Universitatea Tehnică "Gheorghe Asachi" din Iași
Facultatea de Automatică și Calculatoare
Specializarea Tehnologia Informației

GESTIONAREA UNUI CENTRU DE RECOLTARE DE SÂNGE

(pentru analize)

Coordonator,

Cătălin Mironeanu

Student,

Mihalache Nicoleta-Ecaterina

Iași, 2022

Titlu proiect: Gestionarea unui centru de recoltare de sânge (pentru analize)

Analiza, proiectarea și implementarea unei baze de date și a aplicației aferente care să modeleze activitatea unui centru de recoltare de sânge cu privire la gestiunea analizelor realizate, a pacienților și a personalului medical.

Descrierea cerințelor și modul de organizare al proiectului

Volumul mare de informații existente în cazul unui centru de recoltare de sânge cu numeroși pacientidetermină necesitatea fluidizării fluxurilor de date, gestiunea acestora fiind o adevărată provocare.

Activitatea de gestiune a unui centru de recoltare de sânge implică o muncă intensă în ceea ce privește numeroasele documente ce țin de înregistrarea pacienților, a analizelor pe care aceștia le fac, de personalul medical, precum și de realizarea analizelor în urma cărora cabinetul medical oferă pacienților rezultatul acestora.

Informațiile de care avem nevoie sunt cele legate de:

-**pacienți**: datele referitoare la pacienți se vor stoca în două tabele: Pacienti și Detalii_Pacienti. În tabela Pacienți se vor stoca: numele, prenumelepacientului și persoana de care este responsabil acesta (aparținătorul), iar în tabela Detalii_Pacienti vor fi stocate datele referitoare la: data nașterii, CNP-ul, genul, telefonul, adresa de email al acestuia, precum și legătura dintre cele două tabele.

-**analize**: în cazul informațiilor legate de analize, datele se vor stoca în două tabele: Analize și Detalii_Analize. Astfel, în tabela Analize vom stoca informații legate de categoria analizei, data când a fost recoltată, data rezultatului, id-ul medicului ce validează analiza și id-ul asistentei ce recoltează proba de sânge, urmând ca în tabela Detalii_Analize să fie stocate informații legate de numele analizei ce face parte dintr-o anumită categorie din tabela Analize, valoarea rezultată, intervalul de referință, id-ul de legătură cu tabela Analize și valabilitatea ei.

-**angajații**: în cazul informațiilor legate de angajați, informațiile se vor stoca în două tabele: Asistente și Medici. Astfel, în tabela Asistente vom stoca informații legate de asistentele responsabile de recoltarea probelor de sânge de la pacienți: numele și prenumele lor, iar în tabela Medici vom stoca informațiile legate de medicii responsabili de validarea rezultatelor: numele, prenumele acestora și codul de parafa.

Descrierea funcțională a aplicației

Principalele funcții care se pot întâlni într-un centru de recoltare de sânge sunt:

- Evidența pacienților
- Evidența angajaților
- Evidența analizelor efectuate

Descrierea detaliată a entităților și a relațiilor dintre tabele

Tabelele din această aplicație sunt:

- Analize;
- Detalii_Analize;
- Pacienti;
- Detalii_Pacienti;
- Asistente;
- Medici;
- Pacienti_Analize_FK;

În proiectarea acestei baze de date s-au identificat tipurile de **relații** 1:1, 1:n, n:n.

Între tabelele **Analize** și **Detalii_Analize** este o relație **one-to-many** deoarece o analiză are mai multe detalii (o categorie are mai multe tipuri de analize în construcția sa), iar un set de detalii este deținut de o singură analiză (o singură categorie). Legătura dintre cele două este data de câmpul **Analize_id_analiza**.

Între tabelele **Analize** și **Pacienți** este o relație **many-to-many** deoarece o analiză pot fi făcută de unul sau mai mulți pacienți, iar un pacient poate face una sau mai multe analize. Pentru ca tabela să se afle în 3FN această relație se va sparge în două, rezultând două relații **1:n,** iar legătura dintre cele două se va realiza cu ajutorul unei alte tabele **Pacienti_Analize_FK** care va conține cheia primară a fiecărei tabele. Altfel spus legătura se face prin două câmpuri **Pacienti_id_pacient** și **Analize_id_analiza** reunite într-o tabelă comună.

Între tabelele **Pacienți** și **Detalii_Pacienti** este o relație **one-to-one** deoarece un pacient pot avea un singur set de detalii, iar un set de detalii este corespondent unui singur pacient. Legătura dintre cele două tabele se face prin câmpul **Pacienti_id_pacient.**

Între tabelele **Analize** și **Medici** este o relație **many-to-one** deoarece una sau mai multe analize pot fi validate de un medic, iar un medic poate valida una sau mai multe analize. Legătura dintre cele două se va realiza prin câmpul **Medici_id_medic**.

Între tabelele **Analize** și **Asistente** este o relație **many-to-one** deoarece una sau mai multe analize pot fi recoltate de o asistentă, iar o asistentă poate recolta una sau mai multe analize. Legătura dintre cele două se va realiza prin câmpul **Asistente_id_asistenta**.

În proiectarea acestei baze de date s-au identificat următoarele tipuri de **constrângeri de integritate referențială**: primary key și foreign key, precum și **alte tipuri** de constrângeri: check, unique, not null.

Primary key-urile sunt generate de baza de date prin mecanismul de tip **autoincrement** și sunt atribuite tuturor id-urilor din tabele, identificând în mod unic fiecare înregistrare dintr-un tabel. Ele trebuie să conțină doar valori unice și nu pot avea valori NULL, iar un tabel poate avea doar o singură cheie primară. Aici avem totodată și constrângerile de tip **unique** și **not null**.

Constrângerea **unique** este atașată atributului CNP din tabela Pacienți, deoarece nu pot exista doi pacienți cu același CNP. Punând această constrângere impunem ca toate CNP-urile introduse în baza de date să fie unice.

Foreign key-urile sunt utilizate pentru a lega două tabele, fiind utilizate în toate relațiile dintre tabelele bazei de date. Acestea sunt câmpuri din tabele care se referă la Primary key-ul dintr-un anumit tabel. Pot exista și mai multe foreign key-uri și un exemplu este în cazul în care avem o relație many-to-many. Aceasta este spartă în alte două relații one-to-many rezultând astfel 2 foreign key-uri. Un exemplu este apariția tabelei copil **Pacienti_Analize_FK** între tabelele **Pacienti** și **Analize**, având 2 foreign key-uri **Pacienti_id_pacient** și **Analize_id_analiza**.

Constrângerea de tip **check** este utilizată pentru: atributul telefon din tabela Detalii_Pacienti deoarece se impune un anumit format pentru numarul de telefon pe care îl poate avea pacientul (să conțină 10 cifre), atributul email din aceeași tabelă pentru a impune un anumit format, nume și prenume din tabelele Pacienti, Asistente și Medici ca sa aiba dimensiunea mai mare decat 1, la fel și pentru câmpurile nume_analiza și categorie din tabelele Analize, respectiv Detalii_Analize.

Aceasta constrângere mai este utilizată pentru a impune unei valori să aparțină unei liste de valori existente.Un exemplu este dat de către atributul gen din tabela Detalii_Pacienți a cărei valoare trebuie să existe în lista de valori data, dacă este femeie gen=F și dacă este bărbat gen=B.

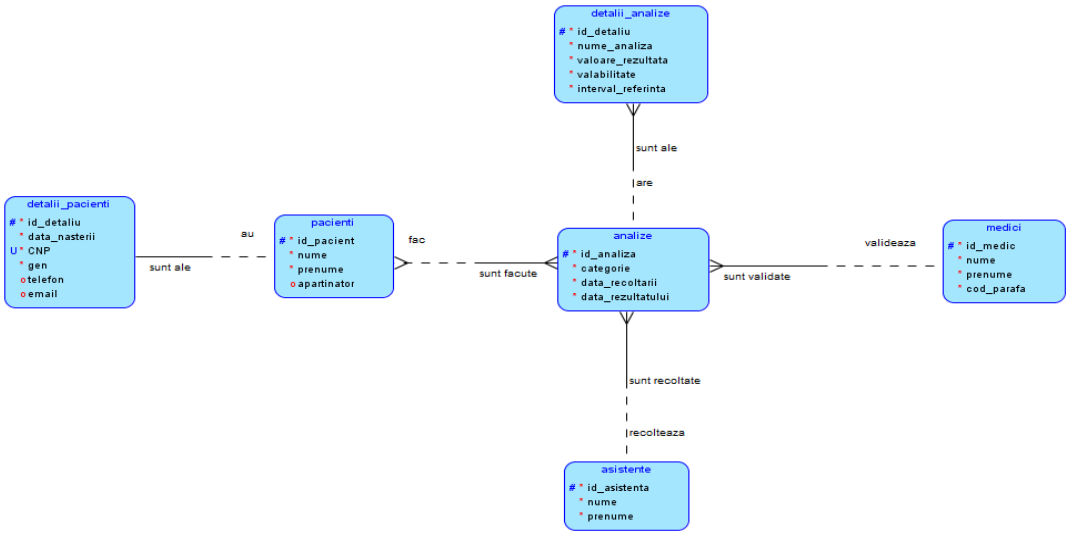

Fig.1 Model logic

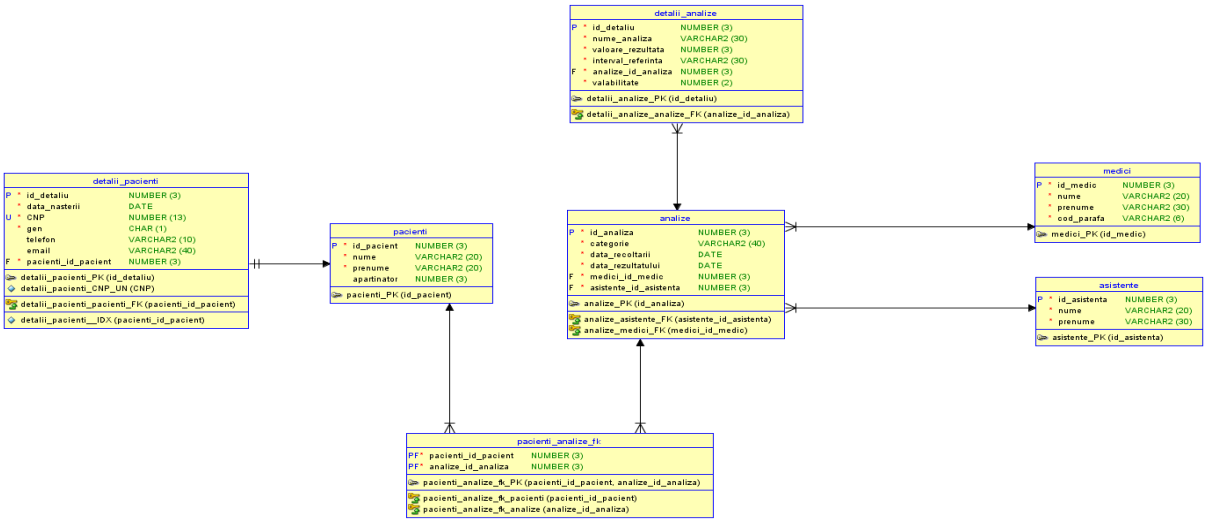

Fig.1 Model relational

Descrierea cazurilor de test:

Acest proiect conține o tranzacție ce urmărește inserarea a doi pacienți cu același număr de telefon. Se va insera primul pacient, iar la inserarea celui de-al doilea se verifică dacă are același număr de telefon ca primul: dacă da, atunci se va face o actualizare asupra câmpului aparținător din cadrul primului pacient, cu id-ul celui de-al doilea pacient. Astfel, dacă avem un pacient cu câmpul aparținător diferit de null, știm că are un apărținător ce se poate găsi la id-ul ăla.

Un caz de test este descris atunci când se șterge un pacient, al cărui id este în câmpul aparținător al altui pacient. În acest caz se va activa un trigger ce va modifica valoarea câmpului aparținător pe null.

O altă situație prezentată este atunci când un pacient, al cărui id este în câmpul aparținător al altui pacient își modifică numărul de telefon, astfel nu mai avem doi pacienți cu același număr de telefon. În acest caz se va activa un trigger ce va modifica valoarea câmpului aparținător pe null.

Cazuri de test sunt și la inserarea datelor în baza de date: se testează cu valori intenționat greșite funcționaliatea excepțiilor și a triggerilor.

Cazuri de test sunt și la ștergerea unui pacient, a unei analize sau a unui pacient cu toate analizele sale. Când vreau să șterg un pacient se șterg automat și toate detaliile acestuia; același lucru se întâmplă și cu ștergerea unei analize. Când vreau să șterg un pacient cu analizele sale se șterg automat: detaliile pacientului, analizele făcute, detaliile analizelor făcute și pacientul.

Cazuri de test sunt evidențiate și în actualizarea anumitor câmpuri: numele unui pacient, adresa de email, telefonul.

Un alt caz de test este evidențiat la inserarea unei noi analize (o analiză ce face parte dintr-o anumită categorie). În acest caz se verifică dacă s-a mai făcut acea analiză recent. Se verifică dacă diferența dintre datele de recoltare a probelor este mai mare decât teremenul de valabilitate a analizei și dacă este mai mare atunci se poate realiza o nouă analiză de acest tip, altfel se activează un trigger iar inserarea analizei nu este posibilă.

Descrierea logicii stocate:

Proiectul conține mai multe pachete ce conturează tranzacția: un pachet ce conține mai multe proceduri de inserare (inserare asistentă, medic, pacient, detalii pacient, analiză, detalii analiză), un pachet ce conține mai multe proceduri de actualizare (actualizarea numelui unui pacient, telefonului unui pacient, aparținătorul unui pacient, email-ul unui pacient, valabilitatea unei analize), un pachet ce conține mai multe proceduri de ștergere (ștergerea unui pacient cu

detaliile sale, ștergerea unui pacient cu analizele sale, ștergerea unei analize cu detaliile sale) și un pachet ce conține mai multe proceduri de afișare a datelor (afișarea unui pacient cu detaliile sale, afișarea unei analize cu detaliile sale, afișarea pacienților, afișarea analizelor etc.).

Există o funcție de validare a datelor unei analize: data rezultatului unei analize trebuie să fie mai mare decât data de recoltare a acesteia.

Aplicația conține 4 triggeri descriși și în cazurile de test de mai sus: un trigger pentru validarea datelor, un trigger pentru verificarea valabilității unei analize înainte de inserarea acesteia, un trigger pentru modificarea câmpului aparținător pe null atunci când se șterge un pacient cu id-ul respectiv și un trigger pentru modificarea câmpului aparținător pe null atunci când se modifică numărul de telefon al pacientului al cărui id este în câmpul aparținător al altui pacient.

Pentru realizarea triggerilor și a procedurilor de afișare s-au folosit cursori expliciți.