DISCIPLINA BAZE DE DATE PROIECT

Gestiunea activității gării Iași

Coordonator,

Prof. Mironeanu Cătălin

Student,

Patap Alexandra

Grupa 1308A

Titlul proiectului

Gestiunea activității gării Iași

Descrierea proiectului (scopul aplicației)

Se studiază modul în care funcționează o gară pentru a modela o bază de date eficientă care să răspundă nevoilor sale. O gară are zeci de angajați, care sunt necesari pentru operarea eficientă a numeroaselor mersuri de tren. Este esențial să se urmărească datele despre mersul trenurilor și despre angajații care lucrează acolo. O gară gestionează și informații despre pasageri și este esențial să se poată accesa detalii despre acești oameni pentru a asigura fluiditatea activității gării.

Scopul aplicației constă în următoarele funcții:

- Monitorizarea plecărilor și sosirilor trenurilor
- Evidența numărului de locuri libere pentru fiecare tren
- Evidența pasagerilor și a rezervărilor lor
- Urmărirea corespondenței pasager-tren-loc
- Gestionarea și reținerea informațiilor referitoare la angajații fiecărui tren, departamentele de apartenență și funcțiile asociate
- Selectarea și evidențierea sumelor obținute din vânzarea biletelor pentru fiecare cursă de tren
- Evidența numărului de trenuri deținute de fiecare companie în fiecare gară

Descrierea entităților și a relațiilor dintre tabele

Tabelele din această aplicație sunt:

- *Gara*
- *Departament*
- *Angajat*
- *Detalii_angajat*
- *Tren*
- *Mers_trenuri*
- *Pasager*
- *Rezervare*

Descrierea coloanelor din tabele:

În tabela **gara,** am atribute *id_gara*, ce implementează un mecanism de auto-incrementare și *oraș* care îmi arată în ce orașe sunt gările din care pot veni cu trenul spre Iași.

În tabela **departament,** am atributele *id_departament*, ce implementează un mecanism de auto-incrementare; *nume_departament* și *id_gara* care îmi specifică ce departamente există în gările respective.

În tabela **angajat**, am atributele *cnp_angajat* , prin care se identifică unic fiecare angajat; *nume și prenume*, numele și prenumele angajatului, *meserie* pentru a vedea cu ce se ocupă în cadrul departamentului dat de atributul *id_departament*.

În tabela **detalii_angajat**, am atributele *cnp_angajat* pentru a putea face legătură cu angajatul căruia vreau să adaug detaliile; *nr_telefon* și *data_nasterii*.

În tabela **tren**, am atributele *nr_tren* , un număr din 4 cifre ce identifică unic un tren; *companie* , compania de care aparține trenul; *capacitate*, capacitatea maximă a trenului; *id_gara* să știm dacă este tren de plecare sau de sosire în Iași și *cnp_angajat*, ce angajat lucrează în trenul respectiv.

În tabela **mers_trenuri**, am atributele *id_mers*, ce identifică în mod unic traseul trenului și implementează un mecanism de auto-incrementare; *data*, data calendaristică la care va merge trenul; *locuri_ocupate*, numărul de locuri ocupate în tren ce este actualizat cu fiecare nouă rezervare făcută și care trebuie să fie mai mic decât capacitatea trenului; *peron*, numărul peronului de la care pleacă/pe care vine trenul și *nr_tren*, numărul trenului ce va circula atunci.

În tabela **pasager**, am atributele *cnp_pasager*, ce identifică în mod unic un pasager; *nume* și *prenume* , numele si prenumele pasagerului și *reducere*, este o valoare numerică de lungime 3 ce specifică dacă pasagerul are un tip de reducere pentru biletul la tren, 0 înseamnă că nu are reducere și 100 că are bilet gratuit.

În tabela **rezervare**, am atributele *nr_rezervare*, ce implementează un mecanism de auto-incrementare; *data_rezervare*, este data la care s-a făcut rezervarea și nu poate fi o dată din viitor, *nr_vagon* și *nr_loc*, identifică împreună locul în trenul la care fac rezervare, *plata*, inițial este o valoare care mai apoi se actualizează în funcție de reducerea pasagerului, *cnp_pasager*, pentru a ști a cui pasager este rezervarea și *id_mers*, pentru a ști la ce mers al trenului face pasagerul rezervare. O inserare în tabela **rezervare** se face printr-o tranzacție, fiind necesare o inserare în tabela **rezervare**, un update în tabela **rezervare** și un update în tabela **mers_trenuri**. După ce fac inserarea rezervării, verific la ce mers de tren am făcut rezervare și actualizez în tabela mers_trenuri , numărul locurilor ocupate. Actualizarea în tabela **rezervare** este necesară pentru a modifica plata pasagerului în funcție de reducerea pe care o are acesta la biletele de tren.

Normalizarea:

În cazul de față, avem:

- Prima formă normală: aplicată la toate tabelele, fiecare coloană neavând grupuri de date sau valori multiple;
- A doua formă normală: toate tabelele se află în această formă deoarece se află în prima formă normală și toate atributele non-cheie depind doar de atributul cheie aferent;
- A treia formă normală: toate tabelele se află în această formă deoarece se află în a doua formă normală și nu există dependențe tranzitive.

Detalii despre relațiile dintre entitățile din proiect:

În proiectarea acestei baze de date am folosit tipurile de relații **1:n**, **n:1** și **1:1**.

Între tabelele **gara** și **departament** se întâlnește o **relație 1:n**, deoarece o gară poate avea mai multe departamente, dar același departament nu poate aparține mai multor gări, departamentele fiind distinctele între ele , chiar dacă poartă același nume.

Departamentele au în subordonare mai mulți angajați. Astfel între tabelele **departament** şi **angajat** este o **relație 1:n,** angajații pot lucra doar într-un departament.

Fiind oameni individuali şi diferiţi , angajaţii au propriile detalii distincte faţă de a unui alt angajat. Între tabelele **angajat** şi **detalii_angajat** există o **relaţie 1:1**.

Trenurile au în alcătuire şi gara de unde pleacă. Raportându-ne la gara Iaşi, gara de bază, trenurile ce au în componenţă o gară diferită de Iaşi fac parte din trenurile de sosire, iar cele ce conţin gara Iaşi, fac parte din trenurile de plecare. Astfel între tabelele **gara** şi **tren** este o **relaţie 1:n,** deoarece o gară poate deţine mai multe trenuri, dar un tren nu poate fi deţinut şi să aibă mai multe gări de plecare în acelaşi timp.

Trenurile sunt conduse de un angajat şi un angajat poate conduce mai multe trenuri, astfel **relaţia** dintre tabelele **tren** şi **angajat** este **n:1**.

Un tren poate pleca de mai multe ori din aceiaşi gară, în zile diferite, având astfel mai multe mersuri, iar un mers de tren aparține unui singur tren, **relaţia** dintre tabelele **tren** şi **mers_trenuri** este **1:n**.

Mersurile trenurilor deţin mai multe rezervări, iar o rezervare aparţine unui mers de tren. **Relația** între tabelele **mers_trenuri** şi **rezervare** este **1:n**.

Pasagerii pot face mai multe rezervări pentru un mers de tren, iar rezervarea este făcută de o singură persoană o dată, **relaţia** dintre tabelele **rezervare** şi **pasager** fiind **n:1**.

Constrângerile folosite:

a. Constrângeri de tip Primary Key

Fiecare tabel conține un câmp de tip index care respectă o constrângere de tip Primary Key. Tabelele **gara**, **departament**, **mers_trenuri** și **rezervare** conțin un câmp de tip index cu auto-incrementare care respectă o constrângere de tip Primary Key.

b. Constrângeri de tip Foreign Key

Aceste constrângeri sunt utilizate în crearea diferitelor tipuri de relații între tabele:

- Relații **1:1** în **detalii_angajat** (s-a moștenit atributul *cnp_angajat* din tabela **angajat**) ;

- Relații **1:n** în **departament**(*id_gara*), **angajat**(*id_departament*), **tren**(*id_gara*, *cnp_angajat*), **rezervare**(*cnp_pasager*, *id_mers*), **mers_trenuri**(*nr_tren*).

c. Constrângeri de tip Not Null

Acest tip de constrângere a fost folosit pentru a specifica ce atribute sunt esențiale și obligatorii pentru o entitate. Toate atributele din toate tabelele respectă constrângerea de tip Not Null.

d. Constrângeri de tip Unique

Atributele ce respectă constrângerea de tip Primary Key , respectă implicit și constrângerea de tip Unique. Înafară de acestea, în baza de date există un atribut care necesită să fie unic: în tabela **detalii_angajat**, atributul *nr_telefon*.

e. Constrângeri de tip Check

Acest tip de constrângere a fost folosit pentru a permite introducerea de date calendaristice mai mari ca ziua curentă pentru entitatea **mers_trenuri** (atributul data) și date calendaristice mai mici ca ziua curenta pentru entitatea **rezervare** (atributul data_rezervare). O altă utilizare a constrângerii de tip Check a fost pentru verificarea ca un angajat să aibă vârsta de 18 ani împliniți, în tabela **detalii_angajat**(atribut *data_nasterii*).

O altă utilizare a constrângerii de tip Check este pentru a asigura că datele sunt introduse într-un format specific:

- pentru a verifica că numele și prenumele persoanelor conțin doar litere în tabelele **angajat** (atributele *nume, prenume*) și **pasager** (atributele *nume, prenume*);
- Pentru a verifica că numele orașului conține doar litere și spații în tabela **gara** (atributul *oras*).

În tabela **mers_trenuri**, am avut o constrângere de tip Check care verifica atributul *locuri_ocupate* să fie mai mic față de atributul *capacitate* din tabela **tren**. Această constrângere s-a transformat doar în trigger ce verifică să nu se depășească capacitatea trenului.

Diagrama logică:

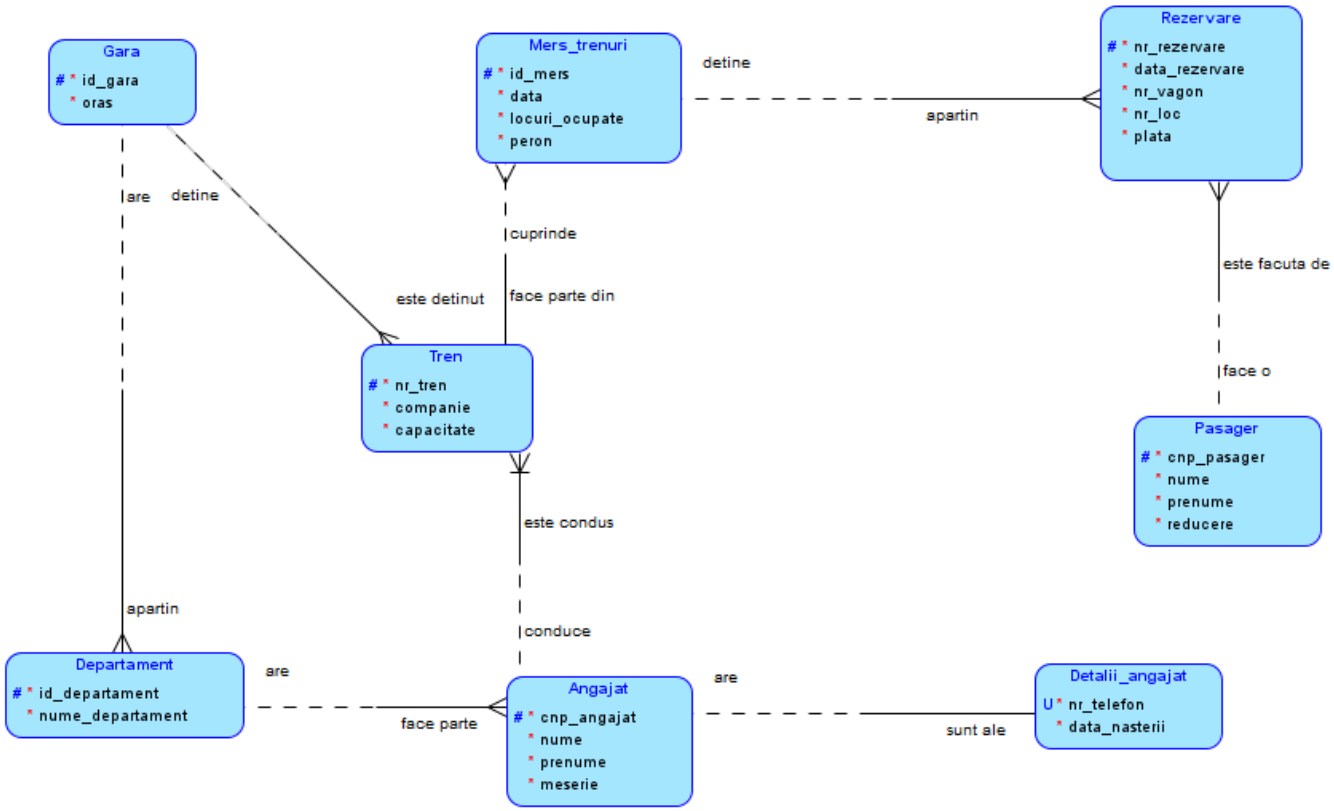

Diagrama relațională: