în care specialiștii sunt ocupați se vor regăsi de asemenea și într-un *orar* dedicat (fiecare intrare având un id unic propriu).

Cu siguranță există și alte detalii utile ce ar trebui să se regăsească într-o bază de date medicală, însă programul este menit să modeleze pe scurt ceea ce este mai important.

Descriere a funcționalității aplicației

Acțiunile principale realizate în cadrul unui spital sunt:

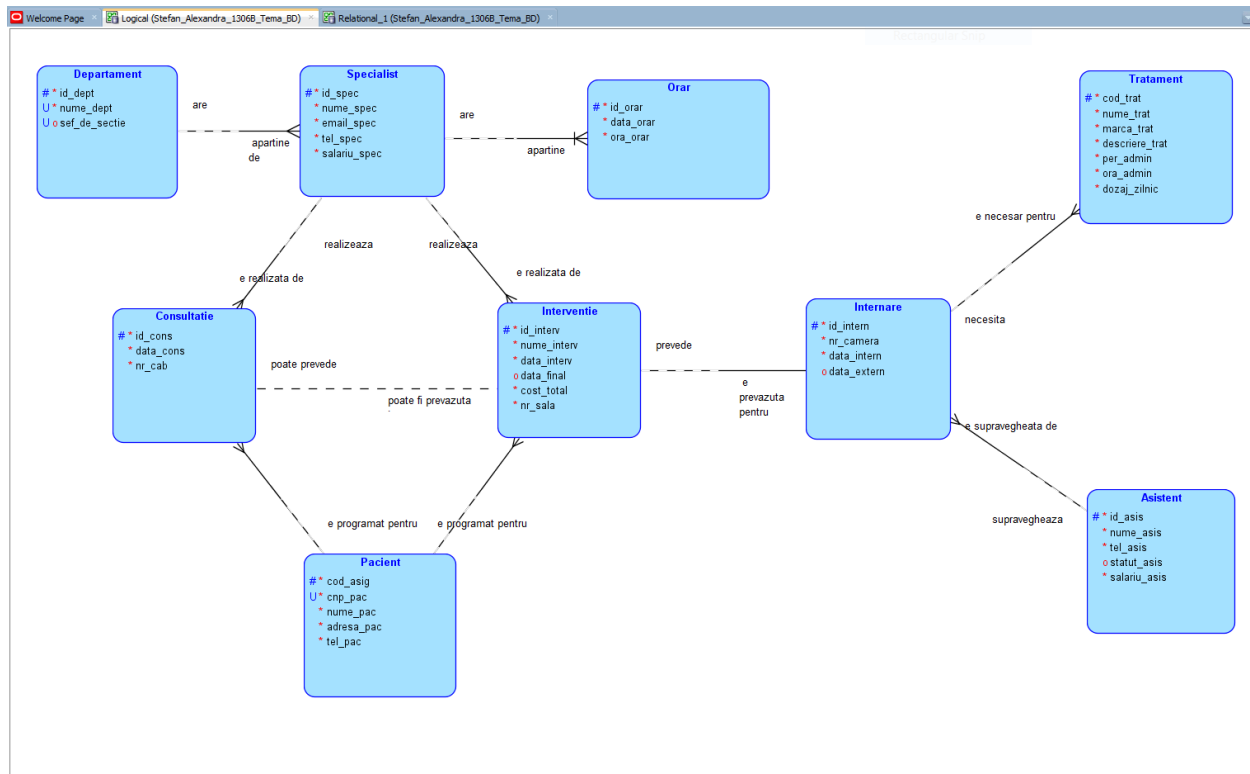
- evidența cadrelor medicale angajate;
- evidența pacienților care se prezintă la unitatea medicală;
- evidența activităților desfășurate (consultații, intervenții chirurgicale, internări, externări);
- evidența tratamentelor administrate.

Structura și interrelaționarea tabelelor

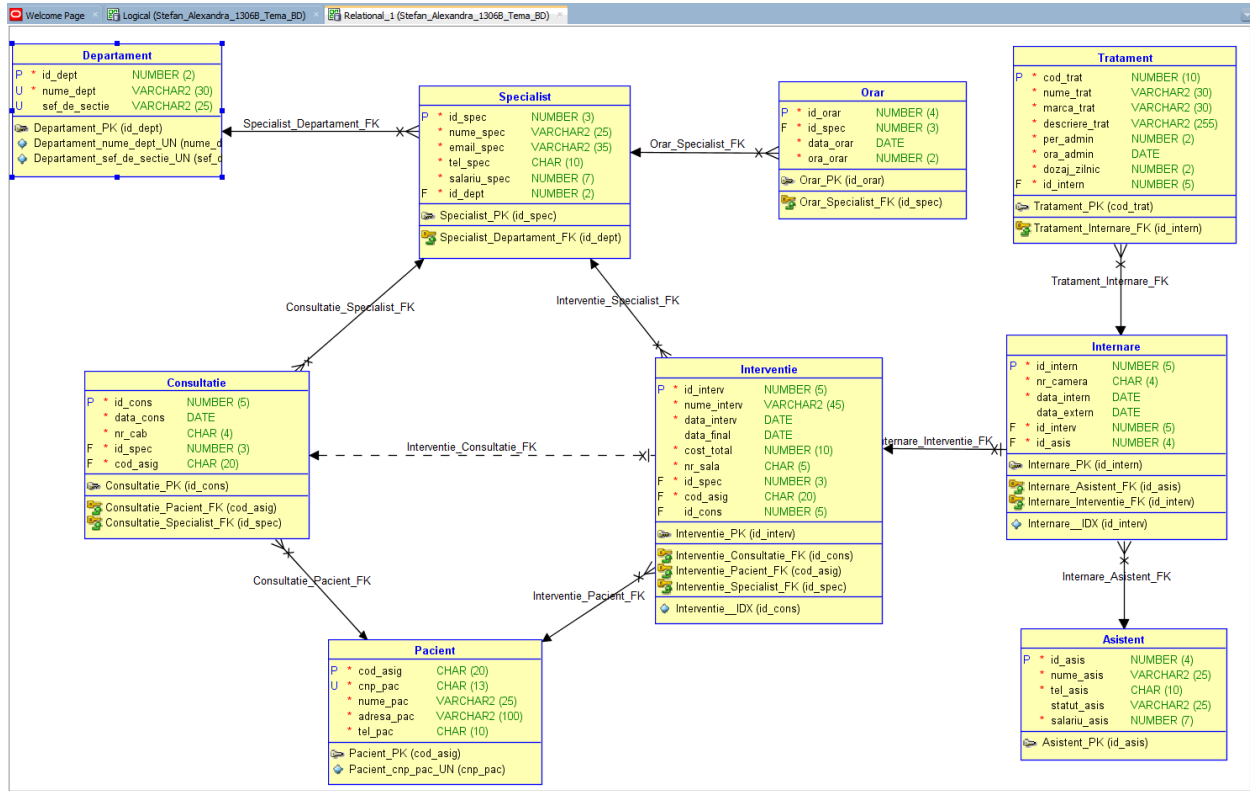
Tabelele ce constituie această bază de date sunt:

- Departament
- Specialist
- Pacient
- Consultatie
- Interventie
- Internare
- Tratament
- Asistent
- Orar

Model logic



Model relațional



Relații între tabele

În baza de date creată se regăsesc relații (ce reprezintă de asemenea și constrângeri de integritate referențială *foreign key*) atât de tip one-to-one cât și de tip one-to-many, menite să confere logică în ceea ce privește datele introduse în tabele și ordinea lor de introducere.

Între *Departament* și *Specialist* este stabilită o relație one-to-many. Într-o anumită secție din spital (chirurgie generală, chirurgie cardiotoracică, neurochirurgie etc.) pot lucra mai mulți medici însă reciprocă nu este posibilă, un specialist nu poate aparține de mai multe secții având în vedere că el deține cunoștințe într-un anumit domeniu în principal.

Între *Specialist* și *Orar* se realizează o legătură de tip one-to-many, astfel încât pentru un anumit medic specialist să se poată memora numeroasele programări cu diverși pacienți și într-un loc special dedicat.

Relațiile *Specialist-Consultatie* și *Specialist-Interventie* sunt de tip one-to-many. Din nou, pentru un medic specialist se pot programa mai multe consultații cu pacienții și intervenții chirurgicale însă inversa ar putea genera diverse probleme organizatorice sau chiar legate de viața pacientului. De exemplu, se presupune că un pacient prezintă două probleme diferite de sănătate. Cum ar putea acesta să fie consultat simultan, la aceeași oră, de medici diferiți, în același cabinet,

cu același instrumentar medical (valabil în cabinetul respectiv, presupunând că este destinat unei singure specializări)? La fel și pentru cazul unei intervenții.

Totodată, relațiile *Consultatie-Pacient* și *Interventie-Pacient* sunt de tip many-to-one. Atâta timp cât este asigurat medical sau își permite din punct de vedere financiar, un pacient are dreptul să programeze una sau mai multe consultații cu un medic specialist, în funcție de nevoi și de timpul disponibil, regulă valabilă și în cazul în care specialistul ajunge la concluzia că pacientul necesită mai multe intervenții medicale.

Între *Interventie* și *Internare* există o relație one-to-one. În mod logic, după o intervenție chirurgicală un pacient are nevoie de o perioadă de refacere a organismului (necesar fiind un singur proces de internare, nu mai multe). Astfel, o internare se poate face doar cu un singur scop, după o anumită intervenție. Abia după un anumit timp de refacere, oricât de mic, pacientul poate suferi o altă intervenție, după care procesul de internare va continua, adăugându-se altă intrare de date în tabelă.

Relația dintre *Consultatie* și *Interventie* este una de tip one-to-one opțională la ambele capete. Prin intermediul ei, un pacient are trei opțiuni în spital: să programeze cu medicul specialist o consultație simplă, o intervenție chirurgicală urmată de internare sau, în cazul cel mai complex, o consultație urmată de o intervenție și internare.

De asemenea, baza de date mai conține două relații importante, de tip one-to-many: *Asistent-Internare* și *Internare-Tratament*. Un asistent medical are datoria de a supraveghea mai mulți pacienți internați, nefiind necesar să se ocupe doar de unul singur pe perioada unei zile. Putem deduce deci că un pacient nu are nevoie să se afle în grija mai multor asistenți pentru a nu irosi timpul acestora. Tratamentul administrat celor internați poate fi de mai multe tipuri, în dozaj diferit. Se presupune că fiecare medicament prescris, împreună cu detaliile de administrare, este înregistrat în mod unic în baza de date, pentru o ținere în evidență mai amănunțită.

Constrângeri

În program sunt utilizate atât constrângeri de tip check, unique și not null cât și trigger, ce reprezintă reguli utile pentru verificarea și validarea datelor.

Toate variabilele din tabele sunt marcate ca fiind de tip „Mandatory” – obligatoriu, fapt ce impune automat condiția ca ele să nu poată fi nule, adică omise atunci când este introdusă o intrare în tabelă (mai puțin variabilele *sef_de_sectie* și *statut_asis* – de importanță minoră pentru activitatea medicală, *data_final* și *data_extern* – pot fi introduse și mai târziu în tabelă, atunci când acțiunea medicală este terminată).

Variabilele marcate ca „Primary UID” (precedate de „#”, care reprezintă totodată constrângeri de integritate referențială *primary key*) și variabila *cnp_pac* din tabela Pacient (precedată de „U”) respectă regula unicității întrucât, în mod evident, două intrări ale oricărei tabele nu pot fi identice. Spre exemplu, presupunând că există doi asistenți cu același nume, același număr de

telefon, același statut medical și același salariu, ei se vor deosebi totuși prin id-ul specific primit la angajare.

Constrângeri de tip check se regăsesc în aproape toate tabelele create, conferind acestora un plus de acuratețe și realism. Are loc verificarea dimensiunii valorii introduse într-o coloană pentru variabilele *nume_dept*, *sef_de_sectie*, *nume_spec*, *email_spec*, *nume_pac*, *adresa_pac*, *nume_interv*, *nume_trat*, *marca_trat*, *descriere_trat* și *nume_asis* astfel încât, în cazul unei erori de tastare, lungimea acestora să nu poată fi doar de un caracter. Verificarea formatului unui nume (astfel încât să fie format doar din litere) pentru *nume_dept*, *sef_de_sectie*, *nume_spec*, *nume_pac*, *nume_interv* și *nume_asis*, unui email (astfel încât să conțină obligatoriu „@” și „.”) pentru *email_spec*, a unui număr de telefon și cod numeric personal (astfel încât să fie formate doar din cifre) pentru *tel_spec*, *tel_pac*, *tel_asis* și *cnp_pac*, a numelui unui cabinet (să înceapă cu „CC”) pentru *nr_cab*, a numelui unei săli de operație (să înceapă cu „SO”) pentru *nr_sala* și a numelui unei camere de internare (să înceapă cu „I”) pentru *nr_camera* are loc cu ajutorul unor tipare regex. Atunci când este introdusă, variabila *statut_asis* are voie să ia doar o valoare din lista constrângerii impuse: „debutant”, „integrat”, „principal”. De asemenea, anumite variabile au voie să ia o valoare doar între anumite intervale: *per_admin* (perioada de administrare) poate fi de minim o zi și maxim 30 de zile, *dozaj_zilnic* poate fi de minim 1 doză și maxim 10, în funcție de medicament, iar *ora_orar* trebuie să fie cuprinsă între 0 și 24. Pentru *salariu_spec*, *cost_total* și *salariu_asis* este stabilită o constrângere astfel încât acestea să nu poată fi negative sau egale cu 0.

Pe lângă toate acestea, în program sunt folosite și triggere pentru datele calendaristice introduse în tabele. Astfel, *data_cons* (data unei consultații), *data_interv* (data de început a unei intervenții) și *data_orar* trebuie să fie obligatoriu mai mari decât SYSDATE (data actuală preluată din sistem). De asemenea, *data_intern* (data internării aferente unei intervenții chirurgicale) trebuie să fie mai mare decât *data_interv* (în cuprinsul trigger-ului este declarată o variabilă auxiliară în care este pusă valoarea datei de început a intervenției, urmând să aibă loc o comparație între *data_intern* și variabila auxiliară dacă *id_interv* este identic în tabelele de comparat). În mod logic, *data_final* trebuie să fie mai mare decât *data_interv* și *data_extern* mai mare decât *data_intern*. Pentru crearea trigger-ului referitor la *data_intern* a fost nevoie atât de utilizarea cunoștințelor actuale de Baze de date cât și folosirea unei sugestii de implementare de pe site-ul stackoverflow.com.

Totodată, tabela *Orar* ajută atât la organizarea activității medicale cât și la îndeplinirea constrângerii conform căreia datele nu au voie să se suprapună (un medic nu poate consulta și face o intervenție chirurgicală în același timp). Deoarece cunoștințele legate de folosirea triggerelor depășesc nivelul disciplinei Baze de date, s-a încercat utilizarea unei metode de constrângere a datelor calendaristice atunci când sunt introduse/modificate în tabele (la INSERT, UPDATE și DELETE) prin structuri de tipul BEGIN-EXCEPTION-END și pe principiul că o consultație sau o intervenție începe la o oră fixă (și are alocat un timp exprimat în ore).

Tehnologii utilizate

În cadrul temei, pentru partea de back-end a aplicației, modelul logic și modelul relațional cu ajutorul cărora se pot implementa tabelele necesare și relațiile (foreign-key-uri) dintre acestea sunt create prin intermediul programului SQL Datamodeler. Informațiile introduse mai apoi sunt stocate într-o bază locală de date Oracle, accesibilă prin programul Oracle Database 11g Express Edition sau SQL Developer.

Partea de front-end a aplicației, mai exact ceea ce vede utilizatorul care introduce informațiile în baza de date medicală, este implementată cu ajutorul programului Visual Studio 2019. Limbajul utilizat în cadrul aplicației de tip Windows Forms este C#. Pentru realizarea conexiunii dintre interfața creată și baza de date este necesară instalarea pachetului *System.Data.OracleClient* din *Project – Manage NuGet Packages*.

Mod de conectare la baza de date

Conexiunea cu o bază de date Oracle se stabilește din proiectul specific, din *View – Server Explorer*. În fereastra nou apărută se apasă click dreapta pe *Data Connections* și se selectează *Add Connection*. La *Data Source* se selectează prin *Change* opțiunea *Oracle Database (OracleClient)*, pentru *Server name* se precizează numele serviciului prin care funcționează baza de date – *XE* în cazul prezentat și în final se introduc datele personale de conectare (username-ul și parola). Se testează conexiunea și, dacă are succes, se merge mai departe.

Pasul următor este ca în fiecare fișier sursă, acolo unde este nevoie de o legătură cu baza de date, să se inițializeze o nouă conexiune (*OracleConnection oracleConnection = new OracleConnection();*), respectiv în fiecare funcție apelantă, prin comenzi încadrate în structuri try-catch-finally (*try { oracleConnection.ConnectionString = c; oracleConnection.Open(); } catch (Exception ex) { MessageBox.Show("Failed! Check again!"); } finally { oracleConnection.Close(); }* unde *string c = "DATA SOURCE = XE;" + "PERSIST SECURITY INFO = True; USER ID = username; password = parola; Pooling = False;"*).

Aspect interfață


Regional Surgery Clinic Database

Schedule

Data entry

Contact Us

Help



18:48:31

According to the World Health Organization (WHO), there were approximately 312.9 million operations carried out worldwide in 2012, increasing from 226.4 million in 2004.

In the United States, the most common surgical procedures include appendectomy, cesarean section, cataract surgery, mastectomy, and coronary artery bypass. Some procedures are very complex and unusual.

Feces from people or animals is an important source of germs like Salmonella, E. coli O157, and norovirus that cause diarrhea, and it can spread some respiratory infections like adenovirus and hand-foot-mouth disease.

Removing germs through handwashing therefore helps prevent diarrhea and respiratory infections and may even help prevent skin and eye infections.


About 1.8 million children under the age of 5 die each year from diarrheal diseases and pneumonia, the top two killers of young children around the world.

Preventing sickness reduces the amount of antibiotics people use and the likelihood that antibiotic resistance will develop. Handwashing

December 2020

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|-----|-----|-----|-----|-----|-----|-----|
| 29 | 30 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Today: 12/20/2020



Data

Specialists

Patients

Assistants

Medications

Home



Consultations

View

Insert

Update

Remove

Apply

| | SId | Specialist | PId | Pacient | CId | Date | CR |
|---|-----|-----------------|------------------|-----------------|-------|--------------------|------|
| ▶ | 102 | Emilian Butescu | 4885534365313... | Teodora Hritcu | 10005 | 7/24/2021 12:00... | CC09 |
| | 103 | Laur Gradescu | 4154448462956... | Grigore Maria | 10004 | 7/9/2021 4:00 PM | CC01 |
| | 104 | Giulia Rossini | 4916838048324... | Eugenia Sasescu | 10000 | 6/3/2021 10:00 ... | CC03 |
| | 101 | Darie Bitman | 4202346552307... | Ioan Anitei | 10003 | 10/19/2021 9:00... | CC12 |
| | 102 | Emilian Butescu | 4957796577300... | Maia Petcu | 10002 | 5/14/2021 3:00 ... | CC10 |
| | 100 | Elisa Mincu | 4206449617390... | Tudor Durea | 10001 | 7/11/2022 1:00 ... | CC09 |

<

Consultations

View

Insert

Update

Remove

Apply

Rectangular Snip

Id

Appointment

Specialist name

Specialist Id

Patient name

Patient Id

Room number

<

Consultations

View

Insert

Update

Remove

Apply

Specialist name

Specialist Id

Patient name

Patient Id

Id

Column

<

Consultations

View

Insert

Update

Remove

Apply

Patient name

Patient Id

Id

<

Exemple de cod

```
1 using System;
2 using System.Collections.Generic;
3 using System.Data;
4 using System.Data.OracleClient;
5 using System.Windows.Forms;
6
7 namespace MDBStefanAlexandra1306B
8 {
9     4 references
10     public partial class SConsultations : Form
11     {
12         string c_ = "DATA SOURCE = XE;" + "PERSIST SECURITY INFO = True; USER ID = mous; password = Soylover_3; Pooling = False;
13         OracleConnection oracleConnection = new OracleConnection();
14
15         List<String> listaSpec = new List<String>();
16         List<String> listaData = new List<String>();
17         List<String> listaOra = new List<String>();
18
19     2 references
20     private void fillIdSpecialist()
21     {
22         oracleConnection.ConnectionString = c_;
23         oracleConnection.Open();
24         try
25         {
26             if (specialistsTab.SelectedTab == specialistsTab.TabPages["insertPage"])
27             {
28                 listaSpec.Clear();
29                 idSpec.Items.Clear();
30                 idSpec.Text = "";
31                 string cmd = "SELECT id_spec FROM Specialist WHERE nume_spec = " + numeSpec.Text.Trim().ToString() + "";
32                 OracleCommand cmdatabase = new OracleCommand(cmd, oracleConnection);
33                 OracleDataReader myReader = cmdatabase.ExecuteReader();
34                 while (myReader.Read())
35                 {
36                     string sname = myReader.GetInt64(0).ToString();
37                 }
38             }
39         }
40     }
41 }
```

```
42         else if (specialistsTab.SelectedTab == specialistsTab.TabPages["updatePage"])
43         {
44             listaSpec2.Clear();
45             idSpec2.Items.Clear();
46             idSpec2.Text = "";
47             string cmd = "SELECT id_spec FROM Specialist WHERE nume_spec = " + numeSpec2.Text.Trim().ToString() + "";
48             OracleCommand cmdatabase = new OracleCommand(cmd, oracleConnection);
49             OracleDataReader myReader = cmdatabase.ExecuteReader();
50             while (myReader.Read())
51             {
52                 string sname = myReader.GetInt64(0).ToString();
53                 idSpec2.Items.Add(sname.ToString());
54             }
55         }
56     }
57     catch (Exception ex)
58     {
59         MessageBox.Show("Failed! The table might not exist! Check again!");
60     }
61     finally
62     {
63         oracleConnection.Close();
64     }
65 }
```

